

**Universidad Nacional de San Agustín de Arequipa**  
**Referente Latinoamericano**



**Facultad de Ingeniería de Producción y Servicios**  
Escuela Profesional de Ingeniería de Telecomunicaciones

**SOFTWARE DE TELECOMUNICACIONES**  
**PRÁCTICA 2: MI PRIMER PROYECTO EN LARAVEL**

**Presentado por:**

**Docente:**

Ing. Efraín Tito Mayhua Lopez

Barrera Begazo, Raquel Yoselin

Quispe Pocohuanca, Gabriel Rodrigo

Rosas Huaihua, Derek Joe

Ticona Ylaquijo, Rony Brayan

**2024**

**I.****RESUMEN**

Este informe presenta un análisis exhaustivo sobre el desarrollo de una aplicación web utilizando Laravel, un framework de PHP. El proyecto fue creado con el apoyo de herramientas como XAMPP, que proporciona un entorno local con Apache y MySQL, y Composer para la gestión de dependencias. A lo largo del informe, se describen los pasos detallados que siguieron desde la instalación del entorno de desarrollo hasta la implementación de funcionalidades clave como el enrutamiento, la creación de controladores, vistas y la interacción con una base de datos. Además, se documentan los desafíos encontrados durante el proceso de desarrollo y las soluciones aplicadas, tomando en cuenta las mejores prácticas para garantizar la seguridad y eficiencia del sistema.

Palabras clave — Laravel, PHP, XAMPP, MySQL, Composer, Sublime Text, Migraciones, Modelo-Vista-Controlador (MVC).

**II.****ABSTRACT**

This report provides an exhaustive analysis of the development of a web application using Laravel, a PHP framework. The project was created with the support of tools such as XAMPP, which provides a local environment with Apache and MySQL, and Composer for dependency management. Throughout the report, detailed steps are described, from installing the development environment to implementing key features such as routing, creating controllers, views, and interacting with a database. Additionally, challenges encountered during the development process and the solutions applied are documented, considering best practices to ensure system security and efficiency.

Keywords — Laravel, PHP, XAMPP, MySQL, Composer, Sublime Text, Migrations, Model-View-Controller (MVC).

## Índice

<b>I. RESUMEN</b>	<b>2</b>
<b>II. ABSTRACT</b>	<b>2</b>
<b>III. INTRODUCCIÓN</b>	<b>4</b>
A. Objetivos . . . . .	4
<b>IV. MARCO TEÓRICO</b>	<b>4</b>
A. Laravel . . . . .	4
B. PHP . . . . .	4
C. XAMPP . . . . .	4
D. Composer . . . . .	4
E. MVC (Modelo - Vista - Controlador) . . . . .	5
<b>V. DESARROLLO</b>	<b>5</b>
A. Preparación del Entorno . . . . .	5
1. Instalación de XAMPP . . . . .	5
2. Instalación de Composer . . . . .	6
3. Instalación de Laravel . . . . .	6
4. Edición del Proyecto . . . . .	7
B. Desarrollo del Proyecto . . . . .	9
1. Paso 1: Crear el primer proyecto . . . . .	9
2. Paso 2: Crear el Router . . . . .	10
3. Paso 3: Crear una Vista . . . . .	11
4. Paso 4: Crear un Controlador . . . . .	13
5. Configuración de la Base de Datos . . . . .	15
6. Paso 6: Crear Migraciones . . . . .	17
7. Paso 7: Crear un Modelo . . . . .	19
<b>VI. OBSERVACIONES Y CONCLUSIONES</b>	<b>23</b>
A. Observaciones . . . . .	23
B. Conclusiones . . . . .	23
<b>VII REFERENCIAS</b>	<b>24</b>

### III. INTRODUCCIÓN

Laravel es un framework que simplifica el desarrollo de aplicaciones web basadas en PHP. Permite gestionar tanto el backend como el frontend mediante una arquitectura estructurada y escalable. Este informe tiene como objetivo documentar el desarrollo de una aplicación web utilizando Laravel, siguiendo los principios del patrón Modelo-Vista-Controlador (MVC). A través del uso de herramientas como Composer y XAMPP, se lograron implementar rutas, controladores y vistas, además de conectar la aplicación con una base de datos MySQL. Se presentan los pasos seguidos en detalle, junto con las dificultades encontradas y las soluciones implementadas por el grupo de trabajo.

#### A. Objetivos

- Desarrollar una aplicación web funcional utilizando el framework Laravel.
- Implementar una arquitectura robusta basada en el patrón MVC.
- Configurar y utilizar herramientas como XAMPP, MySQL y Composer para optimizar el entorno de desarrollo.
- Documentar el proceso paso a paso, resolviendo problemas y adoptando mejoras.

### IV. MARCO TEÓRICO

#### A. Laravel

Laravel es un framework de PHP de código abierto que facilita el desarrollo de aplicaciones web mediante el uso de una arquitectura MVC. Esta estructura permite separar la lógica de negocio (controlador), la presentación (vista), y la gestión de datos (modelo), asegurando un código más limpio y fácil de mantener.

#### B. PHP

PHP es un lenguaje de programación ampliamente utilizado en el desarrollo web. Laravel, al estar basado en PHP, hereda sus ventajas, como la integración con múltiples sistemas de bases de datos y su capacidad para construir aplicaciones robustas.

#### C. XAMPP

XAMPP es un paquete de software libre que proporciona un entorno de servidor local con Apache y MySQL, ideal para el desarrollo de aplicaciones web. Su fácil instalación y configuración lo hacen perfecto para probar proyectos en un entorno controlado.

#### D. Composer

Composer es un gestor de dependencias para PHP, que permite incluir bibliotecas externas de forma sencilla en proyectos de Laravel. Facilita la gestión y actualización de las dependencias necesarias, asegurando la compatibilidad entre diferentes versiones.

## E. MVC (Modelo - Vista - Controlador)

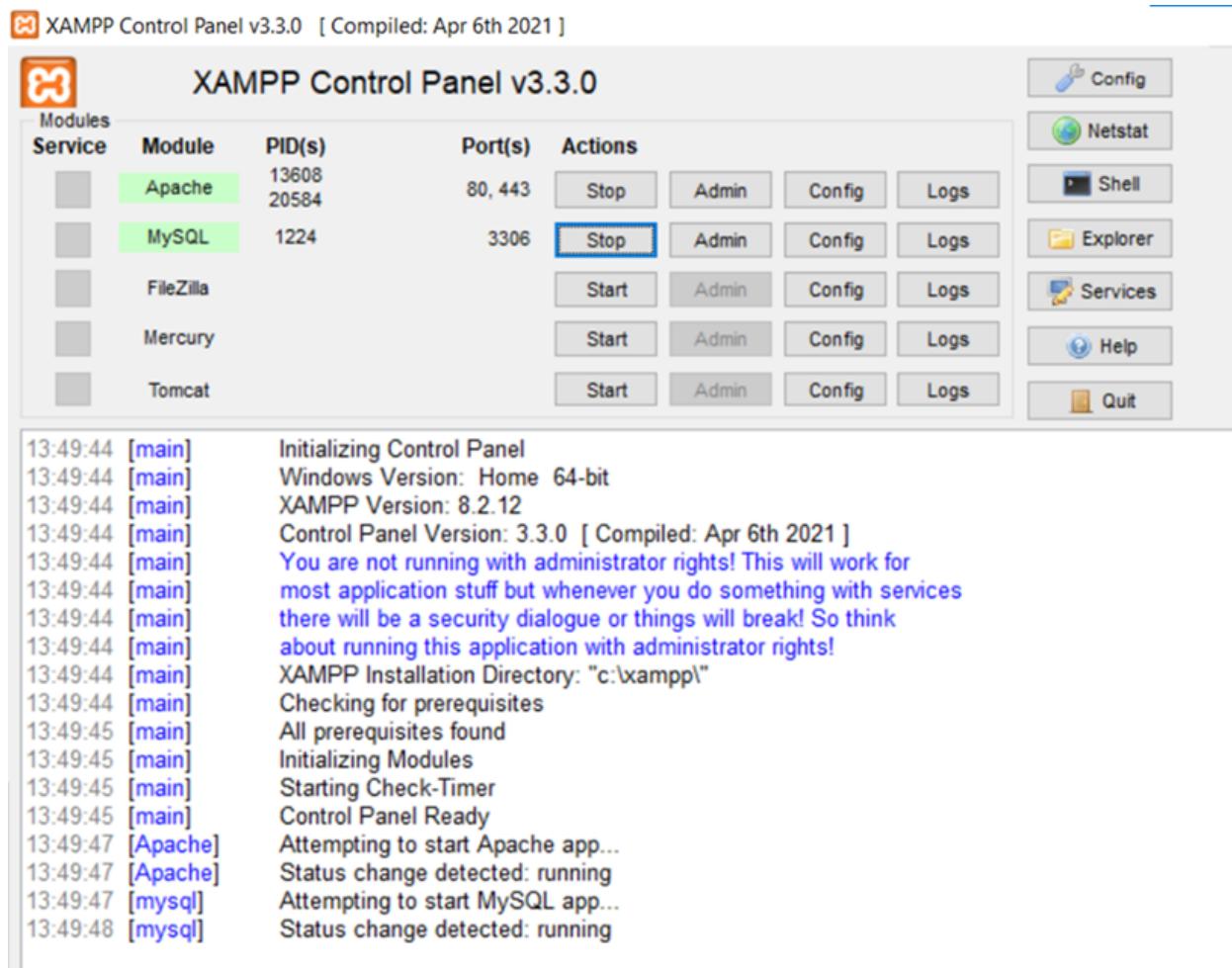
El patrón MVC es un enfoque arquitectónico que permite organizar una aplicación web dividiendo las responsabilidades en tres componentes: el modelo, la vista y el controlador. Esto facilita la gestión del código y su mantenimiento a largo plazo.

## V. DESARROLLO

### A. Preparación del Entorno

#### 1. Instalación de XAMPP

XAMPP se instaló para proporcionar un servidor local. Desde su panel de control, se activaron los servicios de Apache y MySQL para poder ejecutar la aplicación Laravel y gestionar la base de datos.



**Figura 1:** XAMPP configurado con Apache y MySQL activos

## 2. Instalación de Composer

Composer fue instalado para gestionar las dependencias de Laravel. Después de la instalación, se verificó su funcionamiento mediante el comando `composer -v`.

```
Símbolo del sistema
D:\Webs>composer -v

[Progress Bar]
Composer version 2.7.9 2024-09-04 14:43:28

Usage:
  command [options] [arguments]

Options:
  -h, --help          Display help for the given command. When no command is given display help for the
  -q, --quiet         Do not output any message
  -V, --version       Display this application version
  --ansi|--no-ansi   Force (or disable --no-ansi) ANSI output
  -n, --no-interaction Do not ask any interactive question
  --profile          Display timing and memory usage information
  --no-plugins        Whether to disable plugins.
  --no-scripts        Skips the execution of all scripts defined in composer.json file.
  -d, --working-dir=WORKING-DIR If specified, use the given directory as working directory.
  --no-cache          Prevent use of the cache
  -v|vv|vvv, --verbose Increase the verbosity of messages: 1 for normal output, 2 for more verbose output
```

**Figura 2:** Verificación de la instalación de Composer

### 3. Instalación de Laravel

Con Composer, se creó el proyecto con el comando `composer create-project --prefer-dist laravel/laravel revistapp`, lo que generó una nueva estructura de aplicación en el directorio **revistapp**.

```
[Símbolo del sistema] Símbolo del sistema

D:\Webs>composer create-project laravel/laravel revistapp
Creating a "laravel/laravel" project at "./revistapp"
Installing laravel/laravel (v11.2.0)
  - Installing laravel/laravel (v11.2.0): Extracting archive
Created project in D:\Webs\revistapp
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 106 installs, 0 updates, 0 removals
  - Locking brick/math (0.12.1)
  - Locking carbonphp/carbon-doctrine-types (3.2.0)
```

**Figura 3:** Creación del proyecto

```

Símbolo del sistema
- Installing sebastian/cli-parser (3.0.2): Extracting archive
- Installing phpunit/php-timer (7.0.1): Extracting archive
- Installing phpunit/php-template (4.0.1): Extracting archive - Installing phpunit/php-inv
- Installing phpunit/php-file-iterator (5.1.0): Extracting archive - Installing theseer/tokeniz
- Installing sebastian/lines-of-code (3.0.1): Extracting archive
- Installing sebastian/complexity (4.0.1): Extracting archive
- Installing sebastian/code-unit-reverse-lookup (4.0.1): Extracting archive
- Installing phpunit/php-code-coverage (11.0.6): Extracting archive
- Installing phar-io/version (3.2.1): Extracting archive
- Installing phar-io/manifest (2.0.4): Extracting archive
- Installing myclabs/deep-copy (1.12.0): Extracting archive
- Installing phpunit/phpunit (11.4.0): Extracting archive
59 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

78 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 ..... 501.96ms DONE
INFO Running migrations.

0001_01_01_000001_create_users_table ..... 2s DONE
0001_01_01_000001_create_cache_table ..... 232.59ms DONE
0001_01_01_000002_create_jobs_table ..... 539.18ms DONE

D:\Webs>

```

Figura 4: Estructura del proyecto creado por Laravel

4.

## Edición del Proyecto

El proyecto fue abierto en Sublime Text para trabajar con el código y gestionar los archivos.

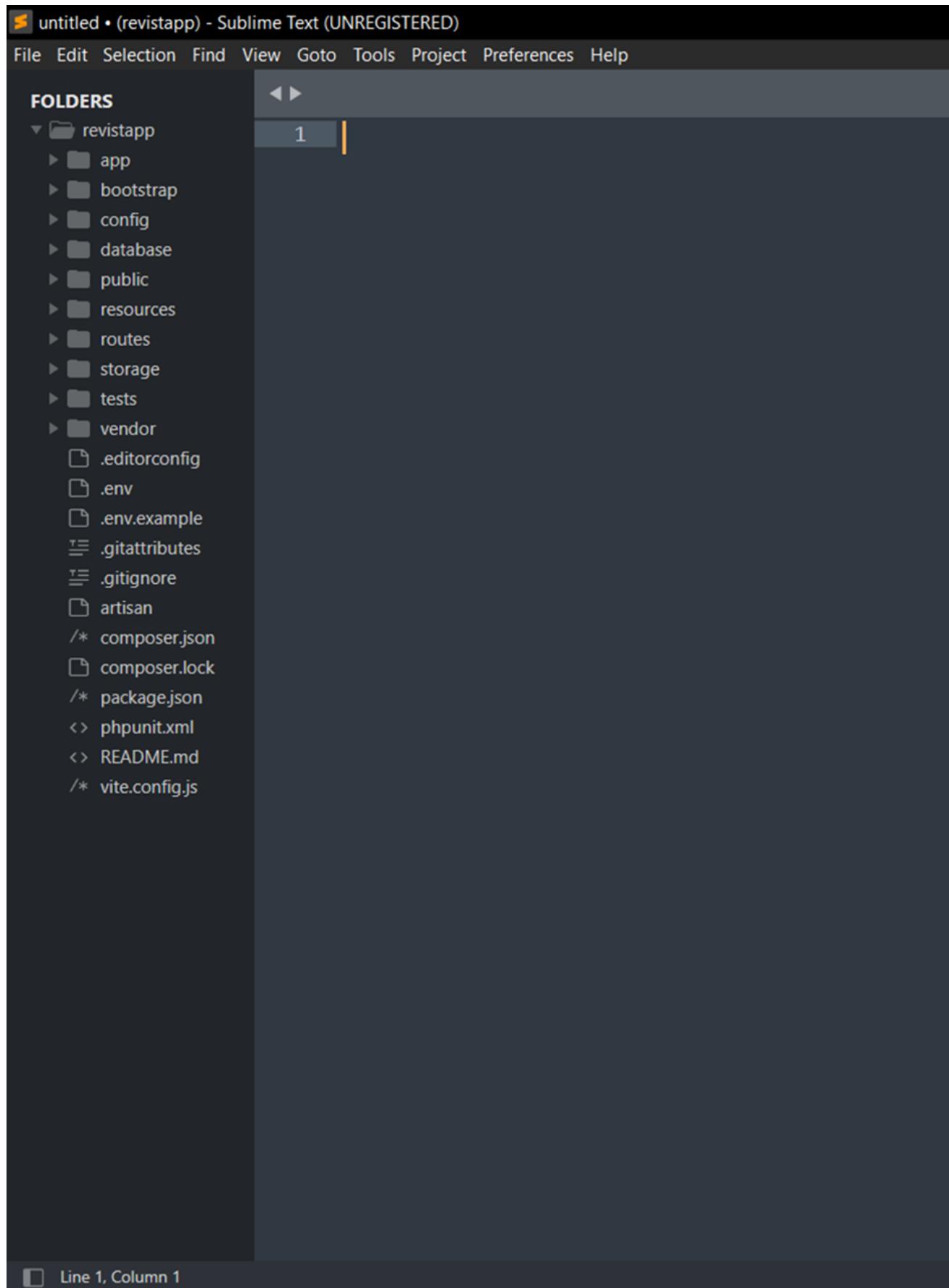


Figura 5: Proyecto abierto en Sublime Text

## B. Desarrollo del Proyecto

### 1. Paso 1: Crear el primer proyecto

Una vez instalado el entorno, se ejecutó el comando `php artisan serve` para levantar el servidor local de Laravel y visualizar la primera versión del proyecto en el navegador.

```
D:\Webs>php artisan serve
Could not open input file: artisan

D:\Webs>cd revistapp

D:\Webs\revistapp>php artisan serve

INFO Server running on [http://127.0.0.1:8000].
Press Ctrl+C to stop the server
```

Figura 6: Levantamiento del servidor local de Laravel

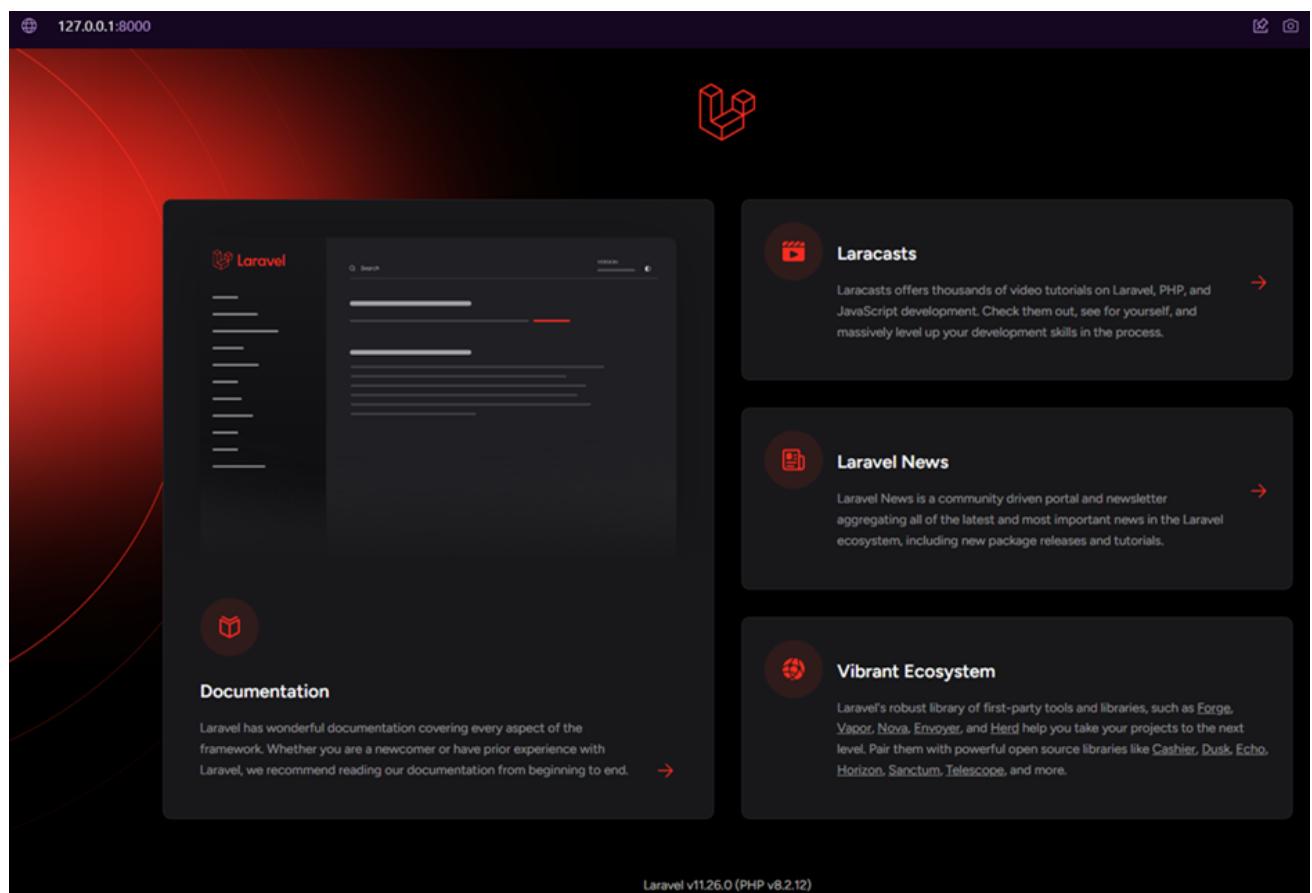
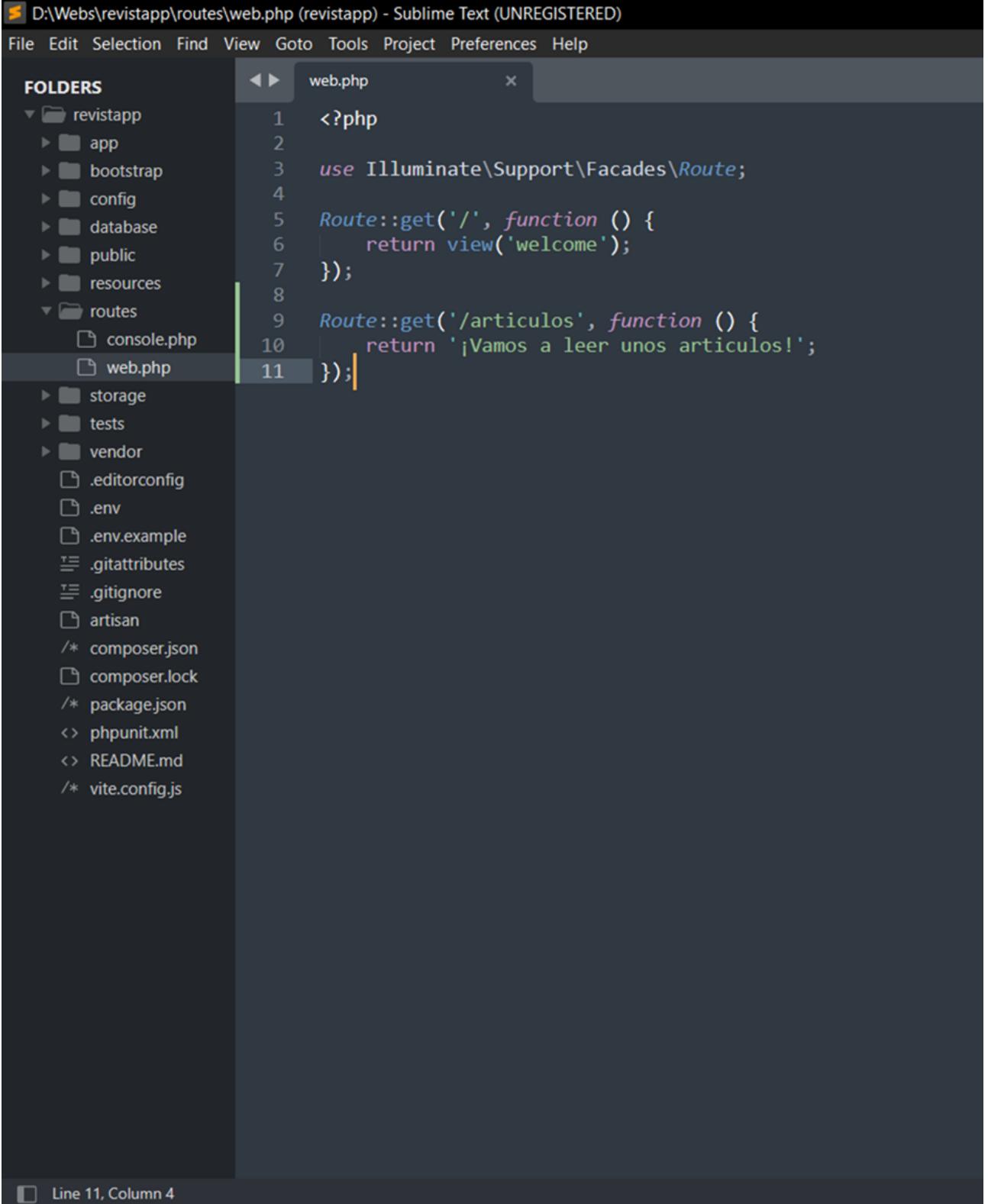


Figura 7: Primera visualización del proyecto en el navegador

## 2. Paso 2: Crear el Router

Se definieron las rutas en el archivo `routes/web.php`. Una de las rutas creadas fue `/articulos`, que enlaza con un controlador.



The screenshot shows the Sublime Text editor with the file `D:\Webs\revistapp\routes\web.php` open. The left sidebar displays the project structure:

- revistapp
- app
- bootstrap
- config
- database
- public
- resources
- routes
  - console.php
  - web.php
- storage
- tests
- vendor
- .editorconfig
- .env
- .env.example
- .gitattributes
- .gitignore
- artisan
- composer.json
- composer.lock
- package.json
- phpunit.xml
- README.md
- vite.config.js

The main pane contains the PHP code for the `web.php` route definition:`<?php  
use Illuminate\Support\Facades\Route;  
  
Route::get('/', function () {  
 return view('welcome');  
});  
  
Route::get('/articulos', function () {  
 return '¡Vamos a leer unos articulos!';  
});`

Line 11, Column 4

Figura 8: Definición de la ruta `/articulos` en el archivo `web.php`

### 3. Paso 3: Crear una Vista

Se crearon vistas utilizando Blade, el motor de plantillas de Laravel. Las vistas en Laravel son plantillas que contienen el código HTML.

The screenshot shows a Sublime Text window with the following details:

- File Path:** D:\Webs\revistapp\resources\views\articulos.blade.php (revistapp) - Sublime Text (UNREGISTERED)
- File Menu:** File Edit Selection Find View Goto Tools Project Preferences Help
- Folders:**
  - revistapp
    - app
    - bootstrap
    - config
    - database
    - public
    - resources
      - css
      - js
      - views
        - articulos.blade.php
        - welcome.blade.php
    - routes
      - console.php
      - web.php
    - storage
    - tests
    - vendor
      - .editorconfig
      - .env
      - .env.example
      - .gitattributes
      - .gitignore
      - artisan
      - composer.json
      - composer.lock
      - package.json
      - phpunit.xml
      - README.md
      - vite.config.js
- Content:**

```

1  <!-- vista almacenada en /resources/views/articulos.blade.php -->
2  <html>
3      <body>
4          <h1>¡Vamos a leer unos artículos!</h1>
5      </body>
6  </html>
7

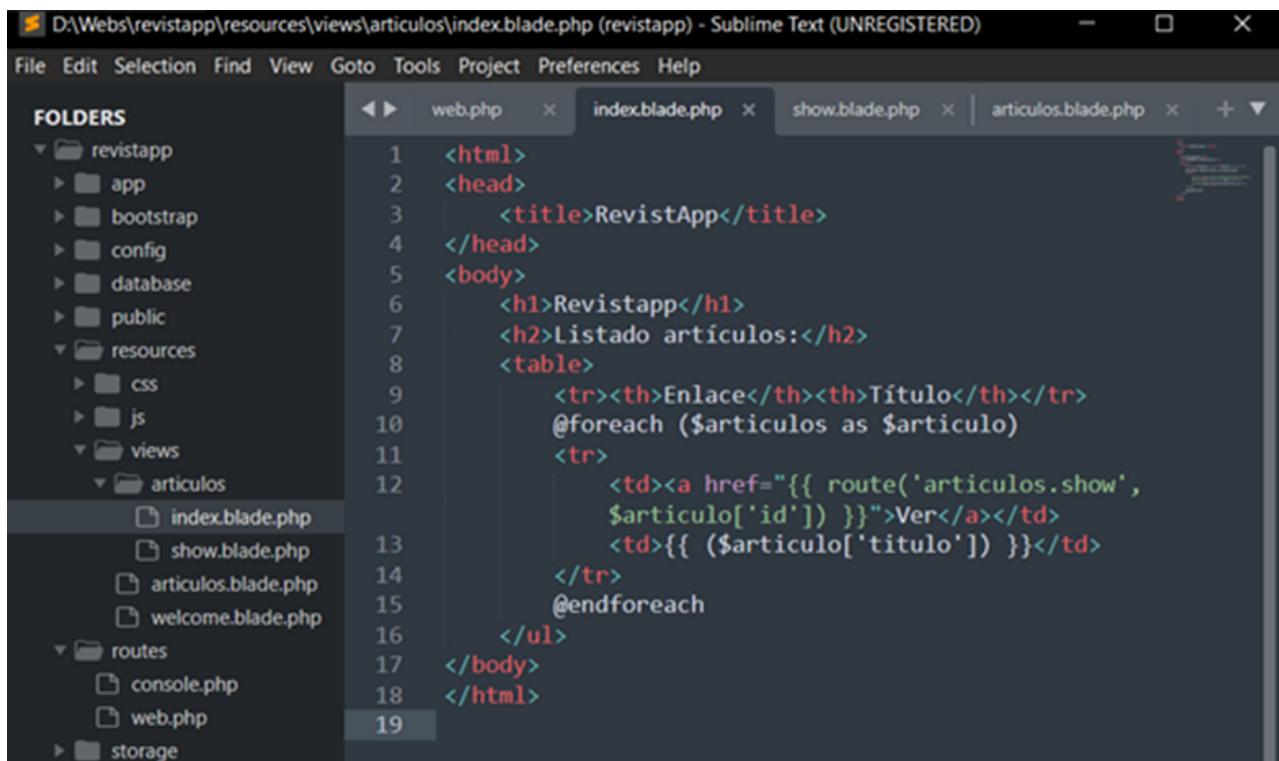
```
- Status Bar:** Line 7, Column 1

Figura 9: Vista *articulos.blade.php*



Figura 10: Primera vista en nuestro local host */articulos*

La vista principal *index.blade.php* muestra una lista de artículos en formato de tabla.



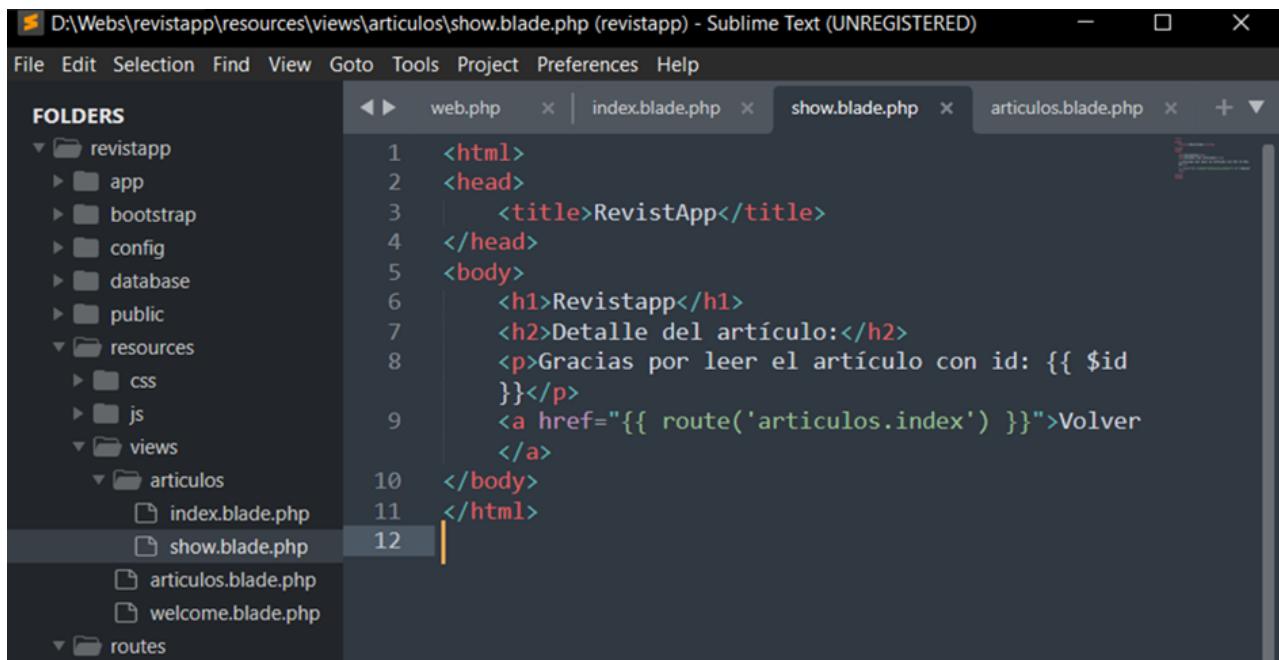
The screenshot shows the Sublime Text interface with multiple tabs open. The current tab is *index.blade.php*, which contains the following code:

```

1 <html>
2 <head>
3     <title>RevistApp</title>
4 </head>
5 <body>
6     <h1>Revistapp</h1>
7     <h2>Listado artículos:</h2>
8     <table>
9         <tr><th>Enlace</th><th>Título</th></tr>
10    @foreach ($articulos as $articulo)
11        <tr>
12            <td><a href="{{ route('articulos.show', $articulo['id']) }}>Ver</a></td>
13            <td>{{ ($articulo['titulo']) }}</td>
14        </tr>
15    @endforeach
16    </ul>
17 </body>
18 </html>
19

```

Figura 11: Vista *index.blade.php* con la lista de artículos



The screenshot shows the Sublime Text interface with multiple tabs open. The current tab is *show.blade.php*, which contains the following code:

```

1 <html>
2 <head>
3     <title>RevistApp</title>
4 </head>
5 <body>
6     <h1>Revistapp</h1>
7     <h2>Detalle del artículo:</h2>
8     <p>Gracias por leer el artículo con id: {{ $id }}</p>
9     <a href="{{ route('articulos.index') }}>Volver</a>
10    </body>
11 </html>
12

```

Figura 12: Vista *show.blade.php* con el id del artículo con la lista de artículos



# Revistapp

## Listado artículos:

Enlace	Título
<a href="#">Ver</a>	Primer artículo...
<a href="#">Ver</a>	Segundo artículo...
<a href="#">Ver</a>	Tercer artículo...

Figura 13: Segunda vista en nuestro local host /artículos



# Revistapp

## Detalle del artículo:

Gracias por leer el artículo con id: 1

[Volver](#)

Figura 14: Tercera vista en nuestro local host /artículos/1

4.

## Paso 4: Crear un Controlador

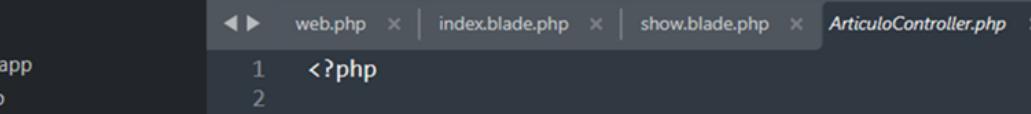
Ingresamos el siguiente comando en el CMD para crear el controlador:

```
Símbolo del sistema
2024-10-06 01:07:49 /articulos/1
2024-10-06 01:07:49 /favicon.ico
2024-10-06 01:07:51 /articulos ...
2024-10-06 01:07:51 /favicon.ico ...
^C
D:\Webs\revistapp>php artisan make:controller ArticuloController
[INFO] Controller [D:\Webs\revistapp\app\Http\Controllers\ArticuloController.php] created successfully.

D:\Webs\revistapp>
```

Figura 15: Creación del controlador en el CMD con artisan

Se generó un controlador llamado ArticuloController para manejar las peticiones HTTP relacionadas con los artículos.



The screenshot shows a Sublime Text window with the following details:

- Title Bar:** D:\Webs\revistapp\app\Http\Controllers\ArticuloController.php (revistapp) - Sublime Text (UNREGISTERED)
- Menu Bar:** File Edit Selection Find View Goto Tools Project Preferences Help
- Folders List:** revistapp (expanded), app (expanded), Http (expanded), Controllers (expanded), ArticuloController.php (selected), Controller.php, Models, Providers, bootstrap, config, database.
- File List:** web.php, index.blade.php, show.blade.php, ArticuloController.php (selected).
- Code Editor:** The code for ArticuloController.php is displayed:

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class ArticuloController extends Controller
8 {
9     //
10 }
11
```

**Figura 16:** Controlador ArticuloController generado

D:\Webs\revistapp\app\Http\Controllers\ArticuloController.php (revistapp) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

**FOLDERS**

- revistapp
- app
  - Http
    - Controllers
      - ArticuloController.php
      - Controller.php
  - Models
  - Providers- bootstrap
- config
- database
- public
- resources
- routes
  - console.php
  - web.php
- storage
- tests
- vendor
  - .editorconfig
  - .env
  - .env.example
  - .gitattributes
  - .gitignore
  - artisan
  - /\* composer.json
  - composer.lock
  - /\* package.json

◀ ▶ web index.blade.php x | show.blade.php x | ArticuloController.php x + ▼

```
1 <?php
2
3 namespace App\Http\Controllers;
4 use Illuminate\Http\Request;
5
6 class ArticuloController extends Controller
7 {
8
9     public function index()
10    {
11         $articulos = [
12             ["id" => 1, "titulo" => "Primer
13                 articulo..."],
14             ["id" => 2, "titulo" => "Segundo
15                 articulo..."],
16             ["id" => 3, "titulo" => "Tercer
17                 articulo..."],
18         ];
19         return view('articulos.index', [
20             'articulos' => $articulos
21         ]);
22     }
23
24     public function show($id)
25     {
26         return view('articulos.show', [
27             'id' => $id
28         ]);
29     }
30 }
```

**Figura 17:** Lógica de las 2 rutas del controlador

```

D:\Webs\revistapp\routes\web.php • (revistapp) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
revistapp
  app
    Http
      Controllers
        ArticuloController.php
        Controller.php
      Models
      Providers
    bootstrap
    config
    database
    public
    resources
  routes
    console.php
    web.php
  storage
  tests
  vendor
    .editorconfig
    .env
    .env.example
web.php
1 <?php
2
3 use Illuminate\Support\Facades\Route;
4
5 Route::get('/', function () {
6   return view('welcome');
7 });
8
9 use App\Http\Controllers\ArticuloController;
10
11 Route::get('articulos/', [ArticuloController::class,
12   'index'])->name('articulos.index');
13 Route::get('articulos/{id}', [ArticuloController::class,
14   'show'])->name('articulos.show');
15 Route::get('articulos/create', [ArticuloController::class,
16   'create'])->name('articulos.create');
17 Route::post('articulos/', [ArticuloController::class,
18   'store'])->name('articulos.store');
19
20
21
22
23
24
25
26
27
28

```

Figura 18: Enrutamiento del controlador

## 5.

## Configuración de la Base de Datos

Se configuró la base de datos en el archivo `.env` para conectar con MySQL de XAMPP. Luego, en phpMyAdmin, se creó la base de datos `revistapp`.

```

public
resources
routes
  console.php
  web.php
storage
tests
vendor
  .editorconfig
.env
.env.example
.gitattributes
.gitignore
artisan
/* composer.json
APP_MAINTENANCE_DRIVER=file
# APP_MAINTENANCE_STORE=database
BCRYPT_ROUNDS=12
LOG_CHANNEL=stack
LOG_STACK=single
LOG_DEPRECATED_CHANNEL=null
LOG_LEVEL=debug
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=revistapp
DB_USERNAME=root
DB_PASSWORD=

```

Figura 19: Configuración de la base de datos en el archivo `.env`

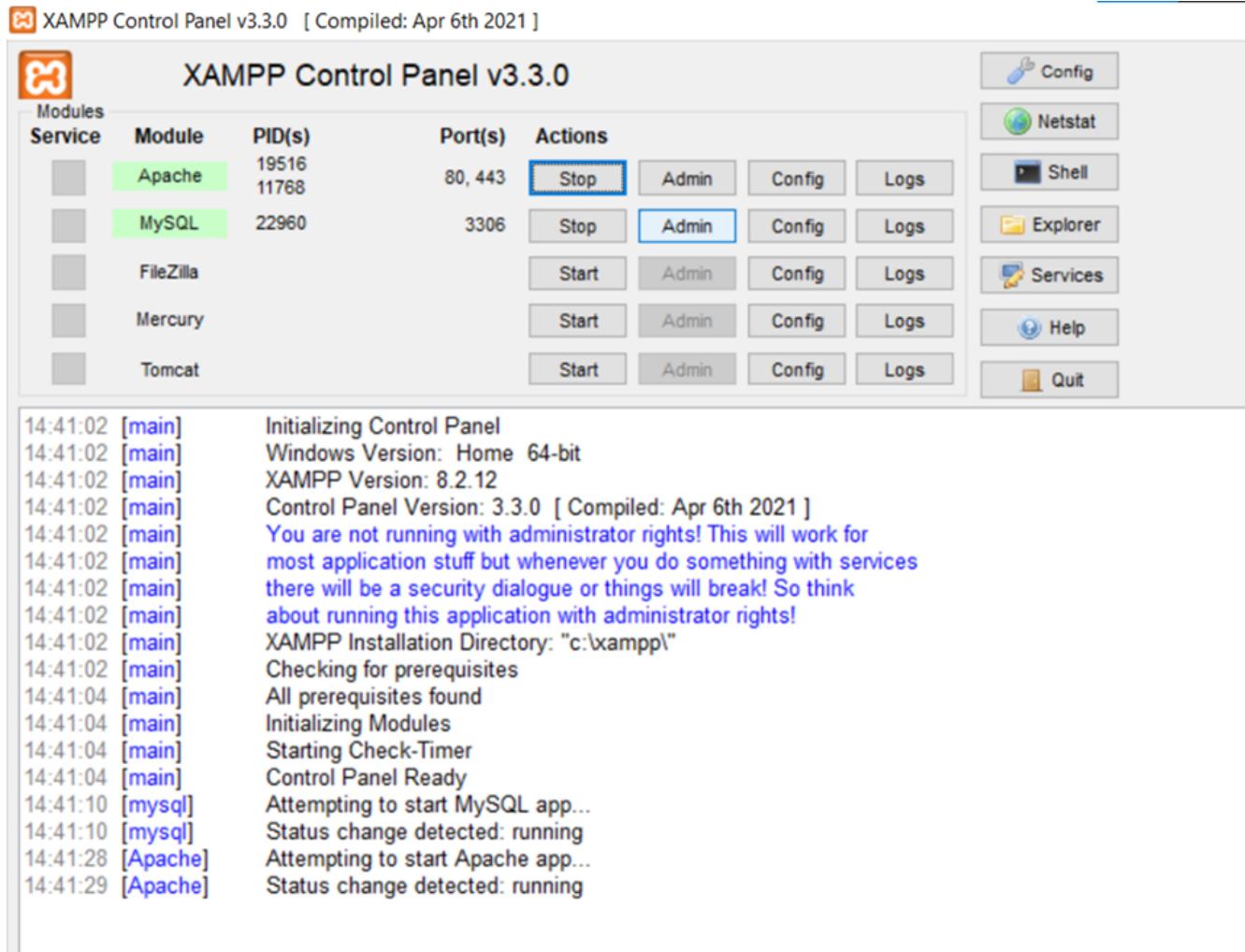


Figura 20: Inicializamos en XAMPP Apache y MySQL

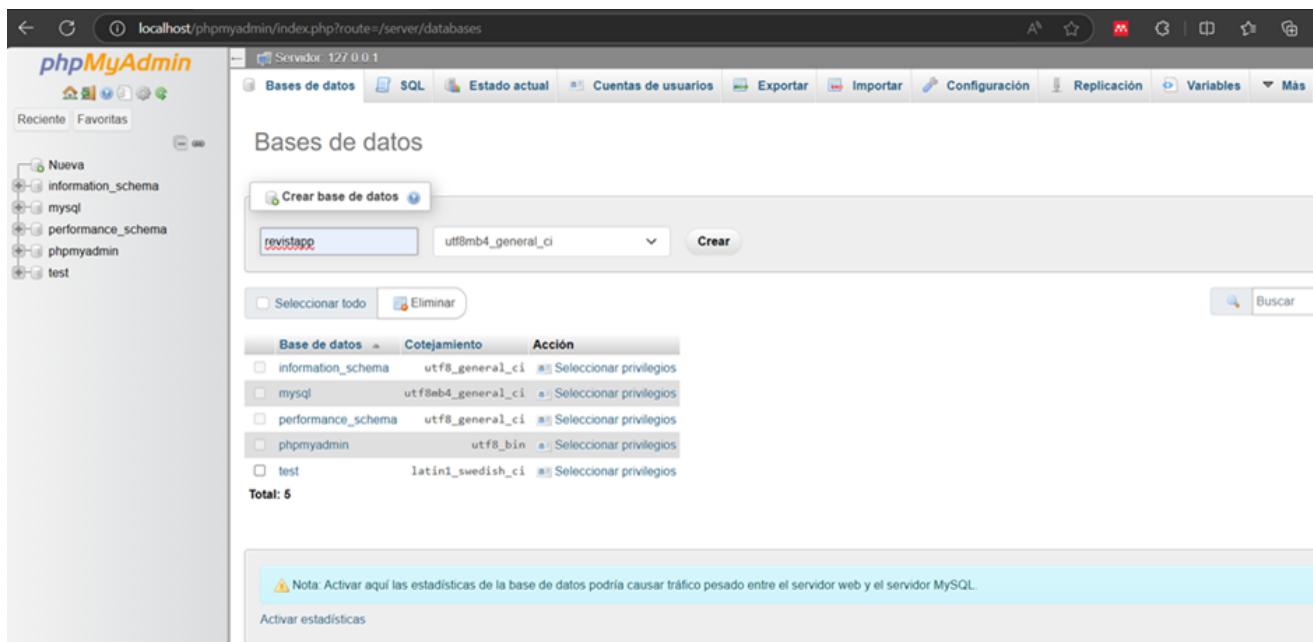
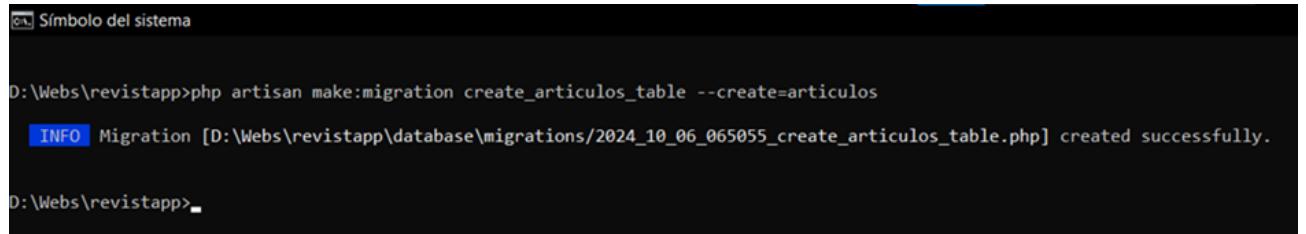


Figura 21: Creación de la base de datos revistapp

## 6. Paso 6: Crear Migraciones

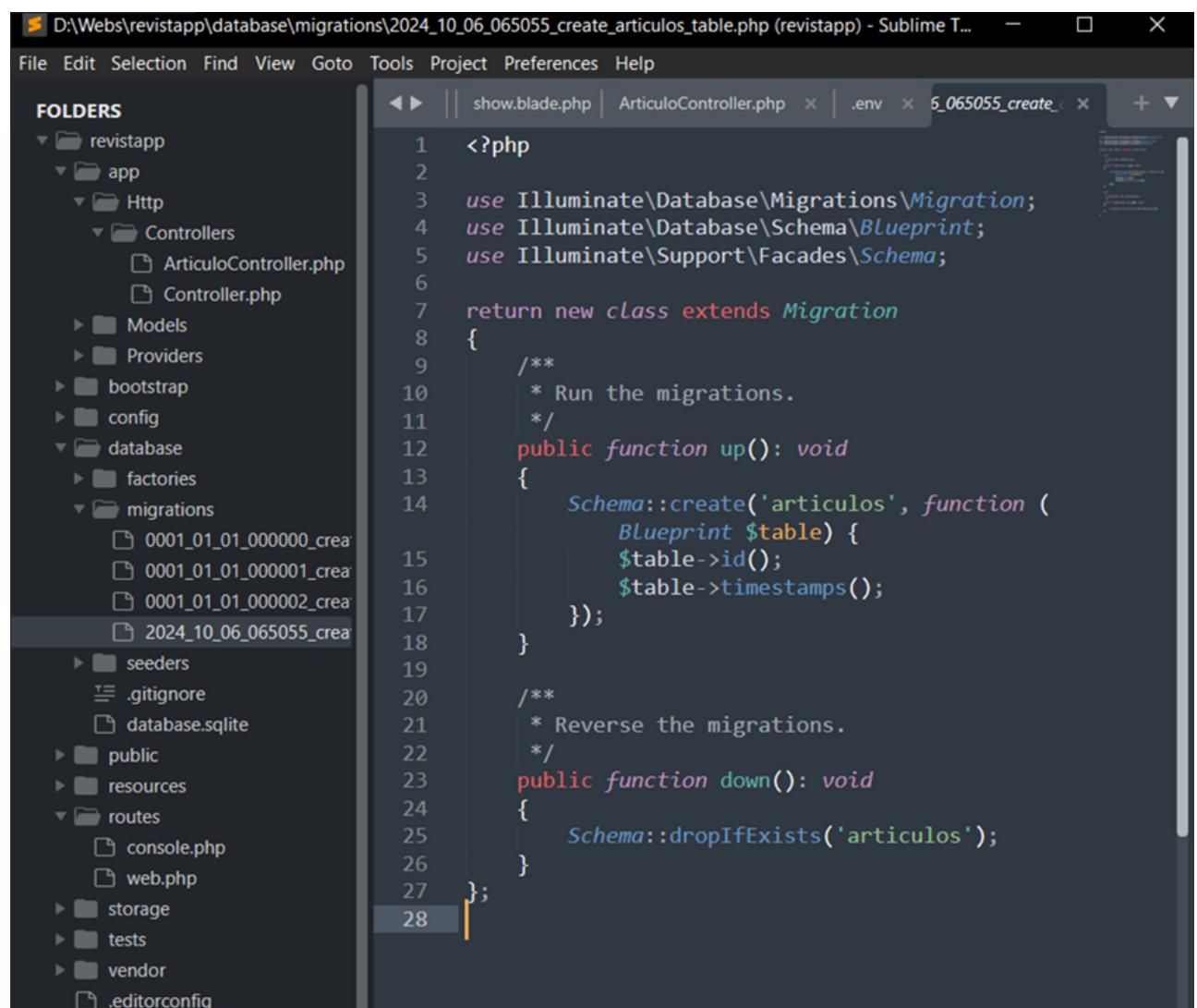
Se crearon migraciones para estructurar la base de datos. Se definió una tabla artículos con los campos *id*, *título*, y *contenido*.



```
D:\Webs\revistapp>php artisan make:migration create_articulos_table --create=articulos
INFO Migration [D:\Webs\revistapp\database\migrations\2024_10_06_065055_create_articulos_table.php] created successfully.

D:\Webs\revistapp>
```

Figura 22: Creación de la migración



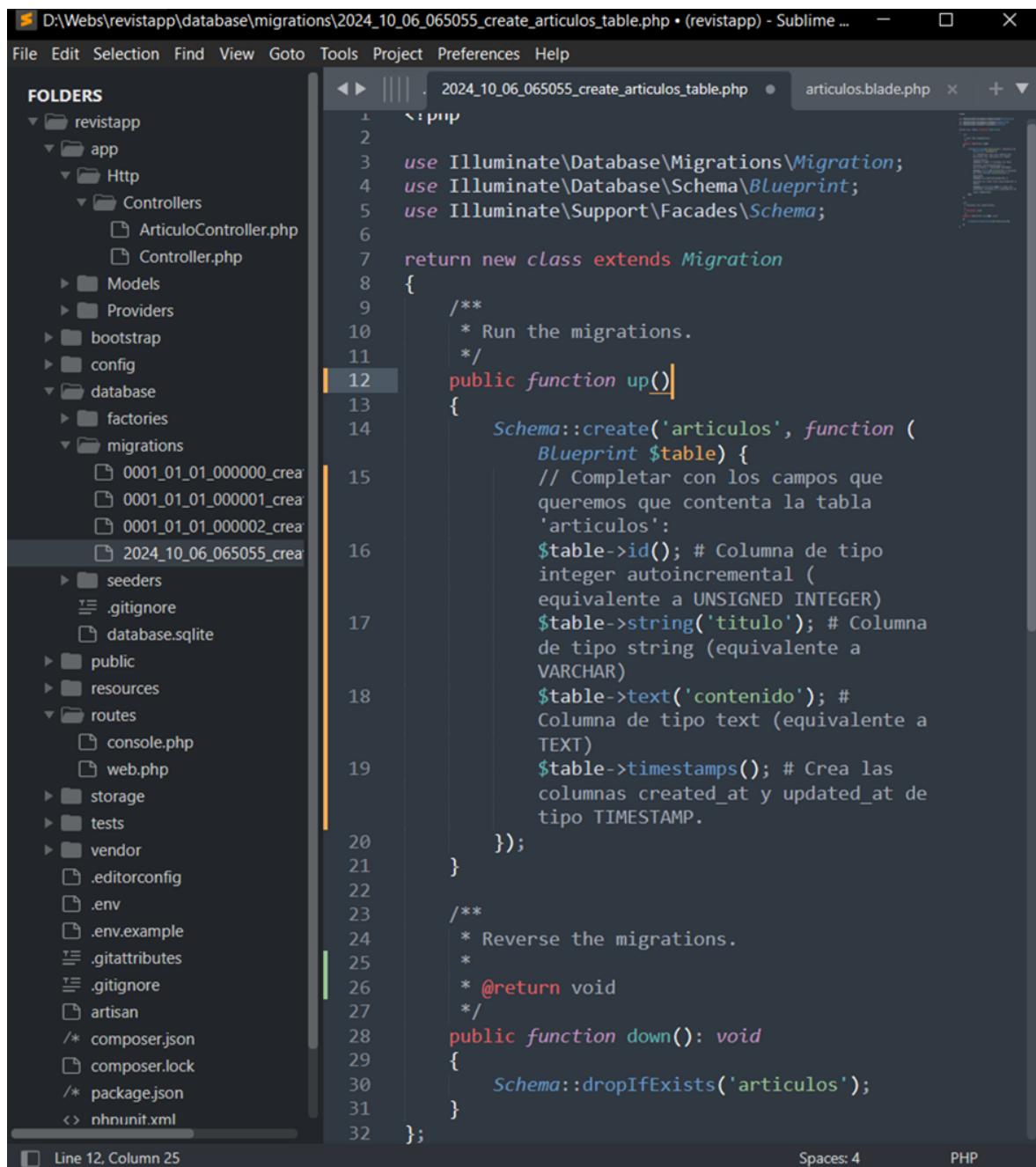
The screenshot shows a Sublime Text editor with several tabs open. The left sidebar displays the project structure of a Laravel application named 'revistapp'. The 'migrations' folder contains several migration files, including '2024\_10\_06\_065055\_create\_articulos\_table.php', which is currently selected and shown in the main editor area. The code for this migration is as follows:

```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('articulos', function (
15             Blueprint $table
16         ) {
17             $table->id();
18             $table->timestamps();
19         });
20     }
21     /**
22      * Reverse the migrations.
23      */
24     public function down(): void
25     {
26         Schema::dropIfExists('articulos');
27     }
28 };

```

Figura 23: El contenido de la clase creada por default



The screenshot shows a Sublime Text editor with two tabs open: '2024\_10\_06\_065055\_create\_articulos\_table.php' and 'articulos.blade.php'. The left panel displays a file tree for a Laravel application structure, including 'revistapp', 'app', 'Http', 'Controllers', 'Models', 'Providers', 'bootstrap', 'config', 'database', 'factories', 'migrations', 'seeders', '.gitignore', 'database.sqlite', 'public', 'resources', 'routes', 'storage', 'tests', 'vendor', and various configuration files like '.env' and '.env.example'. The right panel shows the PHP code for the migration file:

```

1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10      * Run the migrations.
11      */
12      public function up()
13  {
14          Schema::create('articulos', function (
15              Blueprint $table)
16          {
17              // Completar con los campos que
18              // queremos que contenga la tabla
19              // 'articulos':
20              $table->id(); # Columna de tipo
21              integer autoincremental (
22                  equivalente a UNSIGNED INTEGER)
23              $table->string('titulo'); # Columna
24              de tipo string (equivalente a
25              VARCHAR)
26              $table->text('contenido'); #
27              Columna de tipo text (equivalente a
28              TEXT)
29              $table->timestamps(); # Crea las
30              columnas created_at y updated_at de
31              tipo TIMESTAMP.
32      });
33  }
34  /**
35      * Reverse the migrations.
36      */
37  /**
38      * @return void
39      */
40  public function down(): void
41  {
42      Schema::dropIfExists('articulos');
43  }
44 }

```

Line 12, Column 25

Spaces: 4      PHP

Figura 24: Implementación del método *up*

```
D:\Webs\revistapp>php artisan migrate

INFO | Preparing database.

Creating migration table ..... 77.20ms DONE

INFO | Running migrations.

0001_01_000000_create_users_table ..... 159.87ms DONE
0001_01_000001_create_cache_table ..... 37.83ms DONE
0001_01_000002_create_jobs_table ..... 119.53ms DONE
2024_10_06_065055_create_articulos_table ..... 16.15ms DONE
```

Figura 25: Implementación del método *up*

Figura 26: Observación en *phpMyAdmin* la base de datos generada up

## 7. Paso 7: Crear un Modelo

Finalmente, se generó el modelo *Articulo* que interactúa directamente con la base de datos, permitiendo operaciones como la creación y consulta de artículos.

```
D:\Webs\revistapp>php artisan make:model Articulo
INFO Model [D:\Webs\revistapp\app\Models\Articulo.php] created successfully.
```

Figura 27: Creación del modelo *Articulo*

```

D:\Webs\revistapp\app\Models\Articulo.php (revistapp) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
revistapp
  app
    Http
      Controllers
        ArticuloController.php
        Controller.php
    Models
      Articulo.php
      User.php
      Providers
    bootstrap
    config
    database
    factories
    migrations

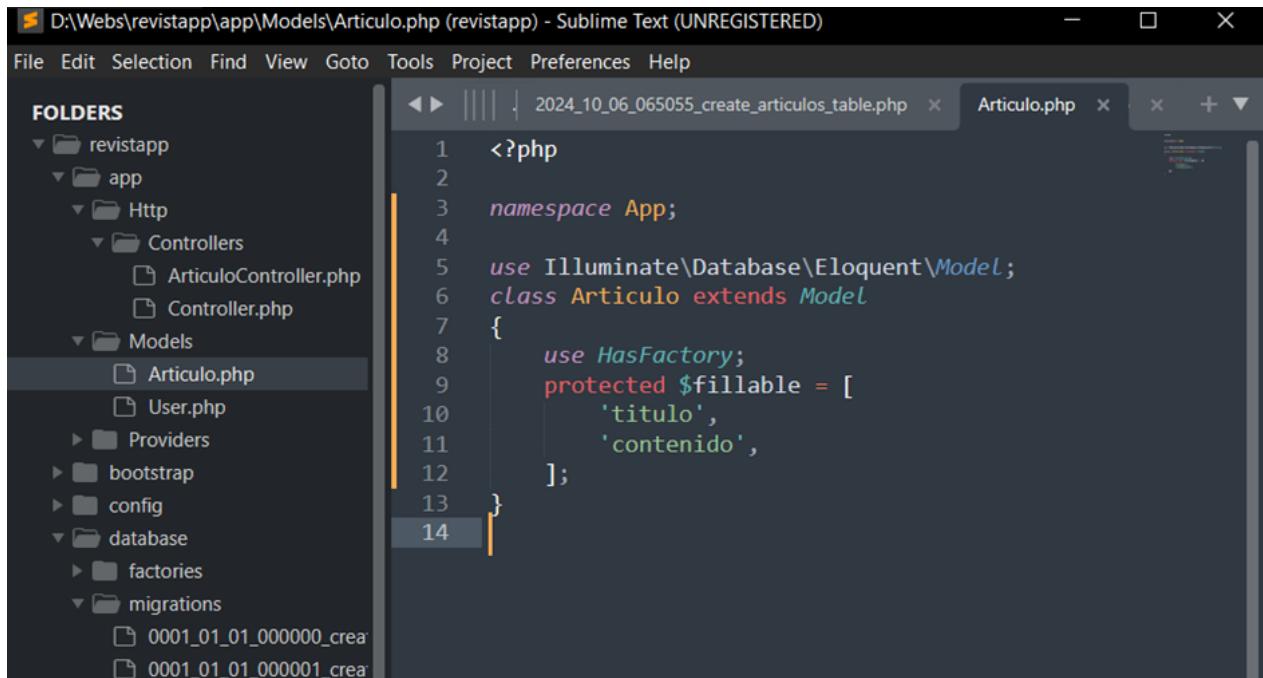
```

```

1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Articulo extends Model
9 {
10   use HasFactory;
11 }
12

```

Figura 28: Modelo *Articulo* creado por default



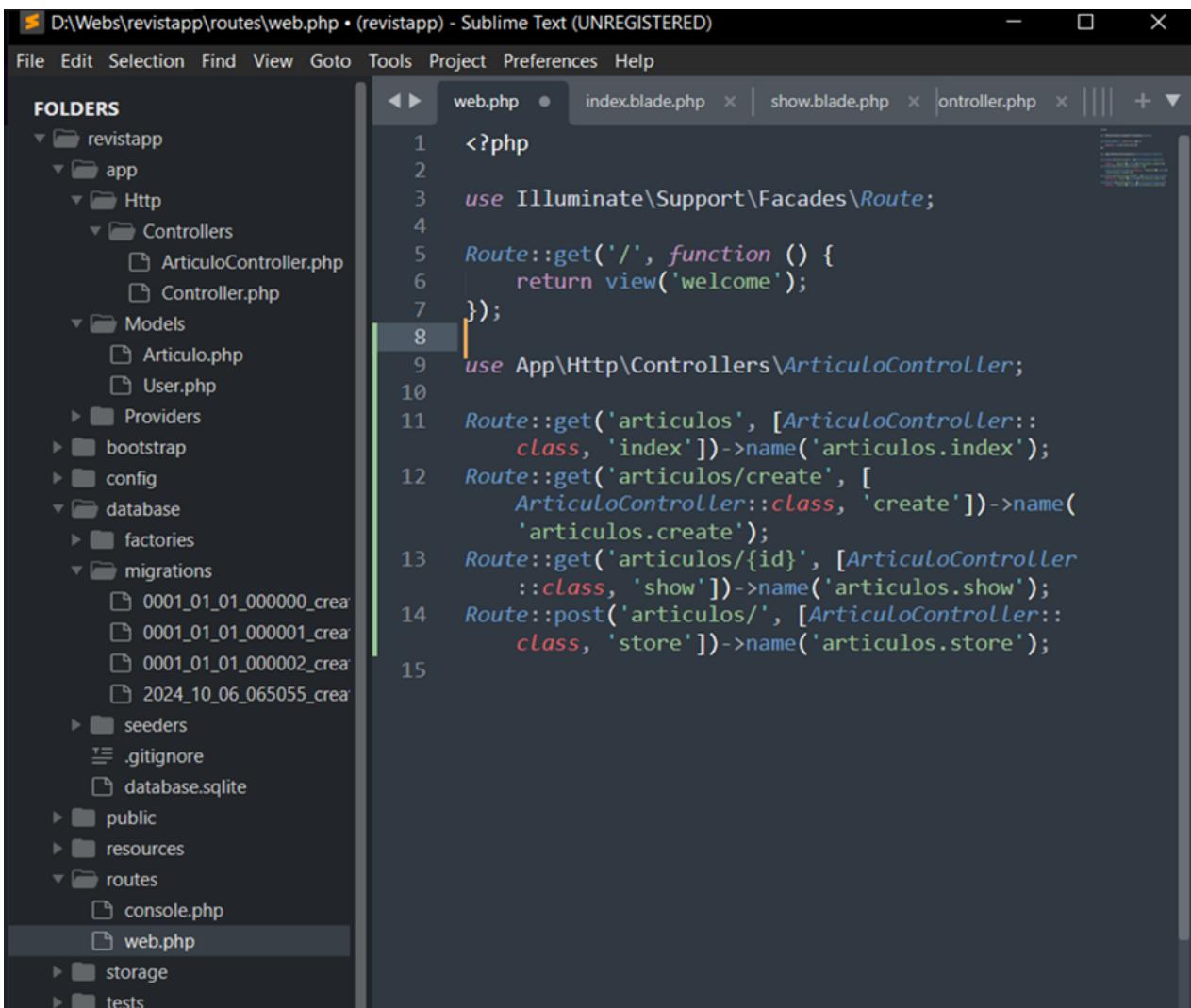
```

D:\Webs\revistapp\app\Models\Articulo.php (revistapp) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
revistapp
  app
    Http
      Controllers
        ArticuloController.php
        Controller.php
    Models
      Articulo.php
      User.php
      Providers
    bootstrap
    config
    database
    factories
    migrations
      0001_01_01_000000_crea
      0001_01_01_000001_crea
      2024_10_06_065055_create_articulos_table.php
      Articulo.php
      index.blade.php
      show.blade.php
      controller.php
      web.php
      console.php
      storage
      tests
      .gitignore
      database.sqlite
  public
  resources
  routes
    web.php
  storage
  tests
  
```

```

1 <?php
2
3 namespace App;
4
5 use Illuminate\Database\Eloquent\Model;
6 class Articulo extends Model
7 {
8     use HasFactory;
9     protected $fillable =
10         [
11             'titulo',
12             'contenido',
13         ];
14 }

```

Figura 29: Inclusión de los campos en una propiedad llamada *fillable*


```

D:\Webs\revistapp\routes\web.php • (revistapp) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
revistapp
  app
    Http
      Controllers
        ArticuloController.php
        Controller.php
    Models
      Articulo.php
      User.php
      Providers
    bootstrap
    config
    database
    factories
    migrations
      0001_01_01_000000_crea
      0001_01_01_000001_crea
      0001_01_01_000002_crea
      2024_10_06_065055_create_articulos_table.php
      Articulo.php
      index.blade.php
      show.blade.php
      controller.php
      web.php
      console.php
      storage
      tests
      .gitignore
      database.sqlite
  public
  resources
  routes
    web.php
  storage
  tests
  
```

```

1 <?php
2
3 use Illuminate\Support\Facades\Route;
4
5 Route::get('/', function () {
6     return view('welcome');
7 });
8 use App\Http\Controllers\ArticuloController;
9
10 Route::get('articulos', [ArticuloController::class, 'index'])->name('articulos.index');
11 Route::get('articulos/create', [ArticuloController::class, 'create'])->name('articulos.create');
12 Route::get('articulos/{id}', [ArticuloController::class, 'show'])->name('articulos.show');
13 Route::post('articulos/', [ArticuloController::class, 'store'])->name('articulos.store');
14
15

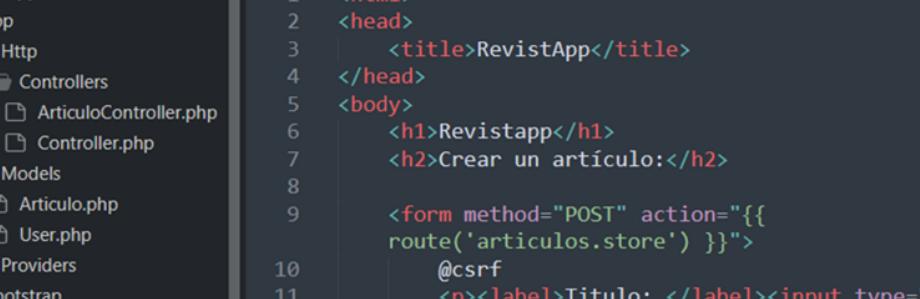
```

Figura 30: Ruta empleada de tipo GET y de tipo POST

The screenshot shows a code editor with two panes. The left pane displays a file tree for a Laravel application:

- FOLDERS
  - revitapp
  - app
    - Http
      - Controllers
        - ArticuloController.php
        - Controller.php
    - Models
      - Articulo.php
      - User.php
    - Providers
    - bootstrap
    - config
    - database
      - factories
      - migrations
        - 0001\_01\_01\_000000\_create\_articulos\_table.php
        - 0001\_01\_01\_000001\_create\_users\_table.php
        - 0001\_01\_01\_000002\_create\_sessions\_table.php
        - 2024\_10\_06\_065055\_create\_articulos\_table.php
    - seeders
      - .gitignore
      - database.sqlite
    - public
    - resources
    - routes
      - console.php
      - web.php
    - storage
    - tests
    - vendor
      - .editorconfig
      - .env
      - .env.example
      - .gitattributes
      - .gitignore

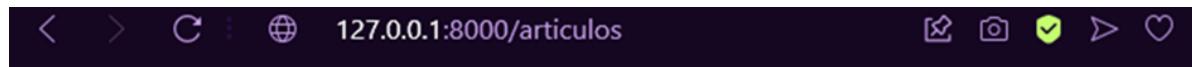
**Figura 31:** Implementación de los métodos del controlador



The screenshot shows a code editor with two main sections. On the left, a tree view displays the project structure:

- FOLDERS
  - revistapp
  - app
    - Http
      - Controllers
        - ArticuloController.php
        - Controller.php
    - Models
      - Articulo.php
      - User.php
    - Providers
    - bootstrap
    - config
    - database
      - factories
      - migrations
        - 0001\_01\_01\_000000\_crea
        - 0001\_01\_01\_000001\_crea
        - 0001\_01\_01\_000002\_crea
        - 2024\_10\_06\_065055\_crea
    - seeders

**Figura 32:** Creación de la vista “create” para mostrar el formulario



# Revistapp

## Listado artículos:

[Crear nuevo](#)

[Enlace Título](#)

Figura 33: Vista en nuestro local host /articulos



# Revistapp

## Crear un artículo:

Titulo:

Contenido:

[Crear](#)

[Volver](#)

Figura 34: Vista y prueba en nuestro local host /articulos/create



# Revistapp

## Listado artículos:

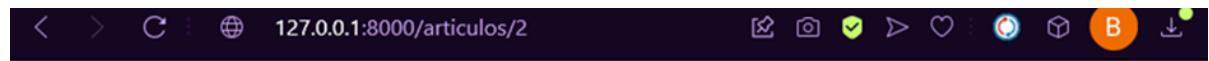
[Crear nuevo](#)

[Enlace](#)

[Título](#)

[Ver](#) Ingeniería de Software en las Telecomunicaciones

Figura 35: Vista de la prueba en nuestro local host /articulos



# Revistapp

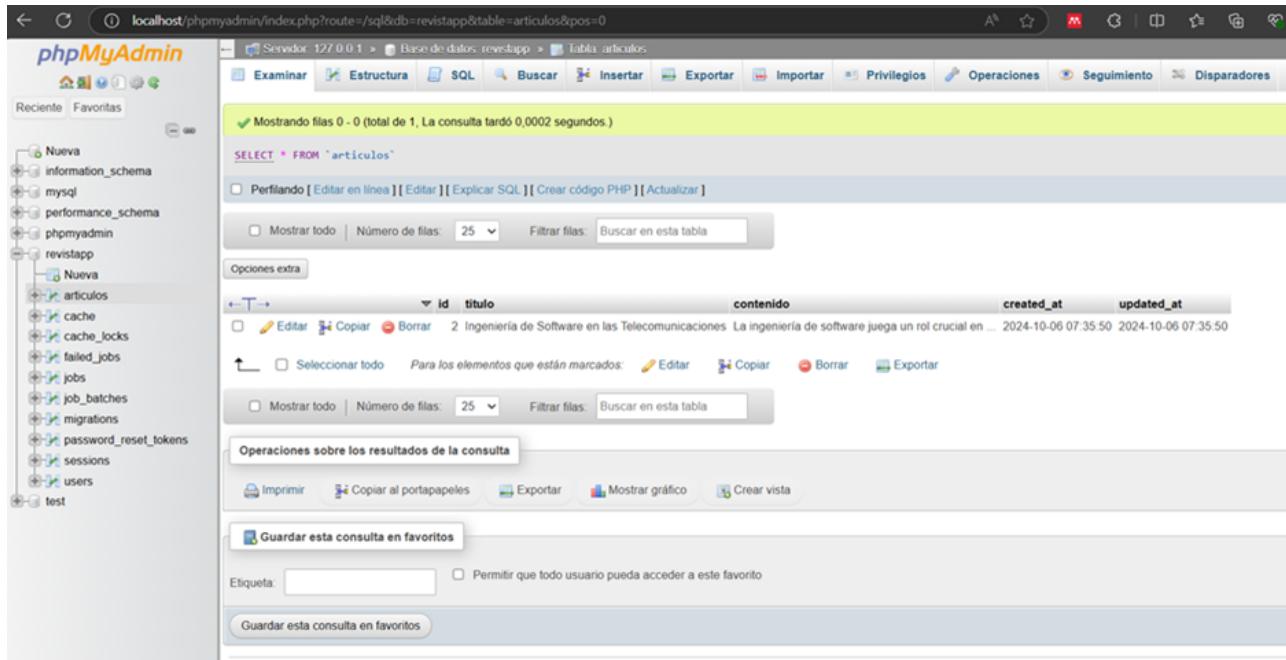
## Detalle del artículo:

Titulo: Ingeniería de Software en las Telecomunicaciones

Contenido: La ingeniería de software juega un rol crucial en el desarrollo de sistemas de telecomunicaciones eficientes y escalables. Permite gestionar la complejidad de las redes y asegurar una comunicación fiable mediante el uso de software robusto.

[Volver](#)

Figura 36: Vista de la prueba en nuestro local host /articulos/2



**Figura 37:** Observación de los datos creados en nuestra base de datos en phpMyAdmin

## VI.

## OBSERVACIONES Y CONCLUSIONES

---

### A.

### Observaciones

- Inicialmente, la conexión a MySQL fallaba debido a configuraciones incorrectas en el archivo .env. Se corrigió ajustando las credenciales.
- Los permisos de escritura en los directorios storage y bootstrap/cache generaron advertencias durante la instalación. Se resolvió otorgando permisos adecuados usando comandos terminal.

### B.

### Conclusiones

---

- Laravel demostró ser una herramienta potente para el desarrollo web, facilitando la gestión de rutas, controladores y vistas mediante su estructura MVC. Esta organización clara y modular nos permitió realizar cambios en el proyecto de manera ágil y precisa, sin afectar otras partes del código.
- La capacidad de Composer para manejar las dependencias del proyecto permitió un desarrollo más ágil y ordenado, simplificando la integración de nuevas funcionalidades y asegurando la compatibilidad de versiones entre los paquetes utilizados.
- Las migraciones de Laravel facilitaron la creación y actualización de la base de datos, garantizando coherencia en el modelo de datos a lo largo del proyecto. Nos permitió gestionar cambios en la base de datos de manera controlada, evitando problemas de incompatibilidad o pérdida de información.
- Una de las grandes ventajas de Laravel es que su estructura es muy escalable. La posibilidad de agregar funcionalidades nuevas sin comprometer el funcionamiento del sistema base es un factor clave para futuros proyectos. Esto también facilita el mantenimiento a largo plazo, ya que las secciones del código están claramente separadas.
- A lo largo del proyecto, nos encontramos con varios problemas que nos ayudaron a aprender más sobre el framework y las herramientas utilizadas. Cada obstáculo nos llevó a mejorar nuestra comprensión sobre las mejores prácticas en desarrollo web, así como la optimización del código y la gestión eficiente del servidor.

## VII. REFERENCIAS

---

- [1] “Primeros pasos - Aprende Laravel 9 paso a paso”. Aprende Laravel 9 paso a paso. Accedido el 6 de octubre de 2024. [En línea]. Disponible: <https://laravel9.netlify.app/03-primeros-pasos/>.
- [2] Overleaf, “Overleaf: The Online LaTeX Editor”, 2024. [Online]. Available: <https://www.overleaf.com/>. [Accessed: Oct-2024].
- [3] Report created in Latex by Rony Ticona.