

Laravel

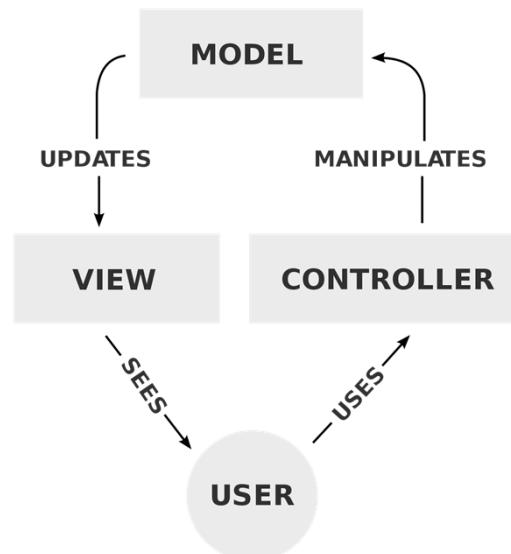
Framework PHP

Laravel

- Es un marco de desarrollo utilizado para crear sitios y aplicaciones web. El creador de Laravel, Taylor Otwell, lanzó su versión Beta el 9 de junio de 2011.
- Laravel 10 = publicado en febrero de 2023.
- El lanzamiento de Laravel 11 está previsto para febrero de 2024.

Laravel

- En la actualidad, Laravel está alojado en GitHub, donde cuenta con 65.900 usuarios y ha sido descargado 144 millones de veces. Tiene licencia MIT.
- Laravel utiliza PHP con la arquitectura MVC.

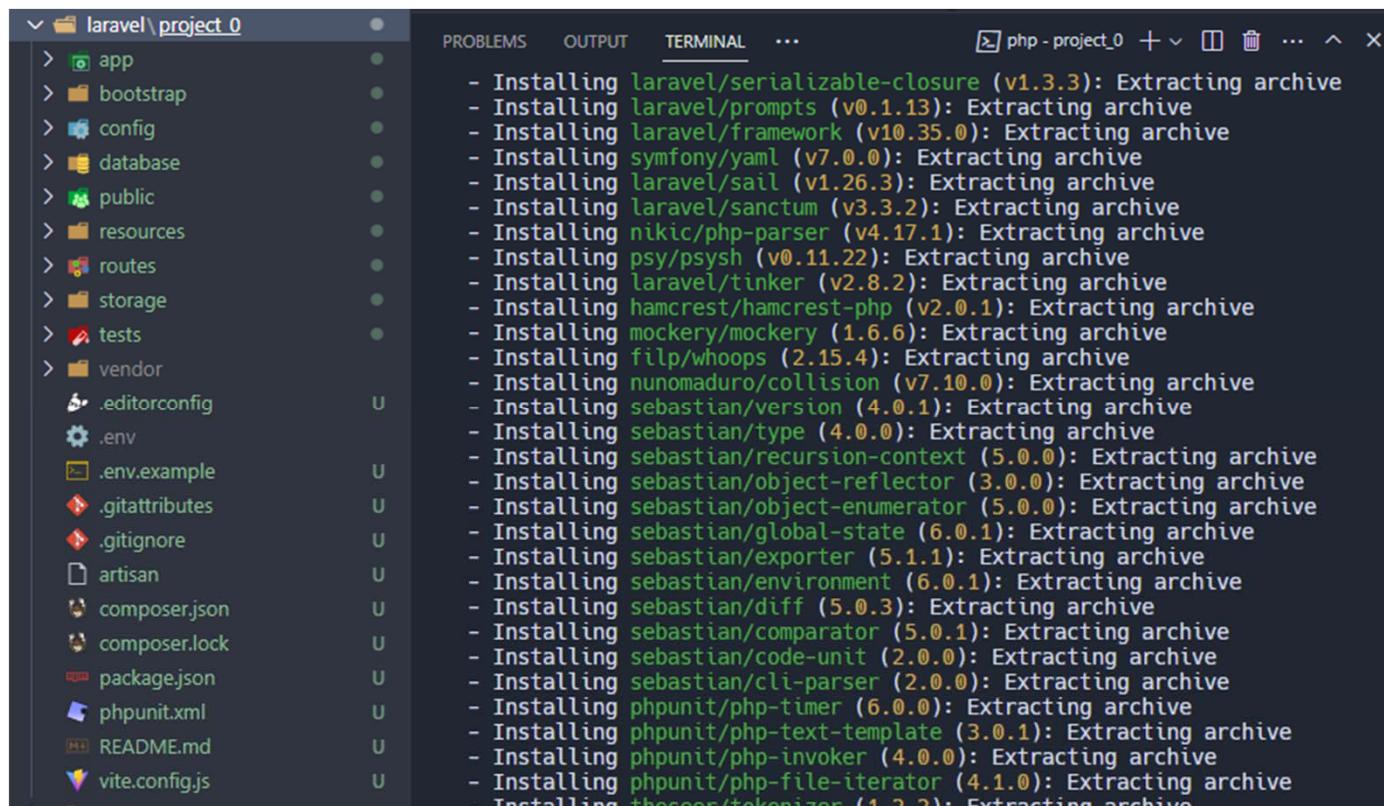


Características de Laravel

- Sistema completo de autenticación
- Mapeo objeto-relacional (ORM)
- Biblioteca de módulos
- Interfaz de línea de comandos (CLI) con comandos preconstruidos
- Pruebas automáticas
- Entorno de desarrollo virtual

Tu primer proyecto Laravel

composer create-project laravel/laravel .

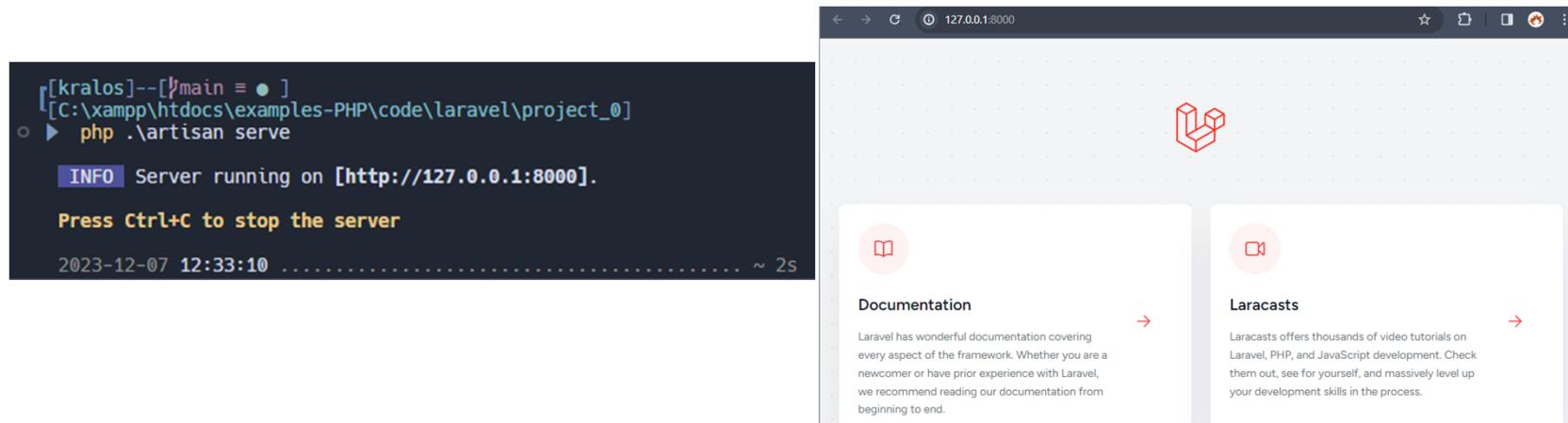


The screenshot shows a terminal window with the title "php - project_0". The left side displays the file structure of a newly created Laravel project named "project_0". The right side shows the output of the Composer command, which lists the installation of various dependencies:

```
- Installing laravel/serializable-closure (v1.3.3): Extracting archive
- Installing laravel/prompts (v0.1.13): Extracting archive
- Installing laravel/framework (v10.35.0): Extracting archive
- Installing symfony/yaml (v7.0.0): Extracting archive
- Installing laravel/sail (v1.26.3): Extracting archive
- Installing laravel/sanctum (v3.3.2): Extracting archive
- Installing nikic/php-parser (v4.17.1): Extracting archive
- Installing psy/psysh (v0.11.22): Extracting archive
- Installing laravel/tinker (v2.8.2): Extracting archive
- Installing hamcrest/hamcrest-php (v2.0.1): Extracting archive
- Installing mockery/mockery (1.6.6): Extracting archive
- Installing filp/whoops (2.15.4): Extracting archive
- Installing nunomaduro/collision (v7.10.0): Extracting archive
- Installing sebastian/version (4.0.1): Extracting archive
- Installing sebastian/type (4.0.0): Extracting archive
- Installing sebastian/recursion-context (5.0.0): Extracting archive
- Installing sebastian/object-reflector (3.0.0): Extracting archive
- Installing sebastian/object-enumerator (5.0.0): Extracting archive
- Installing sebastian/global-state (6.0.1): Extracting archive
- Installing sebastian/exporter (5.1.1): Extracting archive
- Installing sebastian/environment (6.0.1): Extracting archive
- Installing sebastian/diff (5.0.3): Extracting archive
- Installing sebastian/comparator (5.0.1): Extracting archive
- Installing sebastian/code-unit (2.0.0): Extracting archive
- Installing sebastian/cli-parser (2.0.0): Extracting archive
- Installing phpuunit/php-timer (6.0.0): Extracting archive
- Installing phpuunit/php-text-template (3.0.1): Extracting archive
- Installing phpuunit/php-invoker (4.0.0): Extracting archive
- Installing phpuunit/php-file-iterator (4.1.0): Extracting archive
- Installing thescraper/tokenizer (1.2.2): Extracting archive
```

Tu primer proyecto Laravel

- Una vez creado el proyecto, inicia el servidor de desarrollo local de Laravel mediante el comando serve de la CLI Artisan de Laravel.



Carpetas más importantes

- app: Contiene la lógica principal de la aplicación, como los controladores, las excepciones, las solicitudes, etc.
- bootstrap: Contiene los archivos de inicialización de la aplicación, incluyendo la carga de archivos de entorno y la configuración de la configuración.
- config: Contiene la configuración de la aplicación, incluyendo la configuración de la base de datos, el correo electrónico y la aplicación.
- database: Contiene las migraciones de la base de datos, las semillas y las fábricas.
- public: Contiene el archivo index.php que es el punto de entrada de la aplicación, así como cualquier archivo público, como CSS, JavaScript e imágenes.

Carpetas más importantes

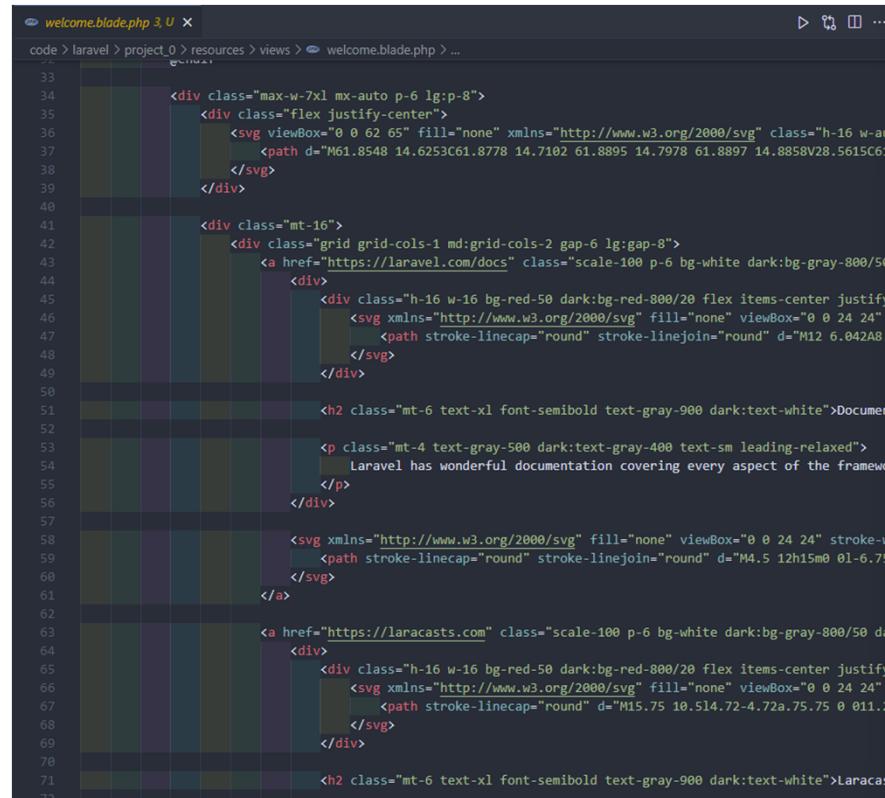
- resources: Contiene los recursos de la aplicación, incluyendo las vistas, los archivos CSS, los archivos JavaScript y los archivos de idioma.
- routes: Contiene las rutas de la aplicación, incluyendo las rutas web y API.
- storage: Contiene los archivos de almacenamiento de la aplicación, como las imágenes cargadas y los registros.
- tests: Contiene las pruebas de la aplicación.
- vendor: Contiene las dependencias de la aplicación instaladas por Composer.
- .env: Archivo de entorno de la aplicación, que contiene la configuración específica de la máquina, como la información de la base de datos y la configuración de correo electrónico.

Carpetas más importantes

- `.env.example`: Ejemplo del archivo de entorno de la aplicación.
- `composer.json`: Archivo de configuración de Composer, que especifica las dependencias de la aplicación.
- `composer.lock`: Archivo generado por Composer que especifica las versiones exactas de las dependencias instaladas.
- `package.json`: Archivo de configuración de npm, que especifica las dependencias de JavaScript de la aplicación.
- `README.md`: Archivo de documentación de la aplicación.

Primeros pasos {Código inicial}

examples-PHP\code\laravel\project_0\resources\views\welcome.blade.php



The screenshot shows a code editor window with the file "welcome.blade.php" open. The code is written in Blade templating language, which is a superset of PHP. It includes HTML-like syntax with Blade directives like @if and @foreach, as well as standard PHP code. The code is styled with dark-themed syntax highlighting, where tags and attributes are in green, strings in blue, and comments in grey. The editor interface includes a top bar with file navigation and a bottom status bar.

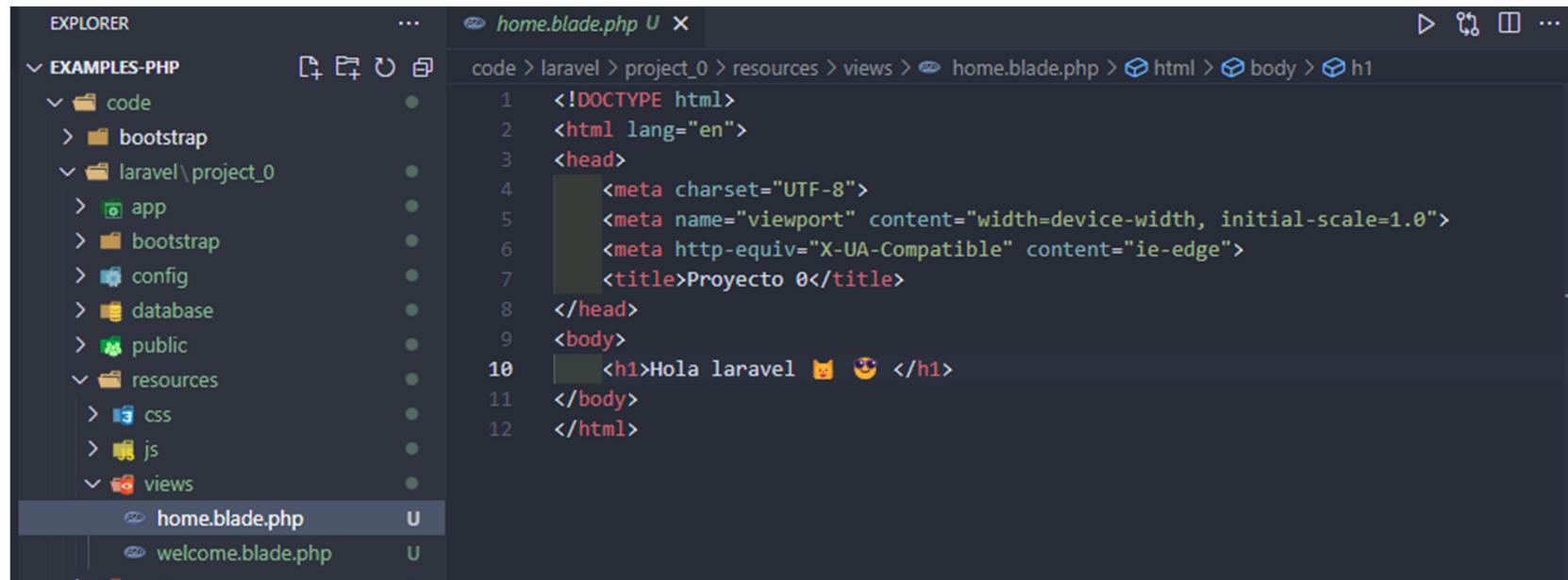
```
<div class="max-w-7xl mx-auto p-6 lg:p-8">
    <div class="flex justify-center">
        <svg viewBox="0 0 65 65" fill="none" xmlns="http://www.w3.org/2000/svg" class="h-16 w-aut
        <path d="M61.8548 14.6253C61.8778 14.7102 61.8895 14.7978 61.8897 14.8858V28.5615C61.
    </div>

    <div class="mt-16">
        <div class="grid grid-cols-1 md:grid-cols-2 gap-6 lg:gap-8">
            <a href="https://laravel.com/docs" class="scale-100 p-6 bg-white dark:bg-gray-800/50
                <div class="h-16 w-16 bg-red-50 dark:bg-red-800/20 flex items-center justify-c
                    <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" s
                        <path stroke-linecap="round" stroke-linejoin="round" d="M12 6.042A8.9
                    </svg>
                </div>
            <div class="mt-4 text-gray-500 dark:text-gray-400 text-sm leading-relaxed">
                Laravel has wonderful documentation covering every aspect of the framework.
            </div>
        </div>
        <div class="mt-16 w-16 bg-red-50 dark:bg-red-800/20 flex items-center justify-c
            <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-wi
                <path stroke-linecap="round" stroke-linejoin="round" d="M4.5 12h15m0 0l-6.75-
            </svg>
        </div>
    </div>

    <a href="https://laracasts.com" class="scale-100 p-6 bg-white dark:bg-gray-800/50 dar
        <div class="h-16 w-16 bg-red-50 dark:bg-red-800/20 flex items-center justify-c
            <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" s
                <path stroke-linecap="round" d="M15.75 10.51472-4.72a.75 .75 0 011.28
            </svg>
        </div>
    </div>
</div>
```

Primeros pasos {Código inicial}

examples-PHP\code\laravel\project_0\resources\views\home.blade.php



The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar: Shows the project structure:
 - EXAMPLES-PHP
 - code
 - bootstrap
 - laravel\project_0
 - app
 - bootstrap
 - config
 - database
 - public
 - resources
 - css
 - js
 - views
 - home.blade.php
 - welcome.blade.php
- home.blade.php** tab: Active file.
- Code Editor Area:**

```
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>Proyecto 0</title>
8  </head>
9  <body>
10     <h1>Hola laravel 🐱 😊 </h1>
11  </body>
12  </html>
```

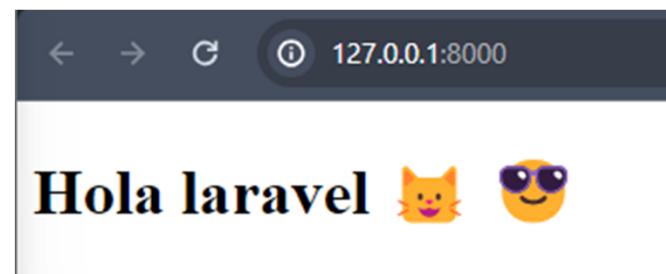
Primeros pasos {Código inicial}

The image shows two side-by-side code editors, likely from a IDE like VS Code, displaying the same file: `web.php`. Both editors show the same code content, which is the initial configuration for a Laravel application's web routes.

```
<?php  
use Illuminate\Support\Facades\Route;  
  
Route::get('/', function () {  
    return view('welcome');  
});
```

The left editor's sidebar shows the project structure under `EXAMPLES-PHP`, including `code`, `bootstrap`, `laravel\project_0` (which contains `app`, `bootstrap`, `config`, `database`, `public`, `resources` with `css` and `js`, and `views` containing `home.blade.php` and `welcome.blade.php`), and `routes` (containing `api.php`, `channels.php`, `console.php`, and `web.php`).

The right editor's sidebar shows a similar project structure, with the `routes` folder containing `api.php`, `channels.php`, `console.php`, and `web.php`.



{¿Qué es artisan?}

- La interfaz de línea de comandos de Laravel.
- Artisan está basado en el componente Console de Symfony y nos ofrece un conjunto de comandos que nos pueden ayudar a realizar diferentes tareas durante el desarrollo e incluso cuando la aplicación se encuentra en producción.
- Para conocer el listado completo de los comandos disponibles ejecutamos en consola, en el directorio raíz de un proyecto de Laravel: ***php artisan list***

{¿Qué es artisan?}

```
[kralos]--[● main = ● ]
[C:\xampp\htdocs\examples-PHP\code\laravel\project_0]
▶ php artisan list
Laravel Framework 10.35.0

Usage:
  command [options] [arguments]

Options:
  -h, --help           Display help for the given command. When no command is given display help for the list command
  and
  -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
  --env[=ENV]           The environment the command should run under
  -v|vv|vvv, --verbose Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug
```

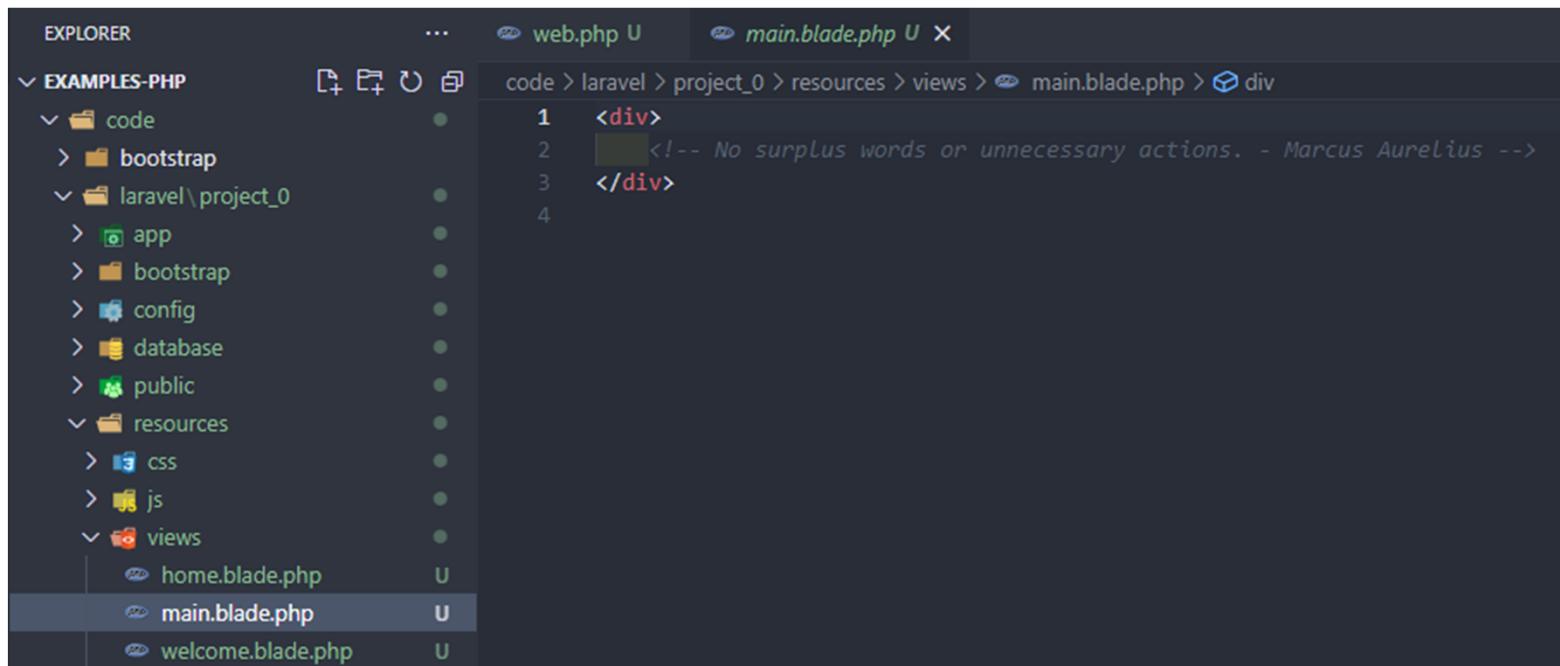
```
[kralos]--[● main = ● ]
[C:\xampp\htdocs\examples-PHP\code\laravel\project_0]
▶ php artisan route:list

GET|HEAD   /
POST      _ignition/execute-solution ignition.executeSolution > Spatie\LaravelIgnition > ExecuteSolutionController
GET|HEAD   _ignition/health-check ..... ignition.healthCheck > Spatie\LaravelIgnition > HealthCheckController
POST      _ignition/update-config .... ignition.updateConfig > Spatie\LaravelIgnition > UpdateConfigController
GET|HEAD   api/user .....
GET|HEAD   sanctum/csrf-cookie ....... sanctum.csrf-cookie > Laravel\Sanctum > CsrfCookieController@show

Showing [6] routes
```

{¿Qué es artisan?}

```
[kralos]--[~] main
[C:\xampp\htdocs\examples-PHP\code\laravel\project_0]
▶ php artisan make:view main
INFO View [C:\xampp\htdocs\examples-PHP\code\laravel\project_0\resources\views\main.blade.php] created successfully.
```



Configuración del proyecto

- Todos los archivos de configuración para el framework Laravel se almacenan en el directorio config. Cada opción está documentada.

```
code > laravel > project_0 > config > app.php
1  <?php
2
3  use Illuminate\Support\Facades\Facade;
4  use Illuminate\Support\ServiceProvider;
5
6  return [
7
8      /*
9      |--------------------------------------------------------------------------
10     | Application Name
11     |--------------------------------------------------------------------------
12     |
13     | This value is the name of your application. This value is used when the
14     | framework needs to place the application's name in a notification or
15     | any other location as required by the application or its packages.
16     |
17     */
18
19     'name' => env('APP_NAME', 'Laravel'),
```

Configuración del proyecto

php artisan about

```
[C:\xampp\htdocs\examples-PHP\code\laravel\project_0]
• ▶ php artisan about

Environment . . . . .
Application Name . . . . . Laravel
Laravel Version . . . . . 10.35.0
PHP Version . . . . . 8.2.12
Composer Version . . . . . 2.6.5
Environment . . . . . local
Debug Mode . . . . . ENABLED
URL . . . . . localhost
Maintenance Mode . . . . OFF

Cache . . . . .
Config . . . . . NOT CACHED
Events . . . . . NOT CACHED
Routes . . . . . NOT CACHED
Views . . . . . CACHED

Drivers . . . . .
Broadcasting . . . . . log
Cache . . . . . file
Database . . . . . mysql
Logs . . . . . stack / single
Mail . . . . . smtp
Queue . . . . . sync
Session . . . . . file
```

Configuración del entorno

- Laravel utiliza la librería PHP DotEnv. En una nueva instalación de Laravel, el directorio raíz de su aplicación contendrá un archivo `.env.example` que define muchas variables de entorno comunes.
- Durante el proceso de instalación de Laravel, este archivo se copiará automáticamente a `.env`.

Configuración del entorno

- El archivo .env por defecto de Laravel contiene algunos valores de configuración comunes que pueden diferir en función de si la aplicación se ejecuta localmente o en un servidor web de producción. Estos valores se recuperan de varios archivos de configuración de Laravel dentro del directorio config utilizando la función env de Laravel.

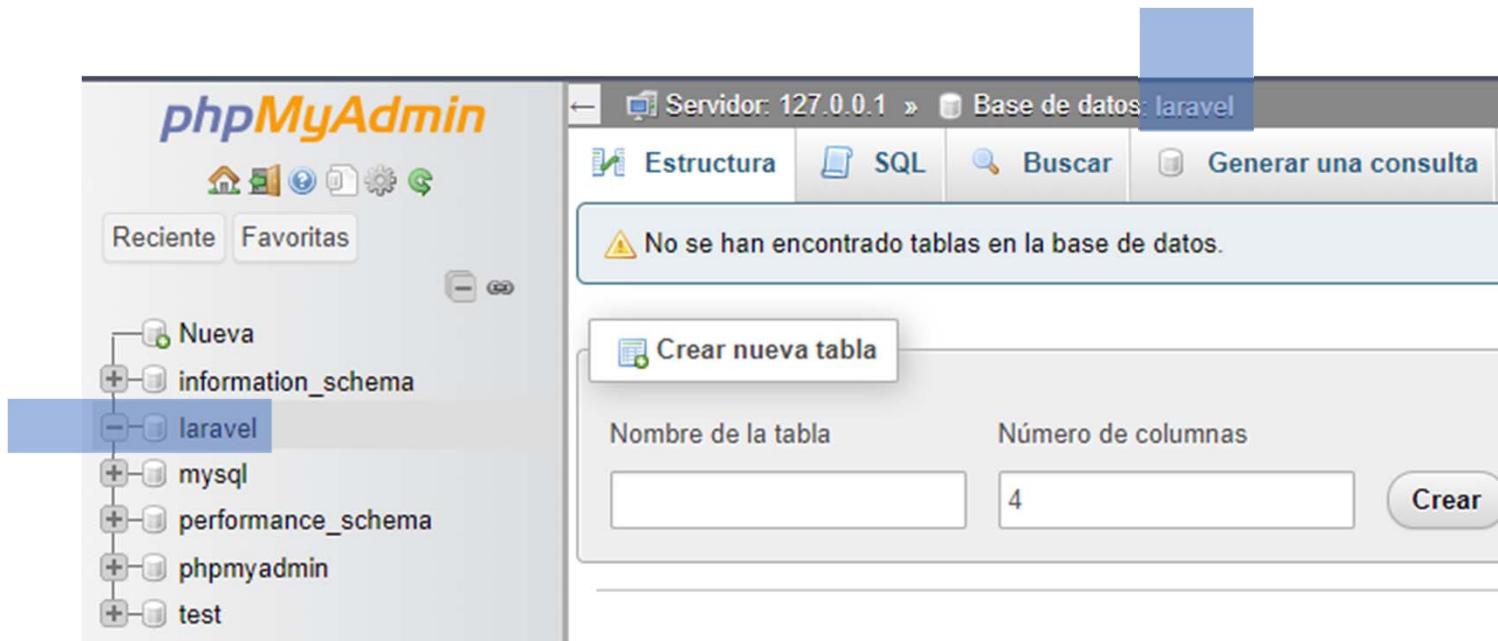
Configuración del entorno

- Su archivo .env no debe ser enviado al control de código fuente de la aplicación, ya que cada desarrollador / servidor que utilice la aplicación podría requerir una configuración de entorno diferente.
- Además, esto supondría un riesgo de seguridad en el caso de que un intruso accediera a su repositorio de control de código fuente, ya que cualquier credencial sensible quedaría expuesta.

Configuración del entorno

```
code > laravel > project_0 > ⚙ .env
 1 APP_NAME=Laravel
 2 APP_ENV=local
 3 APP_KEY=base64:R+8sdo8VlOUWEB01Nnd6+2PxM6jMWiTdkMSfAdVqO0E=
 4 APP_DEBUG=true
 5 APP_URL=http://localhost
 6
 7 LOG_CHANNEL=stack
 8 LOG_DEPRECATED_CHANNEL=null
 9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=laravel
15 DB_USERNAME=root
16 DB_PASSWORD=
```

Creando la BD

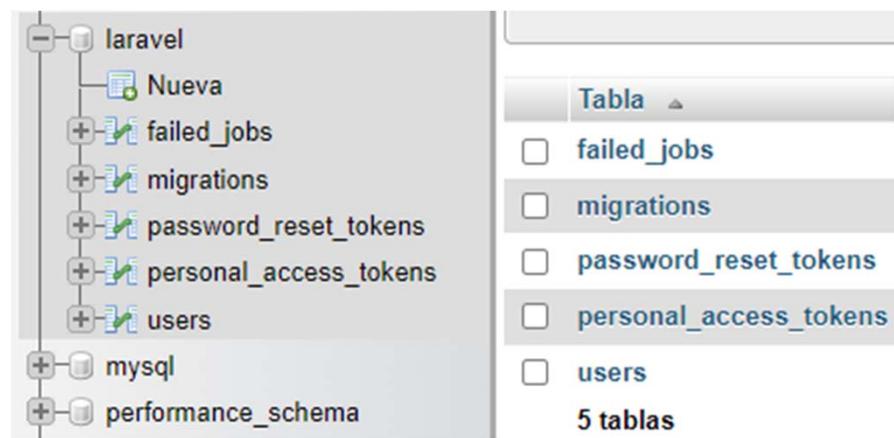


Creando la BD

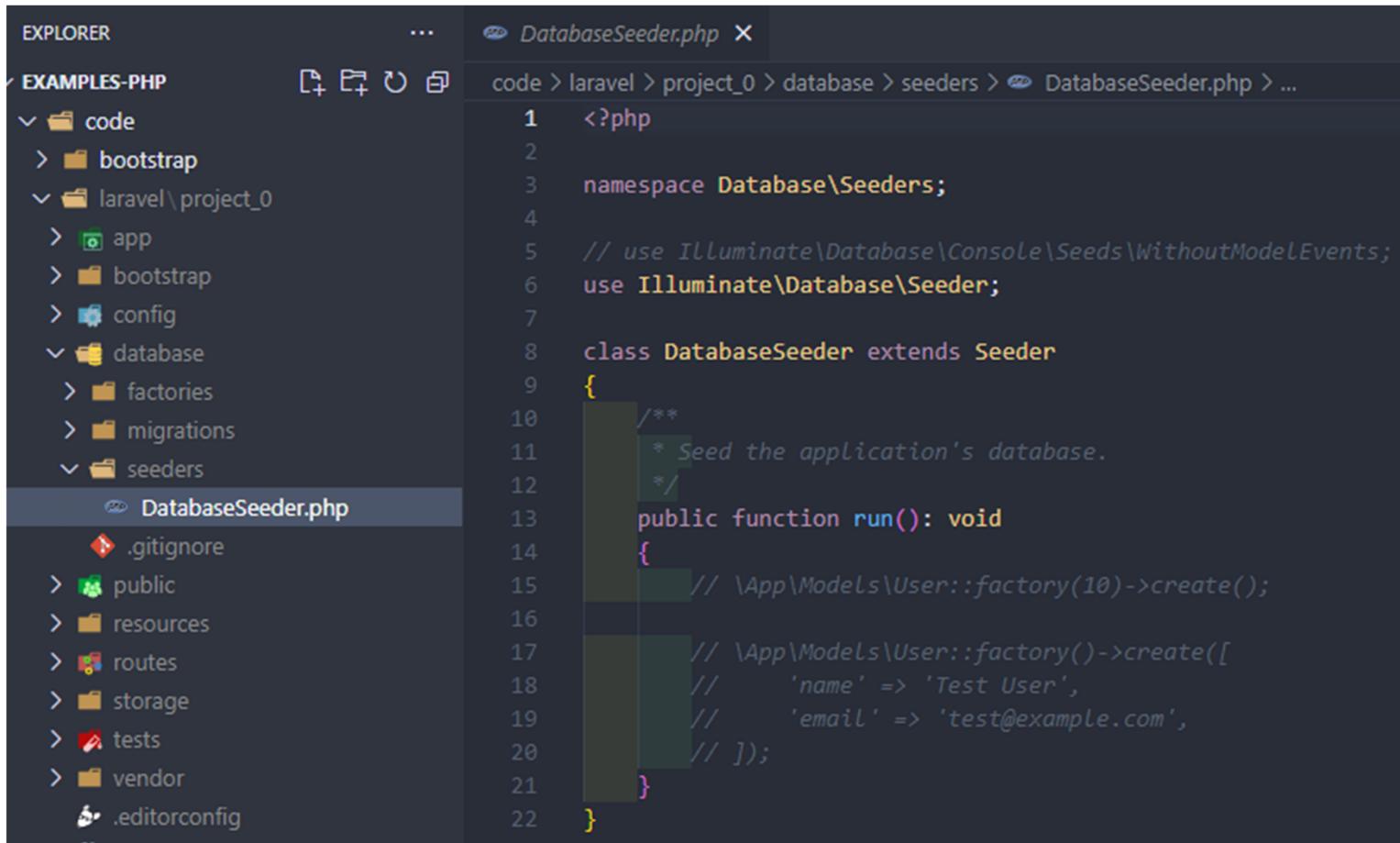
php artisan migrate

```
[kralos]--[¶main ≡]
[C:\xampp\htdocs\examples-PHP\code\laravel\project_0]
• ► php artisan migrate
  INFO | Preparing database.
  Creating migration table ..... 78ms DONE
•
  INFO | Running migrations.

  2014_10_12_000000_create_users_table ..... 62ms DONE
  2014_10_12_100000_create_password_reset_tokens_table ..... 14ms DONE
  2019_08_19_000000_create_failed_jobs_table ..... 99ms DONE
  2019_12_14_000001_create_personal_access_tokens_table ..... 24ms DONE
```



Comprobar el Login en Laravel



The screenshot shows a code editor interface with two main panes. The left pane, titled 'EXPLORER', displays the project structure of 'EXAMPLES-PHP'. It includes a 'code' folder containing 'bootstrap', and a 'laravel\project_0' folder containing 'app', 'bootstrap', 'config', 'database' (with 'factories' and 'migrations' subfolders), 'seeders' (containing 'DatabaseSeeder.php'), and other files like '.gitignore', 'public', 'resources', 'routes', 'storage', 'tests', 'vendor', and '.editorconfig'. The right pane shows the content of 'DatabaseSeeder.php'. The code is as follows:

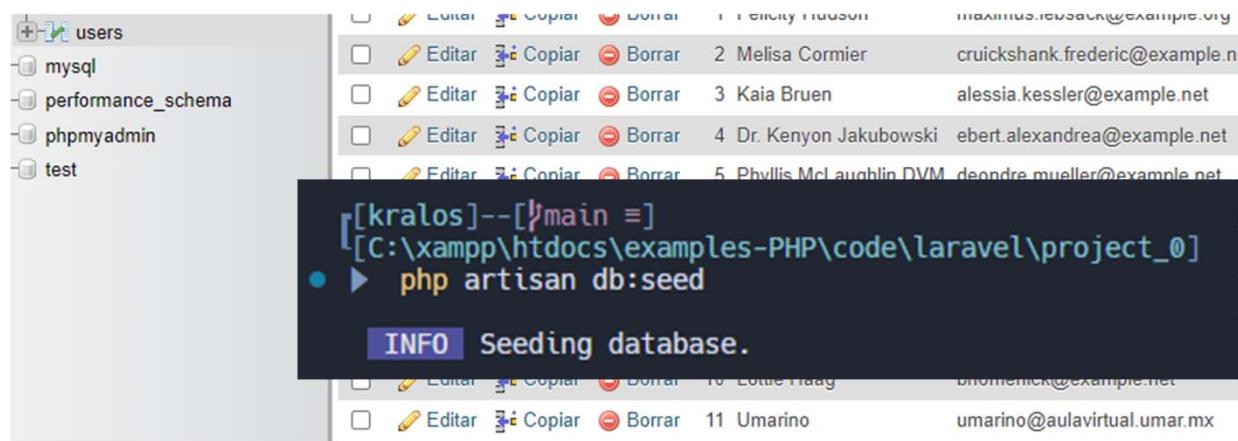
```
<?php  
namespace Database\\Seeders;  
  
// use Illuminate\\Database\\Console\\Seeds\\WithoutModelEvents;  
use Illuminate\\Database\\Seeder;  
  
class DatabaseSeeder extends Seeder  
{  
    /**  
     * Seed the application's database.  
     */  
    public function run(): void  
{  
        // \\App\\Models\\User::factory(10)->create();  
        // \\App\\Models\\User::factory()->create([  
        //     'name' => 'Test User',  
        //     'email' => 'test@example.com',  
        // ]);  
    }  
}
```

Comprobar el Login en Laravel

```
public function run(): void
{
    \App\Models\User::factory(10)->create();

    \App\Models\User::factory()->create([
        'name' => 'Umarino',
        'email' => 'umarino@aulavirtual.umar.mx',
        'password' => bcrypt('12345678')
    ]);
}
```

php artisan db:seed



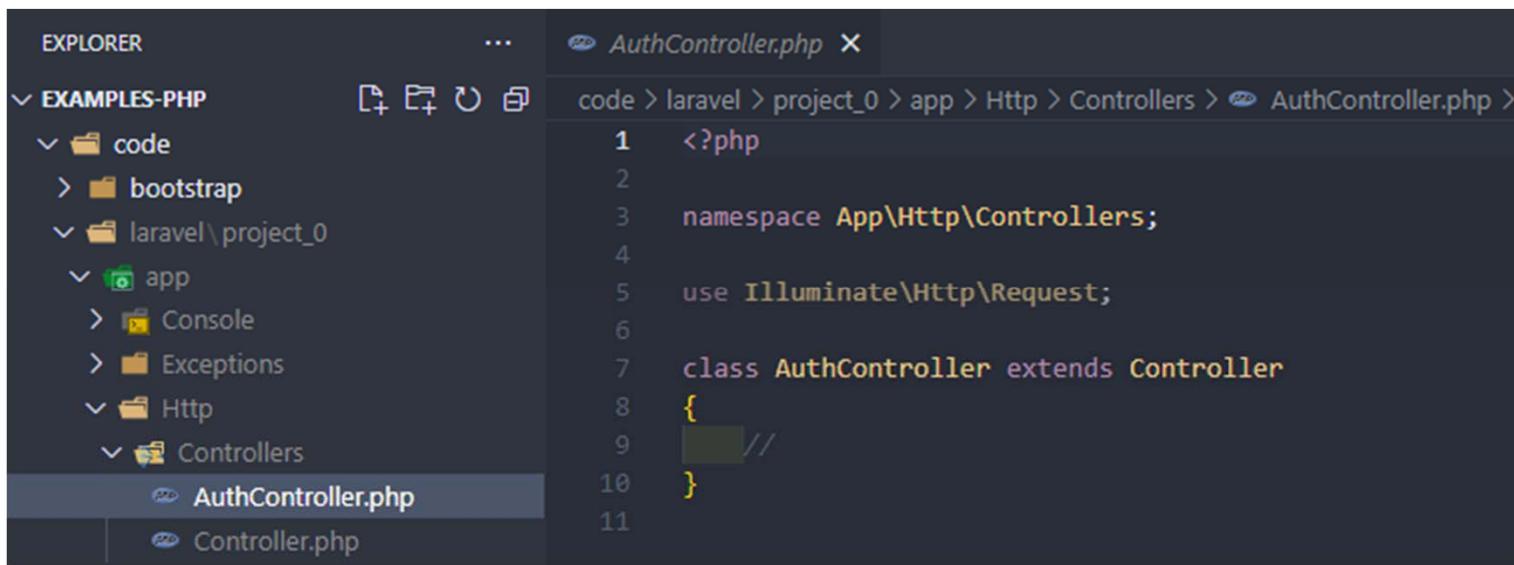
¿Bcrypt?

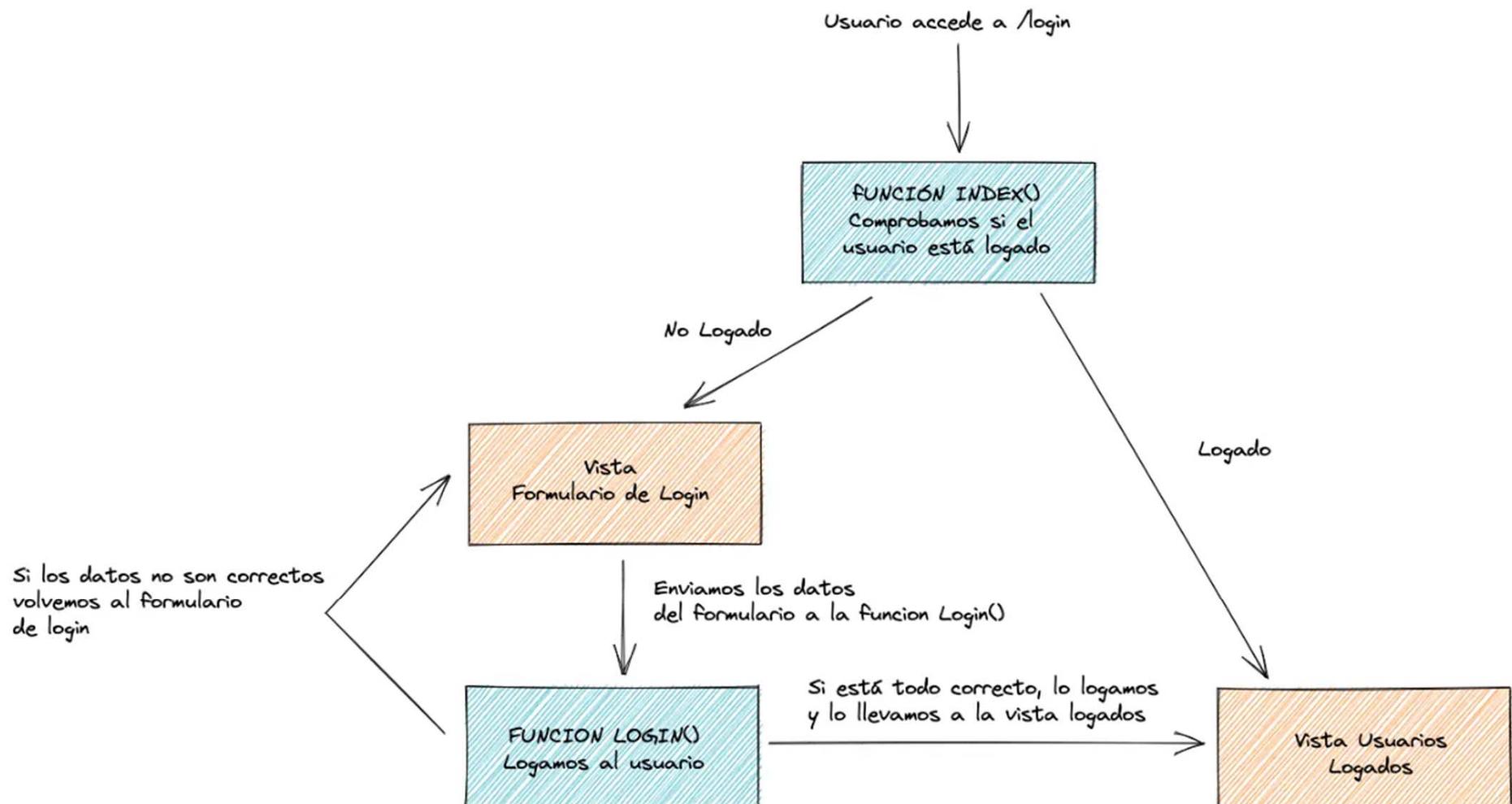
- Bcrypt es una función de hash de contraseñas y derivación de claves para contraseñas basada en el cifrado Blowfish. Se supone que la función de derivación de claves es lenta, lo que dificulta la fuerza bruta de la contraseña. Tanto el sistema operativo OpenBSD como el software OpenSSH emplean este algoritmo hash.

Controlador para la Autenticación

php artisan make:controller AuthController

```
[kralos]--[~] main
[C:\xampp\htdocs\examples-PHP\code\laravel\project_0]
● ▶ php artisan make:controller AuthController
INFO Controller [C:\xampp\htdocs\examples-PHP\code\laravel\project_0\app\Http\Controllers\AuthController.php] created successfully.
```





@dcreations_dev

Index

```
/*
 * Función que muestra la vista de logados o la vista con el formulario de Login
 */
public function index()
{
    // Comprobamos si el usuario ya está logado
    if (Auth::check()) {

        // Si está logado le mostramos la vista de logados
        return view('logados');
    }

    // Si no está logado le mostramos la vista con el formulario de login
    return view('login');
}
```

Login

```
/*
 * Función que se encarga de recibir los datos del formulario de login, comprobar que el usuario existe y
 * en caso correcto logar al usuario
 */
public function login(Request $request)
{
    // Comprobamos que el email y la contraseña han sido introducidos
    $request->validate([
        'email' => 'required',
        'password' => 'required',
    ]);

    // Almacenamos las credenciales de email y contraseña
    $credentials = $request->only('email', 'password');

    // Si el usuario existe lo logamos y lo llevamos a la vista de "logados" con un mensaje
    if (Auth::attempt($credentials)) {
        return redirect()->intended('logados')
            ->withSuccess('Logado Correctamente');
    }

    // Si el usuario no existe devolvemos al usuario al formulario de login con un mensaje de error
    return redirect("/")->withSuccess('Los datos introducidos no son correctos');
}
```

Logados

```
/*
 * Función que muestra la vista de logados si el usuario está logado y si no le devuelve al formulario de login
 * con un mensaje de error
 */
public function logados()
{
    if (Auth::check()) {
        return view('logados');
    }

    return redirect("/")->withSuccess('No tienes acceso, por favor inicia sesión');
}
```

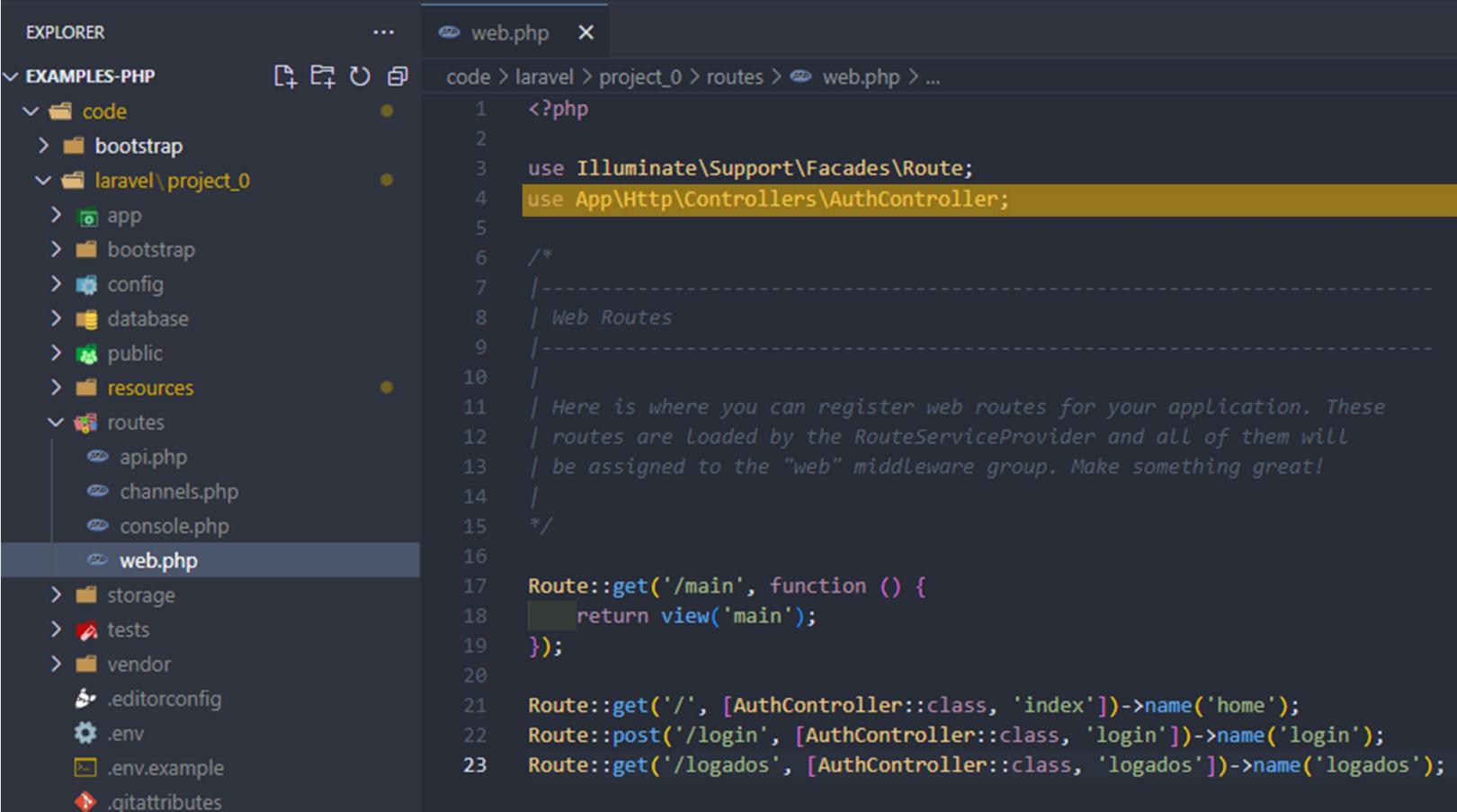
Crear las rutas para la Autenticación

The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar: Shows the project structure under "EXAMPLES-PHP". Key folders include "code", "bootstrap", "laravel\project_0" (containing "app", "bootstrap", "config", "database", "public", "resources"), and "routes" (containing "api.php", "channels.php", "console.php", and "web.php").
- web.php** tab in the editor: The file is located at "code > laravel > project_0 > routes > web.php".
- Code Content:**

```
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4
5  /*
6  |--------------------------------------------------------------------------
7  | Web Routes
8  |--------------------------------------------------------------------------
9  |
10 | Here is where you can register web routes for your application. These
11 | routes are loaded by the RouteServiceProvider and all of them will
12 | be assigned to the "web" middleware group. Make something great!
13 |
14 */
15
16 Route::get('/', function () {
17     return view('home');
18 });
19
```

Crear las rutas para la Autenticación



The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar on the left showing the project structure:

 - EXAMPLES-PHP** root folder
 - code** folder
 - laravel\project_0** folder
 - app**
 - bootstrap**
 - config**
 - database**
 - public**
 - resources**
 - routes** folder
 - api.php**
 - channels.php**
 - console.php**
 - web.php** (highlighted)
 - storage**
 - tests**
 - vendor**
 - .editorconfig**
 - .env**
 - .env.example**
 - .gitattributes**

- web.php** tab in the top bar.
- File content (highlighted line 4):

```
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\AuthController;
```

- Commented section (lines 11-15):

```
/ 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!
/- Actual route definitions (lines 17-23):


```
Route::get('/main', function () {
 return view('main');
});

Route::get('/', [AuthController::class, 'index'])->name('home');
Route::post('/login', [AuthController::class, 'login'])->name('login');
Route::get('/logados', [AuthController::class, 'logados'])->name('logados');
```

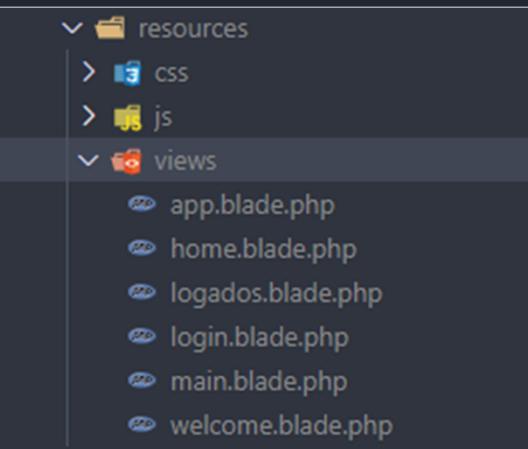

```

Crear las vistas para la Autenticación

```
[kralos]--[~/main =>]
[C:\xampp\htdocs\examples-PHP\code\laravel\project_0]
▶ php artisan make:view app
INFO View [C:\xampp\htdocs\examples-PHP\code\laravel\project_0\resources\views\app.blade.php] created successfully.

[kralos]--[~/main =>]
[C:\xampp\htdocs\examples-PHP\code\laravel\project_0]
▶ php artisan make:view login
INFO View [C:\xampp\htdocs\examples-PHP\code\laravel\project_0\resources\views\login.blade.php] created successfully.

[kralos]--[~/main =>]
[C:\xampp\htdocs\examples-PHP\code\laravel\project_0]
▶ php artisan make:view logados
INFO View [C:\xampp\htdocs\examples-PHP\code\laravel\project_0\resources\views\logados.blade.php] created successfully.
```



The screenshot shows a file explorer window with the following directory structure:

- resources
 - css
 - js
 - views
 - app.blade.php
 - home.blade.php
 - logados.blade.php
 - login.blade.php
 - main.blade.php
 - welcome.blade.php