



TALLER PRÁCTICO

PROYECTO PERSONAS

Fecha: 12/04/2023

Descargar el Proyecto si no lo tiene

1. Si no cuenta con el proyecto realizado en la guía No.1, descargue del campus, el proyecto personas o haciendo clic en el siguiente enlace <https://github.com/yovaromo-cecep/personas-app/tags>
2. Descomprima el proyecto en algún sitio de su computador
3. Corra el comando **composer install**
4. Corra el comando **npm install**

GENERANDO UN SISTEMA DE AUTENTICACIÓN

Laravel cuenta con varios paquetes que permiten implementar un sistema de autenticación, entre ellos: Laravel Breeze, Laravel Fortify, Laravel Jetstream, Laravel Passport, cada uno con sus propias características.

Vamos a trabajar con el más simple de ellos, Laravel Breeze. Laravel Breeze es una implementación mínima y simple de todas las funciones de autenticación de Laravel, incluido el inicio de sesión, el registro, el restablecimiento de contraseña, la verificación de correo electrónico y la confirmación de contraseña. Además, Breeze incluye una página de "perfil" simple donde el usuario puede actualizar su nombre, dirección de correo electrónico y contraseña.

La capa de vista predeterminada de Laravel Breeze se compone de plantillas Blade simples diseñadas con Tailwind CSS . O bien, Breeze puede montar su aplicación usando Vue o React and Inertia.

1. Antes de iniciar la instalación de Breeze, versione su proyecto y cree el commit Initial Project
2. Para realizarla instalación de Laravel Breeze, primero se debe correr la migración que contiene la creación de la tabla user, o crear manualmente la tabla con sus correspondientes campos.

Luego de contar con la tabla user, **se debe descargar el paquete:**

composer require laravel/breeze --dev

Una vez descargado el paquete verifique que archivos se modificaron.

Instalar el paquete

php artisan breeze:install

Si el instalador pregunta:

Which stack would you like to install? seleccione **0** (blade)

Would you like to install dark mode support? (yes/no) **[no]**

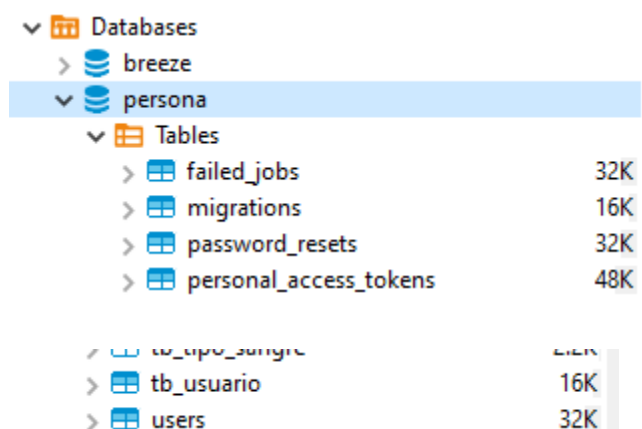
Would you prefer Pest tests instead of PHPUnit? (yes/no) **[no]**

Una vez instalado el paquete verifique que archivos se modificaron.

Correr de nuevo las migraciones

php artisan migrate

Si observa en su cliente sql (DBEaver o PhpMyAdmin) puede ver que su base de datos ahora cuenta con más tablas, creadas por las migraciones



Entre ellas la tabla user, que tiene la siguiente estructura

Properties

Data

ER Diagram

Table Name:

users

Engine:

InnoDB

Auto Increment:

1

Charset:

utf8mb4

Collation:

utf8mb4_unicode_ci

Description:

Columns

Constraints

Foreign Keys

References

Triggers

Indexes

Partitions

Statistics

DDL

Virtual

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra
id	1	bigint(20) unsigned	[v]	[v]	PRI		auto
name	2	varchar(255)	[v]	[]			
email	3	varchar(255)	[v]	[]	UNI		
email_verified_at	4	timestamp	[]	[]		NULL	
password	5	varchar(255)	[v]	[]			
remember_token	6	varchar(100)	[]	[]		NULL	
created_at	7	timestamp	[]	[]		NULL	
updated_at	8	timestamp	[]	[]		NULL	

Instalar las librerías Javascript

npm install

npm run dev

Al correr el último comando si el proyecto tiene configurado **webpack.mix** como herramienta compiladora de javascript, se generará un error. Esto lo puede corroborar si en la raíz del proyecto aparece el archivo `webpack.mix.js`. Por tanto, deberá migrar el proyecto a Vite, dado que Breeze opera con esta herramienta.

<https://laravel.com/docs/10.x/vite>

Nota: La versión de Laravel mínima debe ser **9.19.0**

Para verificar la versión de Laravel ejecute el comando

php artisan -V

3. Realice una nueva versión de su proyecto en su repositorio.
4. Se debe ejecutar la guía de migración de webpack a vite.

<https://github.com/laravel/vite-plugin/blob/main/UPGRADE.md#migrating-from-laravel-mix-to-vite>

- Se debe instalar Vite y el Plugin de Larvel Vite
npm install --save-dev vite laravel-vite-plugin

Una vez instalados los paquetes verifique que archivos se modificaron.

- Verifique que en la raíz del proyecto está el archivo vite.config.js, si no está creelo y pegue el siguiente código

```
import laravel from 'laravel-vite-plugin';
import { defineConfig } from 'vite';

export default defineConfig({
  plugins: [
    laravel({
      input: ["resources/css/app.css",
"resources/js/app.js"],
      refresh: true,
    }),
  ],
  resolve: {
    alias: {
      "@": "/resources/js",
    },
  },
});
```

- Actualizar los scripts del archivo package.json, el archivo debe quedar como se observa el siguiente código

```
"scripts": {  
    "dev": "vite",  
    "build": "vite build"  
},
```

- Verifique que el archivo **app.js** de la carpeta `resources/js` no contenga órdenes **require**, si las tiene se debe cambiar por **import**, dado que vite solo trabaja con import

```
import './bootstrap';
```

- También debe verificar el archivo **bootstrap.js** de la misma carpeta. Si lo desea podría reemplazar el contenido por lo el archivo que tiene el repositorio de Laravel <https://github.com/laravel/laravel/blob/10.x/resources/js/bootstrap.js>

- Agregar al final del archivo `.env`, las llaves de **VITE_PUSHER**, como se observa

```
VITE_PUSHER_APP_KEY="${PUSHER_APP_KEY}"  
VITE_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

Puede quitar las de **MIX_PUSHER**

- Verificar que el archivo **app.blade.php** de la carpeta `resource/views/layouts` tenga la importación del script de **vite**

```
<!-- Scripts -->  
  
@vite(['resources/css/app.css', 'resources/js/app.js'])
```

- Agregar al archivo **resources/js/app.js**

```
import './bootstrap';  
import '../css/app.css';
```

- **Desinstalar Laravel Mix**
npm remove laravel-mix
- **Eliminar el archivo webpack.mix.js**
rm webpack.mix.js (Linux)
del webpack.mix.js (Windows)
- Ignorar el versionamiento de la carpeta **/public/build** abra el archivo **.gitignore** y agregue esa ruta

```
/public/storage  
  
/public/build  
  
/storage/*.key
```

- Limpiar las vistas con **php artisan view:clear**
- **Corra Laravel Vite con**
npm run dev

Si todo quedó bien, deberá correr vite y se deberá observar un despliegue de pantalla como el siguiente

```
cal: npm run dev

VITE v4.2.1 ready in 848 ms
  Local:   http://127.0.0.1:5173/
  Network: use --host to expose
  press h to show help

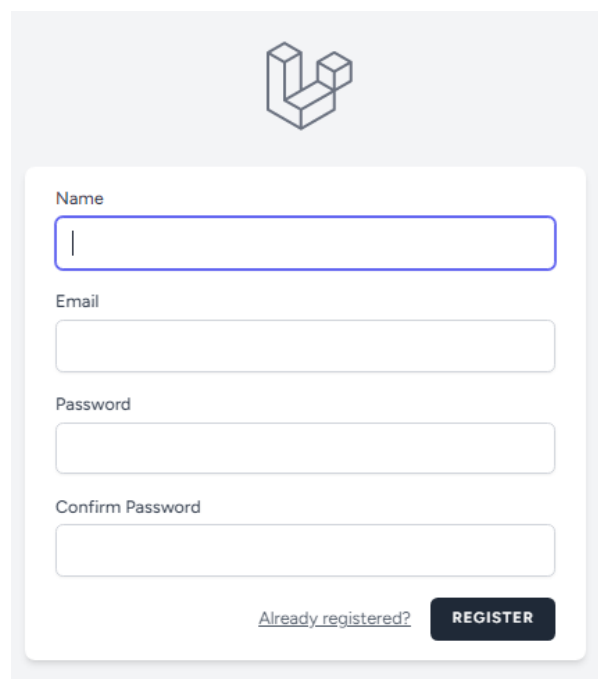
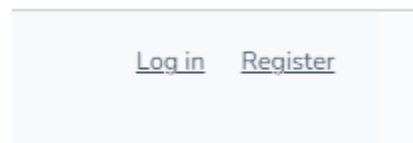
LARAVEL v9.52.4 plugin v0.7.4

  APP_URL: http://localhost
```

- Corra el proyecto de Laravel

php artisan serve

Ahora su aplicación cuenta con la funcionalidad para: Crear Usuario, Autenticar, Cambiar Clave

A registration form with a light gray background. At the top center is a 3D logo made of blocks forming the letter 'L'. Below the logo is a white rounded rectangle containing the form fields. The fields are labeled 'Name', 'Email', 'Password', and 'Confirm Password'. Each label is followed by an input field. At the bottom right of the form is a dark blue button with the word 'REGISTER' in white. To the left of the button is a link that says 'Already registered?'.

3D 'L' logo

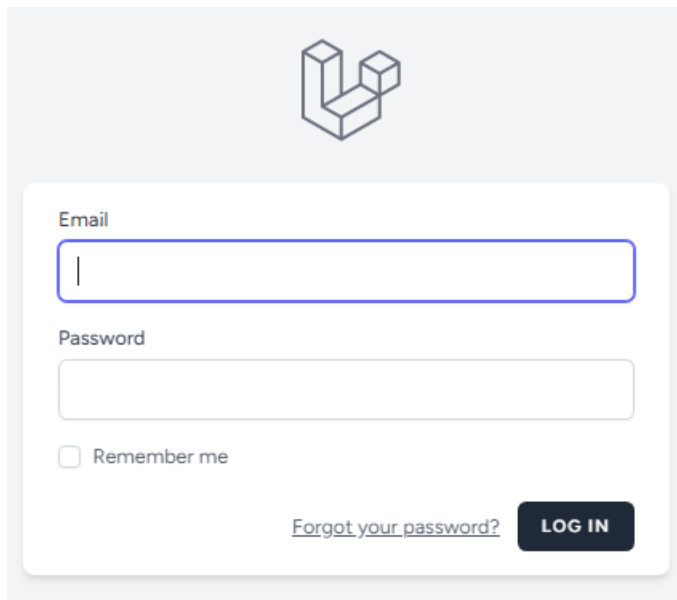
Name

Email

Password

Confirm Password

[Already registered?](#) **REGISTER**



The image shows a login form with a light gray background. At the top center is a logo consisting of three stacked cubes. Below the logo is a white rectangular box containing the form fields. The first field is labeled 'Email' and has a blue border. The second field is labeled 'Password' and has a light gray border. Below the password field is a checkbox labeled 'Remember me'. At the bottom right of the box is a link that says 'Forgot your password?' and a dark blue button with the text 'LOG IN' in white.

Si observa el archivo de rutas **web.php** se dará cuenta que hay nuevas rutas configuradas, y las que usted había configurado ya no están. También puede observar que quizá en la última línea tiene la orden

```
require __DIR__.'/auth.php';
```

- Lo que indica que al archivo se le está incluyendo el contenido del archivo **auth.php** que se ha agregado a la carpeta **routes**, el cual tiene configurados rutas a los controladores:
 - **RegisteredUserController**
 - **AuthenticatedSessionController**
 - **PasswordResetLinkController**
 - **NewPasswordController**
 - **EmailVerificationPromptController**
 - **VerifyEmailController**
 - **EmailVerificationNotificationController**
 - **ConfirmablePasswordController**
 - **PasswordController**
 - **AuthenticatedSessionController**

Los cuales son creados por Breeze y son los que tienen toda la funcionalidad que tiene que ver con la autenticación y autorización.

- Por tanto, es aconsejable que si ya ha configurado rutas y luego realiza la instalación de Breeze, debe copiar las rutas configuradas y luego de la instalación volverla a pegar

Agregue al archivo web.php las siguientes rutas:

```
Route::get('/comunas', [ComunaController::class,
'index'])->name('comunas.index');
Route::post('/comunas', [ComunaController::class,
'store'])->name('comunas.store');
Route::get('/comunas/create', [ComunaController::class,
'create'])->name('comunas.create');
Route::delete('/comunas/{comuna}', [ComunaController::class,
'destroy'])->name('comunas.destroy');
Route::put('/comunas/{comuna}', [ComunaController::class,
'update'])->name('comunas.update');
Route::get('/comunas/{comuna}/edit', [ComunaController::class,
'edit'])->name('comunas.edit');
```

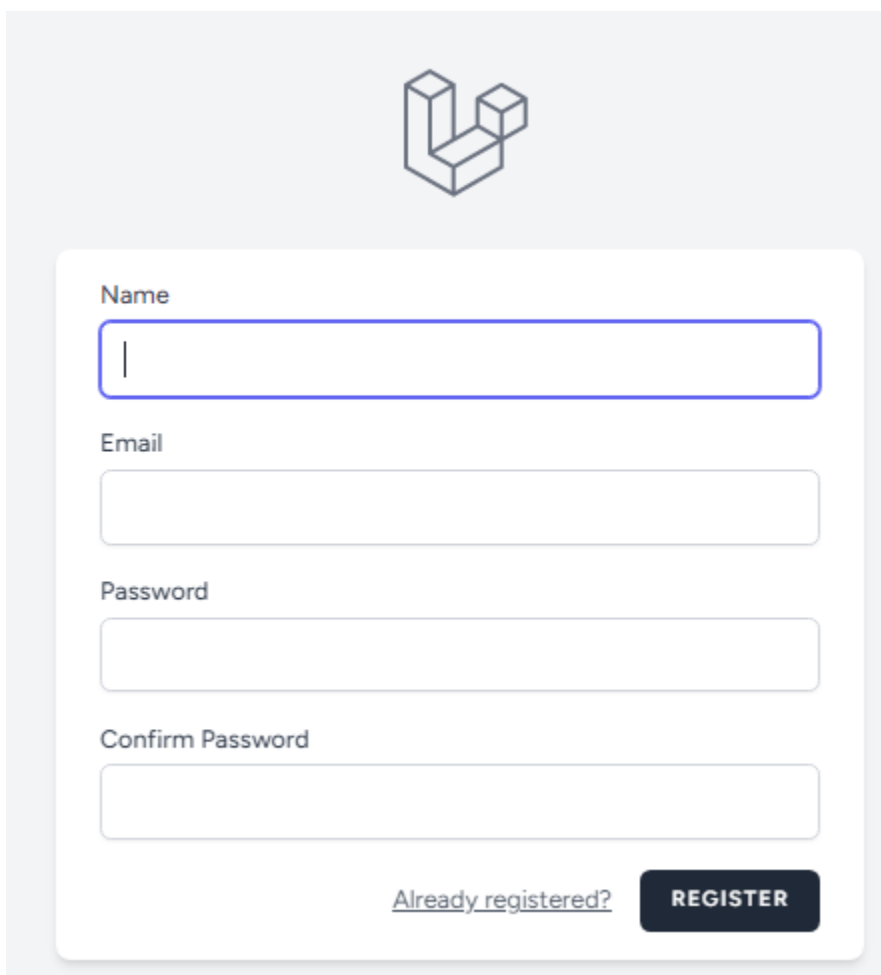
Verifique que en la parte superior del archivo se haya importado el controlador

```
use App\Http\Controllers\ComunaController;
```

- Si llama la ruta, observa que ahora levanta la funcionalidad de comunas

CREANDO USUARIO

Su aplicación ya cuenta con las funcionalidades para registrar usuarios, autenticar y desautenticar, invoque su aplicación <http://localhost:8000/> vaya al enlace Register y cree un nuevo usuario



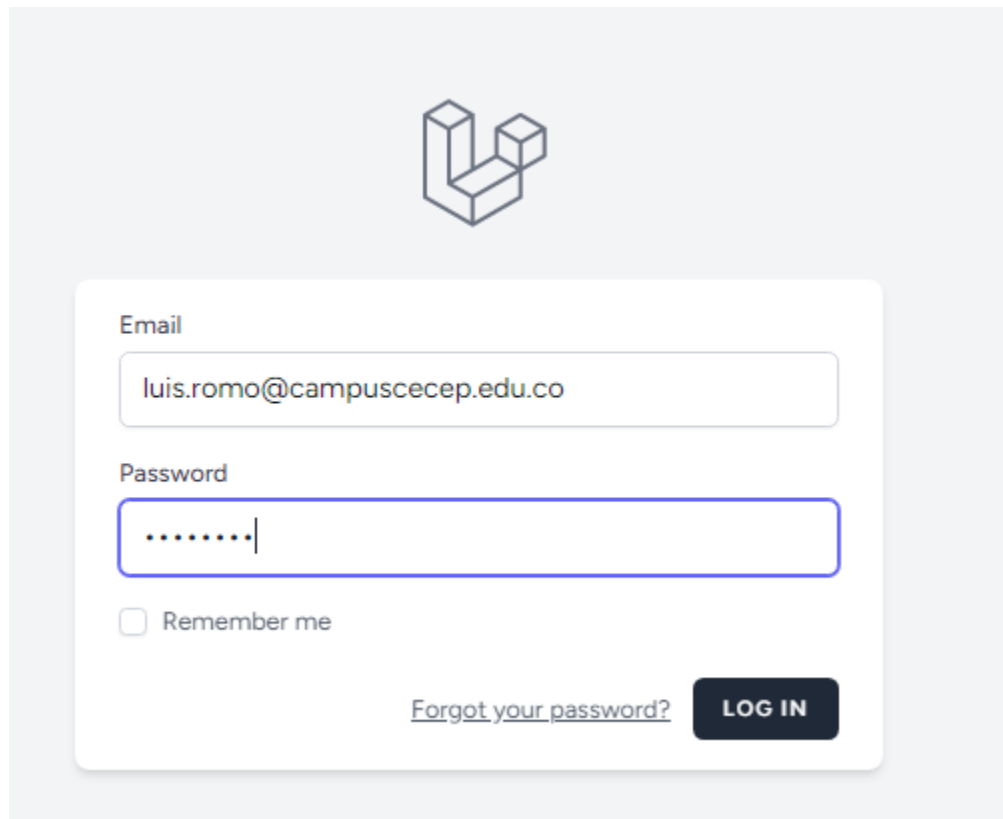
The image shows a registration form on a light gray background. At the top center is a 3D logo made of gray cubes forming an 'L' shape. Below the logo is a white rounded rectangle containing the form fields. The fields are labeled 'Name', 'Email', 'Password', and 'Confirm Password'. Each label is followed by a text input field. At the bottom of the form, there is a link that says 'Already registered?' and a dark blue button with the word 'REGISTER' in white capital letters.


Si el usuario se crea con éxito, se autenticará y lo llevará a la vista **Dashboard**, al lado izquierdo podrá observar un enlace con el nombre del usuario y la opción **Logout**.

También podrá consultar que en la tabla user de su base de datos hay un nuevo registro con los datos del usuario registrado.

users							
Enter a SQL expression to filter results (use Ctrl+Space)							
	id	name	email	email_verified_at	password	remember_token	
1	1	yovaromo	luis.romo@campuscecep.edu.co	[NULL]	\$2y\$10\$Ve5zaYy3GjQh7nIAwjQfyeYUgj7odGtrTRNiRd8cBkzidhdsc6i	[NULL]	

Salga de la aplicación, y pruebe la opción de autenticar (login) con los datos del usuario recién creado:





Email

Password

☐ Remember me

[Forgot your password?](#)

Si todo va bien, deberá cargar de nuevo la vista Dashboard.

PROTEGIENDO LAS RUTAS DE COMUNA PARA QUE SOLO ESTÉ DISPONIBLE SI EL USUARIO ESTÁ AUTENTICADO

- Configure las rutas de comunas para que solo se puedan acceder si el usuario está autenticado, para ello encierre las rutas en un middleware auth, dispuesto por Breeze

```
Route::middleware('auth')->group(function () {

    Route::get('/comunas', [ComunaController::class, 'index'])->name('comunas.index');

    Route::post('/comunas', [ComunaController::class, 'store'])->name('comunas.store');

    Route::get('/comunas/create', [ComunaController::class, 'create'])->name('comunas.create');

    Route::delete('/comunas/{comuna}', [ComunaController::class, 'destroy'])->name('comunas.destroy');

    Route::put('/comunas/{comuna}', [ComunaController::class, 'update'])->name('comunas.update');

    Route::get('/comunas/{comuna}/edit', [ComunaController::class, 'edit'])->name('comunas.edit');

});
```

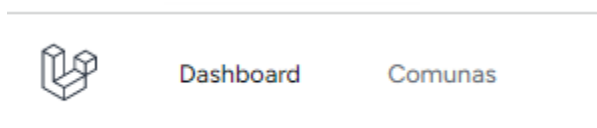
- Si intenta acceder a la ruta <http://localhost:8000/comunas> y el no hay usuario autenticado, observará que lo redirecciona a la vista de autenticación. Si autentica y vuelve a invocar la ruta, observa que si lo deja acceder. Esto debido a que las rutas fueron configuradas con el middleware auth quien es el que realiza el trabajo.

AGREGAR A LA VISTA DASHBOARD UN ENLACE PARA LA RUTA COMUNA

- Abra el archivo **navigation.blade.php** que está en la carpeta **resources/views/layouts** y agregue debajo de la ruta Dashboard lo siguiente

```
<div class="hidden space-x-8 sm:-my-px sm:ml-10 sm:flex">
  <x-nav-link
:href="route('comunas.index')":active="request()->routeIs('comunas.i
ndex')">
                                {{ __('Comunas') }}
  </x-nav-link>
</div>
```

- Si está autenticado puede observar que ahora cuenta con un enlace Comunas en el menú de la vista Dashboard



Si accede a ella puede observar que carga la vista de comunas

- Sin embargo, puede observar que la vista de comunas levanta en una interfaz diferente, para que cague en la la estructura del dashboard debemos ajustar la vista de comuna para que utilice el layout.
- Abra el archivo **index.blade.php** de la carpeta comunas de las vistas y reemplace el código con el siguiente código:

```
<x-app-layout>
  <x-slot name="header">
    <h2 class="font-semibold text-xl text-gray-800 leading-tight">
      {{ __('Communes') }}
    </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">
        <a href="{{ route('comunas.create') }}"
          class="bg-green-700 hover:bg-green-900 text-white
font-bold py-2 px-4 rounded ml-2">Add</a>
        <table class="table">
          <thead>
            <tr>
              <th scope="col">Code</th>
              <th scope="col">Commune</th>
              <th scope="col">Municipality</th>
              <th scope="col">Actions</th>
            </tr>
          </thead>
          <tbody>
            @foreach ($comunas as $comuna)
              <tr>
                <th scope="row">{{ $comuna->comu_codi
}}</th>
                <td>{{ $comuna->comu_nomb }}</td>
                <td>{{ $comuna->muni_nomb }}</td>
                <td>
                  <a href="{{ route('comunas.edit',
['comuna' => $comuna->comu_codi]) }}"
                    class="bg-blue-500
hover:bg-blue-700 text-white font-bold py-2 px-4 rounded">
                    Edit </a></td>
                  <form action="{{
route('comunas.destroy', ['comuna' => $comuna->comu_codi]) }}"
                    method='POST' style="display:
inline-block">
                    @method('delete')
                    @csrf
                    <input
                      class="bg-red-500
hover:bg-red-700 text-white font-bold py-2 px-4 rounded ml-2"
                      type="submit" value="Delete">
                    </form>
                  </td>
                </tr>
              @endforeach
            </tbody>
          </table>
        </div>
      </div>
    </div>
  </div>
</x-app-layout>

```

- Vuelva a cargar las comunas y ahora observará que se carga en la estructura de la aplicación.
- Es hora que ajuste las fiestas para `new.blade.php` y `edit.blade.php` para que se carguen en la estructura de la aplicación

TALLER EN SISVEN

- Instalar Breeze en el proyecto SISVEN
- Proteger con el middleware de todas las rutas creadas
- Crear las opciones de menú en la vista Dashboard
- Ajustar las vistas para que carguen en la estructura

Nota: Por cada cambio realizado debe hacer un commit