

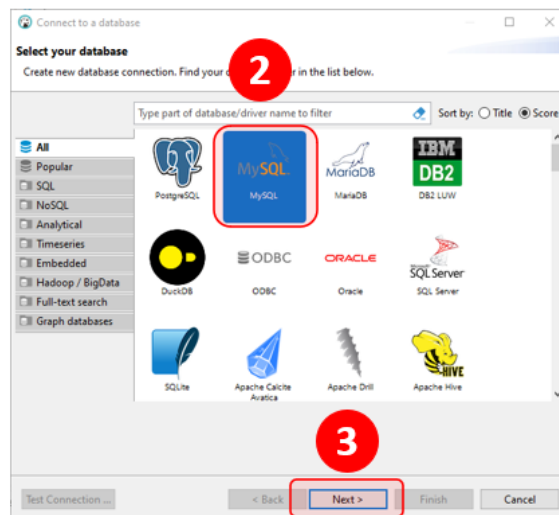
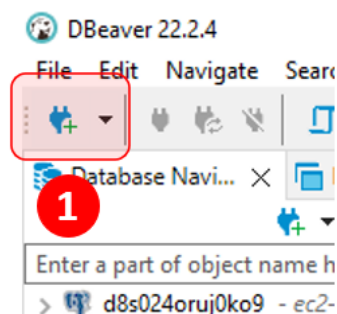


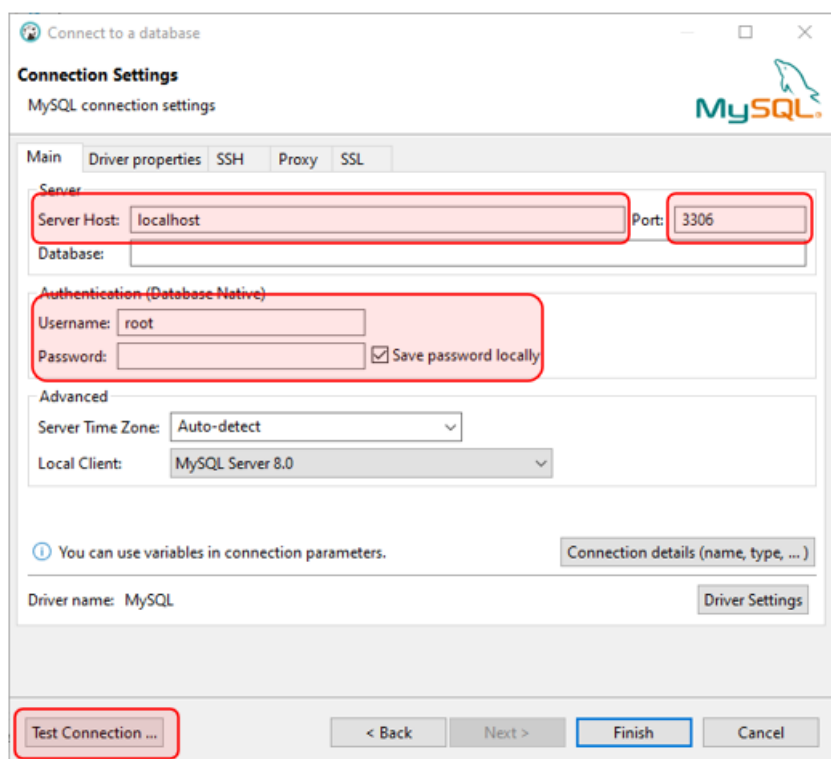
TALLER PRÁCTICO

Fecha: 22/03/2023

1. Configurando la Base de Datos

- 1.1. Corra la aplicación XAMPP Control y levante el servidor de BD
- 1.2. Corra la aplicación DBEaver
- 1.3. Cree una nueva conexión en DBEaver al motor de BD MySQL





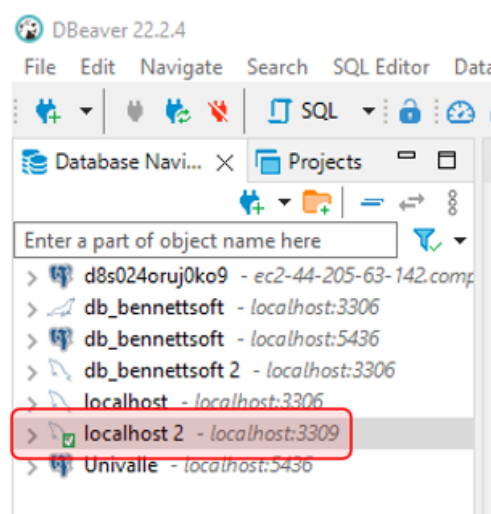
Configure los parámetros de conexión: Dirección IP o DNS donde está el Servidor de base BD (Server Host: **localhost**), puerto (**3306**) y credenciales del usuario con el que se conectará (Username: **root**).

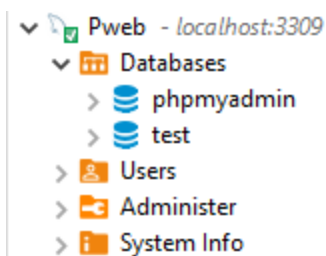
Finalmente pruebe si la conexión es exitosa, presionando clic en el botón **Test Connection**

Si la conexión es exitosa, presione clic en el **botón Finish**, con lo cual aparecerá la conexión con nombre localhost en el árbol del **Database Navigation**.

Puede cambiar el nombre de la conexión presionando click derecho en el nombre de la conexión (localhost) y luego clic en la opción **Edit Connection**

En la ventana **Connection "localhost"** **Configuration** presione clic en la opción **General** y cambie el nombre de la conexión en la caja de texto de **Connection Name**, para la práctica nuestra práctica a **Pweb**, presione clic en el botón **OK** y finalmente acepte el cambio presionando clic en el botón **YES**.



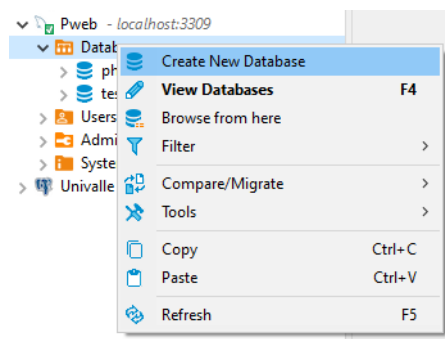


Despliegue el contenido de la conexión, presionando click en el símbolo > que está al lado izquierdo del nombre de la conexión (**Pweb**). Despliegue el contenedor **Databases** para que observe las bases de datos creadas en dicha conexión, para la imagen las

base de datos **phpmyadmin** y **test**.

- 1.4.** Crear la base de datos **Persona**, presionando clic derecho sobre el contenedor Databases de la conexión **Pweb** y luego click sobre la opción **Create New Database**.

En la ventana emergente que aparece escriba en nombre de la Base de Datos en la caja de texto **Database Name**, para nuestra práctica el nombre será **Persona**, finalmente presione clic en el botón **OK**.

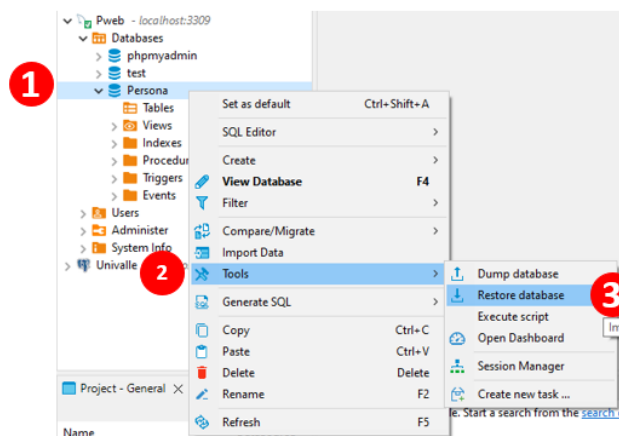


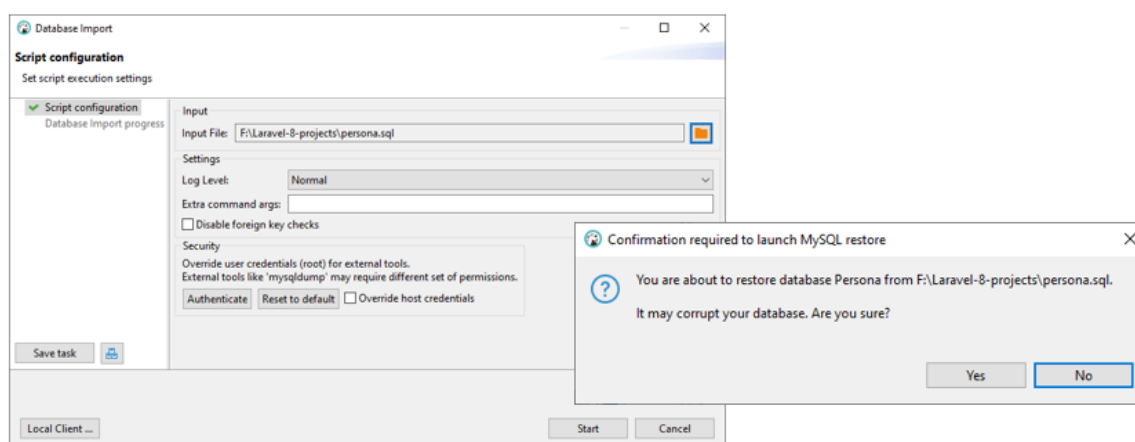
El nombre de la base de datos **Persona** aparecerá en el contenedor Databases de la conexión **Pweb**, puede desplegar su contenido y observa que no hay tablas para esta.

- 1.5.** Descargue del Moodle la base de datos de ejemplo que se encuentra en la sesión # 8 (Base de datos de Ejemplo), presione clic derecho sobre el enlace y luego **guardar enlace como**, grabe el archivo en el carpeta de su preferencia. El archivo contiene las sentencias SQL para la creación de tablas y registros en ellas.

- 1.6.** Importar el archivo de descargado (**personas.sql**) a la base de datos creada (**Persona**), presiona clic derecho sobre el nombre de la base de datos (**Persona**), selecciona la opción **tools** y luego la opción **Restore database**.

En la ventana Database Import seleccione el archivo descargado (**Persona.sql**) y presione clic en el botón **Start**.

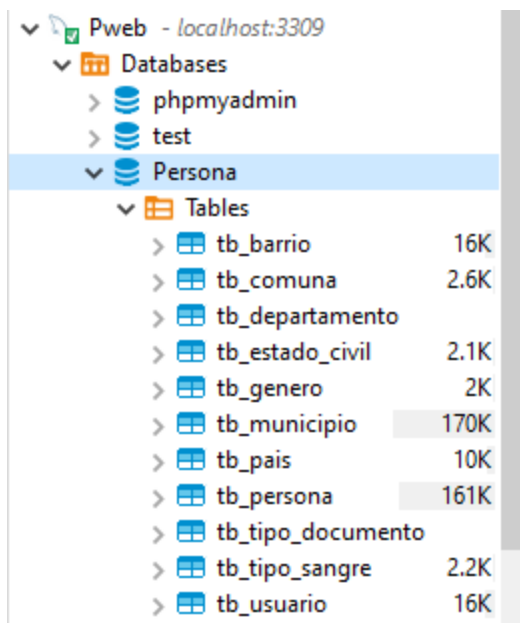




Si aparece el mensaje para confirmación, presione clic en el botón **Yes**. Espere que se ejecute el proceso, y solo cuando aparezca que ha terminado la restauración, presione clic en el botón **Cancel**.

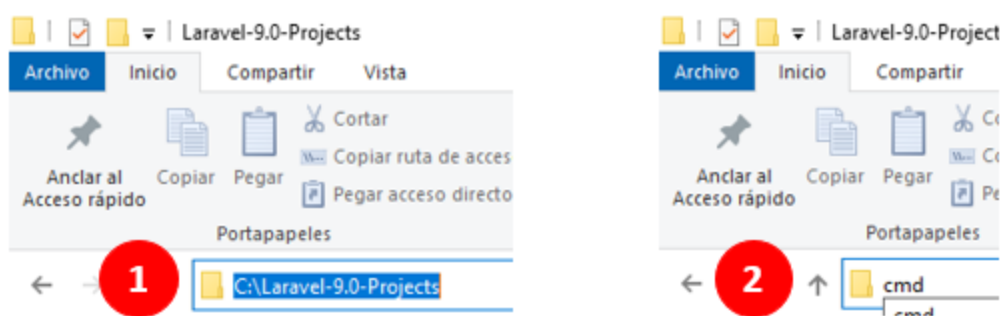
Observe que en el contenedor **Tables** de la base de datos **Persona** aparecerán las tablas importadas. Puede presionar doble clic sobre el nombre de alguna de ellas y podrá ver los registros que contienen, por ejemplo presione clic en el nombre **tb_comuna**.

En la pestaña que se despliega con el título **tb_comuna**, puede observar tres pestañas: Properties, Data y Diagrama. En la pestaña **Properties** puede ver la estructura de la tabla, mientras que en la pestaña **Data** observa los registros de la misma.



2. Crear el proyecto Personas-app en Laravel 9.0

- 2.1. En el explorador de archivos de Windows, crear la carpeta laravel-9.0-projects en la raíz del disco duro C:\
- 2.2. Presiona clic en la barra de direcciones del explorador de windows, escriba **cmd** y presione la tecla **enter**, con lo cual se abrirá una ventana de línea de comandos situada en la carpeta c:\Laravel-9.0-projects.



- 2.3. Comprobar que se tiene instalado php, para la práctica se requiere la versión 8.0 o superior. El comando para comprobar la versión es **php - - version**.

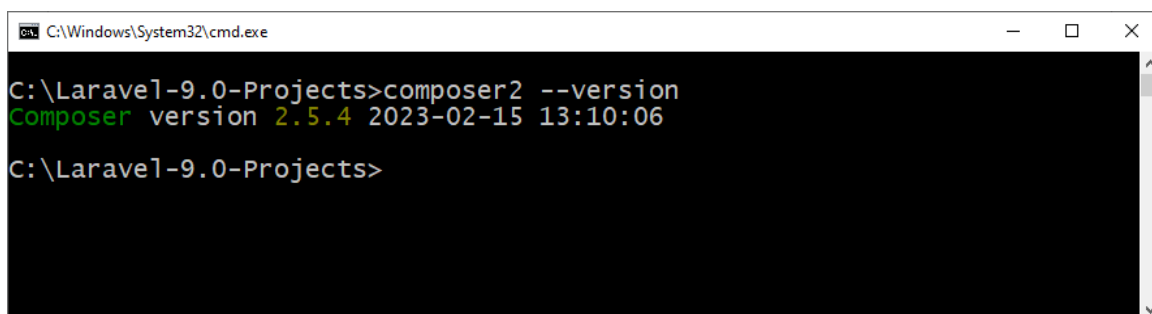
Nota: en las imágenes verá que el comando digitado es **php82**, ésto debido a que en la máquina donde se preparó la guía tiene instalada varias versiones de php y al archivo ejecutable php.exe de la versión 8.2 se le cambió el nombre a **php82.exe**, para el caso de la sala de cómputo u otro computador el comando será **php**.

```
C:\Windows\System32\cmd.exe
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Laravel-9.0-Projects>php82 --version
PHP 8.2.0 (cli) (built: Dec 6 2022 15:31:23) (ZTS Visual C++ 2019 x64)
Copyright (c) The PHP Group
Zend Engine v4.2.0, Copyright (c) Zend Technologies

C:\Laravel-9.0-Projects>
```

- 2.4. Comprobar que tenga instalado composer, escribiendo en el prompt del sistema **composer - - version**



```
C:\Windows\System32\cmd.exe

C:\Laravel-9.0-Projects>composer2 --version
Composer version 2.5.4 2023-02-15 13:10:06

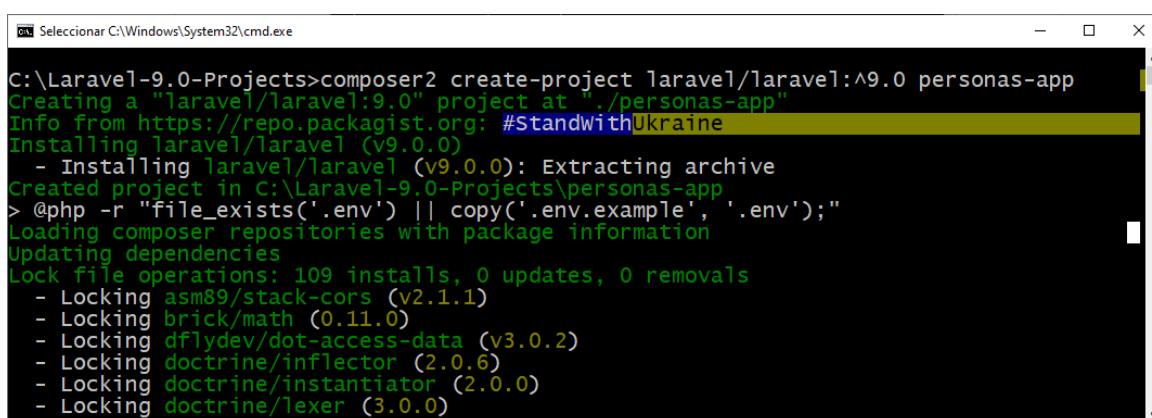
C:\Laravel-9.0-Projects>
```

- 2.5. Crear el proyecto en Laravel en la versión 9.x, llamado **personas-app** utilizando **composer**

Nota: en las imágenes verá que el comando digitado es **composer2**, ésto debido a que en la máquina donde se preparó la guía tiene instalada varias versiones de composer y al archivo por lotes que ejecuta composer versión 2 se le renombró a **composer2.bat**, para el caso de la sala de cómputo u otro computador el comando será **composer**.

Comando: composer create-project laravel/laravel:^9.0 personas-app

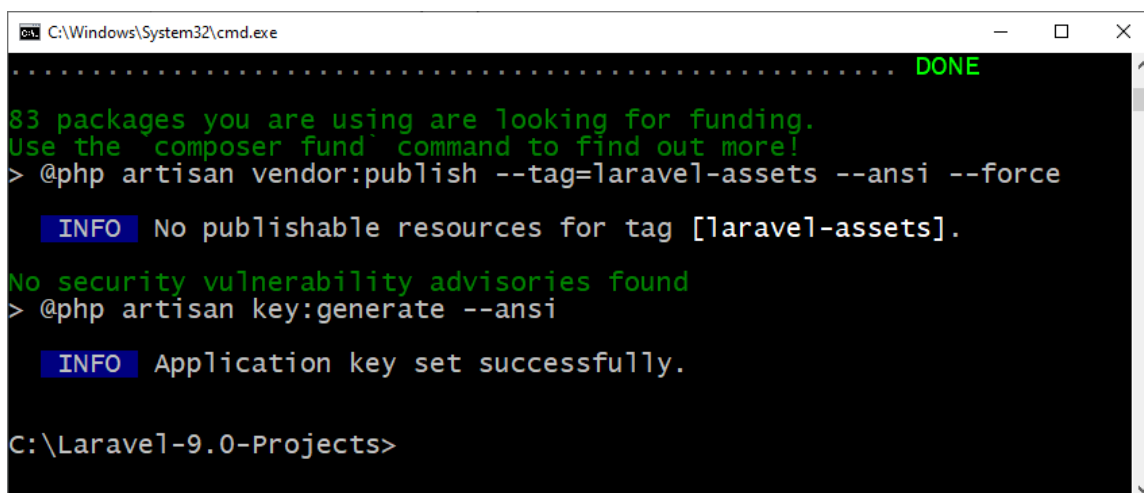
<https://laravel.com/docs/9.x/installation#your-first-laravel-project>



```
Selecccionar C:\Windows\System32\cmd.exe

C:\Laravel-9.0-Projects>composer2 create-project laravel/laravel:^9.0 personas-app
Creating a "laravel/laravel:9.0" project at "../personas-app"
Info from https://repo.packagist.org: #StandWithUkraine
Installing laravel/laravel (v9.0.0)
- Installing laravel/laravel (v9.0.0): Extracting archive
Created project in C:\Laravel-9.0-Projects\personas-app
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 109 installs, 0 updates, 0 removals
- Locking asm89/stack-cors (v2.1.1)
- Locking brick/math (0.11.0)
- Locking dflydev/dot-access-data (v3.0.2)
- Locking doctrine/inflector (2.0.6)
- Locking doctrine/instantiator (2.0.0)
- Locking doctrine/lexer (3.0.0)
```

Composer descargará el proyecto, el tiempo de descarga será relativo a la velocidad de conexión de internet que se tenga. Una vez termine la descarga composer le informará y retornará al prompt del sistema operativo.



```
C:\Windows\System32\cmd.exe
..... DONE
83 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

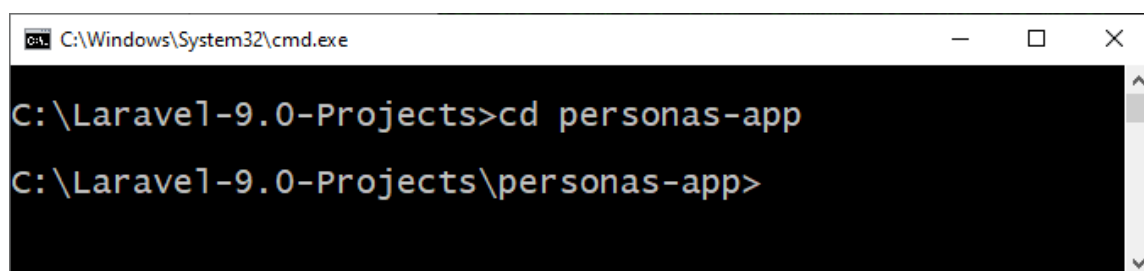
  INFO  No publishable resources for tag [laravel-assets].

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

  INFO  Application key set successfully.

C:\Laravel-9.0-Projects>
```

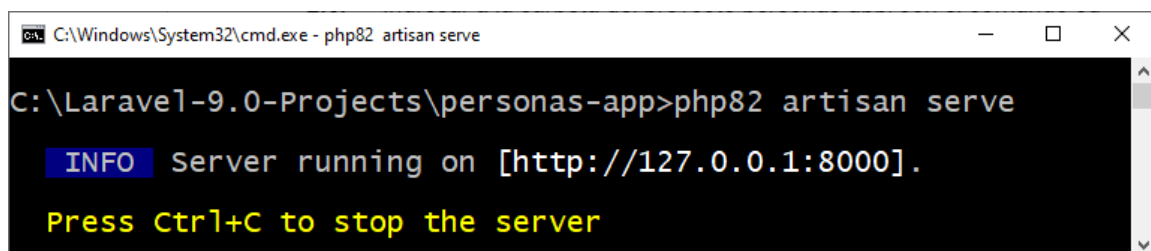
- 2.6. Ingresar a la carpeta del proyecto **personas-app**, con el comando **cd personas-app**



```
C:\Windows\System32\cmd.exe

C:\Laravel-9.0-Projects>cd personas-app
C:\Laravel-9.0-Projects\personas-app>
```

- 2.7. Ejecutar el proyecto utilizando el comando **php artisan serve**, el cual levantará un servidor web local (Apache) y pondrá a disposición la aplicación, por defecto en la dirección <http://127.0.0.1:8000>, 8000 es el puerto del servidor. La dirección 127.0.0.1 es equivalente a localhost.



```
C:\Windows\System32\cmd.exe - php82 artisan serve

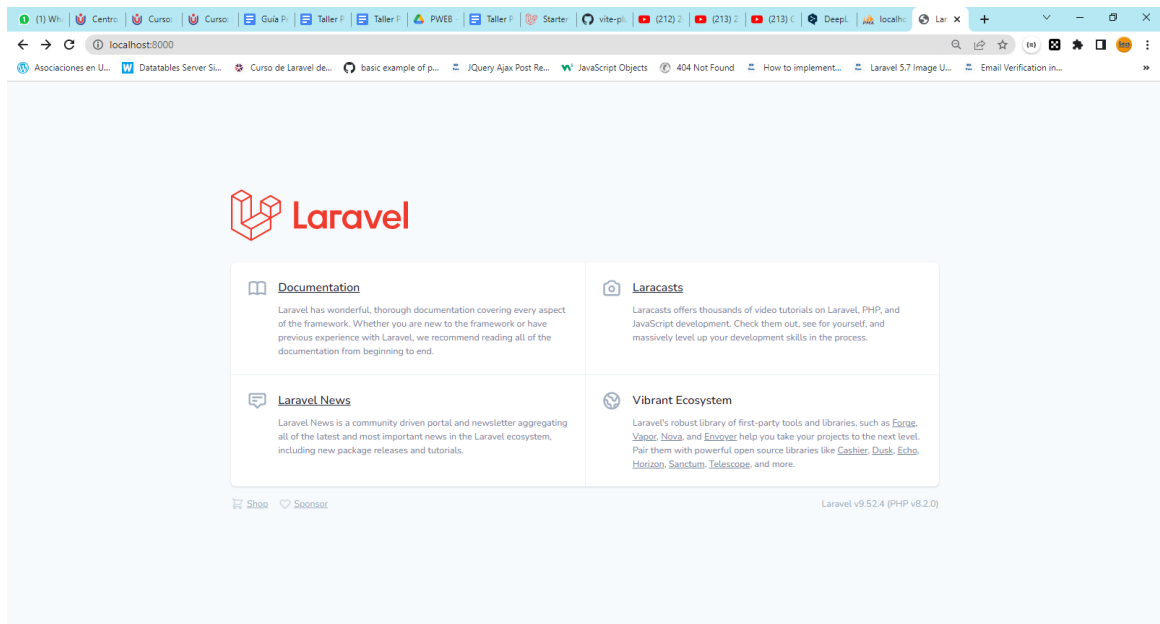
C:\Laravel-9.0-Projects\personas-app>php82 artisan serve

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

Press Ctrl+C to stop the server
```

El servidor lo puede detener presionando las teclas CTRL + C.

Abra un navegador y escriba la dirección <http://127.0.0.1:8000>, deberá cargar la aplicación, como se observa en la siguiente imagen:



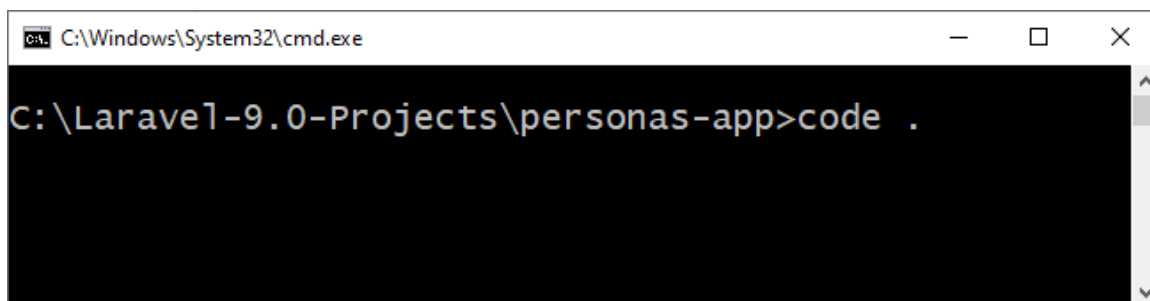
En la parte inferior derecha del despliegue de pantalla se observa la versión de Laravel y PHP (Laravel v9.52.4 (PHP v8.2.0)).

- 2.8.** Detenga la ejecución del proyecto, presionando CTRL + C en la ventana de línea de comando donde lo ejecutó.

```
C:\Windows\System32\cmd.exe
Press Ctrl+C to stop the server

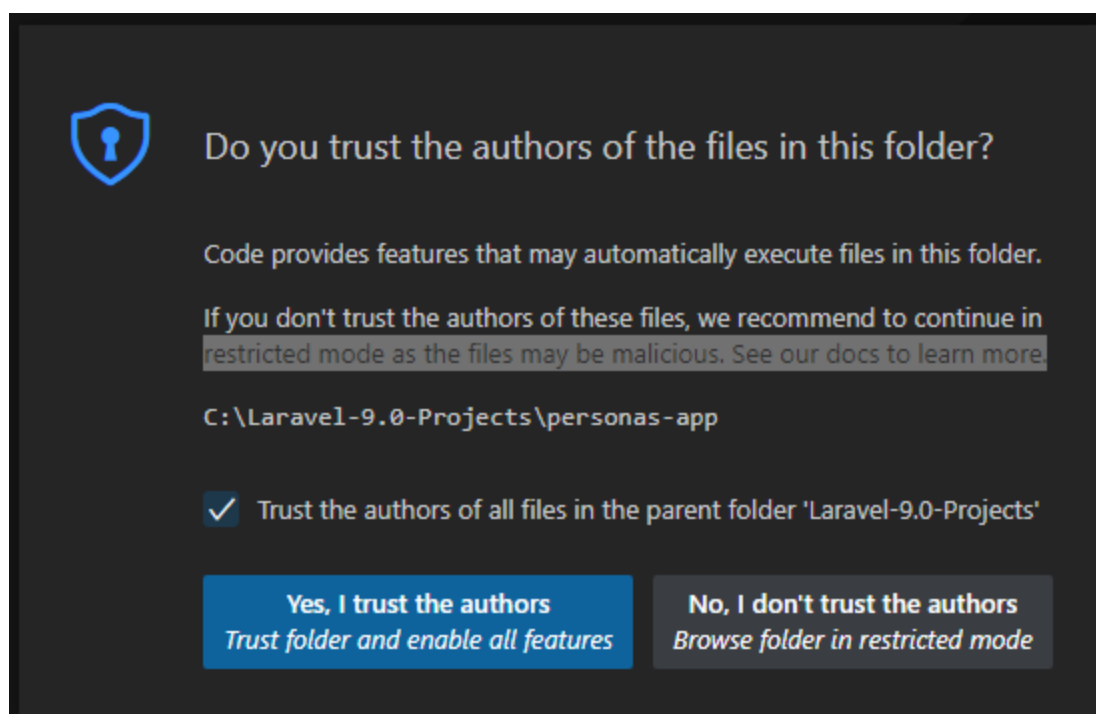
2023-03-20 13:32:47 .....
~ 0s
2023-03-20 13:32:47 /favicon.ico .....
~ 1s
2023-03-20 13:32:47 .....
~ 78s
2023-03-20 13:37:42 .....
~ 1s
2023-03-20 13:37:42 /favicon.ico .....
~ 1s
2023-03-20 13:37:43 .....
~ 60s
^C
C:\Laravel-9.0-Projects\personas-app>
```


- 2.9. Abra el proyecto en Visual Studio Code, digite en la ventana de comandos **code** .



```
C:\Windows\System32\cmd.exe
C:\Laravel-9.0-Projects\personas-app>code .
```

Se abrirá Visual Studio Code con la carpeta del proyecto **personas-app**, si aparece el mensaje **Do you trust the authors of the files in this folder?** seleccione la casilla de verificación y luego presione clic en **Yes, I Trust the authors**.



- 2.10. Al proyecto se le llevará un control de versiones, utilizando la herramienta **Git**. Para verificar si se tiene Git instalado, abra una ventana de comando en Visual Studio Code, presionando las teclas CTRL + SHIFT + Ñ, en la parte inferior de Visual Studio Code aparecerá una ventana de comandos.

Digite el comando **git - - version**, para verificar la versión de git instalada.

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE
PS C:\Laravel-9.0-Projects\personas-app> git --version
git version 2.20.1.windows.1
PS C:\Laravel-9.0-Projects\personas-app> 
```

- 2.11. Se debe configurar el git con la información del usuario, para ello se utilizará el comando git config

git config --global user.name "Mi nombre"

git config --global user.email "usuario@dominio.com"

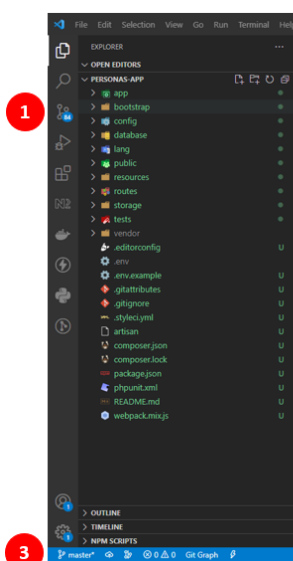
Si desea tener la configuración solo para el proyecto creado, omita el parámetro - **global**, con ello la configuración será local.

```
TERMINAL  PROBLEMS  OUTPUT  GITLENS  DEBUG CONSOLE
PS C:\Laravel-9.0-Projects\personas-app> git config user.name "Luis Romo"
PS C:\Laravel-9.0-Projects\personas-app> git config user.email luis.romo@campuscecep.edu.co 
```

- 2.12. Para comenzar a versionar el proyecto escriba el comando **git init**

```
TERMINAL  PROBLEMS  OUTPUT  GITLENS  DEBUG CONSOLE
PS C:\Laravel-9.0-Projects\personas-app> git init
Initialized empty Git repository in C:/Laravel-9.0-Projects/personas-app/.git/
PS C:\Laravel-9.0-Projects\personas-app> 
```

En Visual Studio Code se muestran varios elementos que indican que el proyecto se está versionando.



1. El icono de control de cambios, muestra la cantidad de archivos que tienen algún cambio desde el último versionamiento.
2. Indicador informado si el archivo está sin versionar (U: **U**ntracked, A: **A**dded, M: **M**odified, D: **D**eleted), se ha agregado para versionar, si se ha modificado
3. Rama actualmente seleccionada, inicialmente un proyecto tiene una rama llamada **master**.

Para agregar archivos al **Staging Area** (lugar donde se guardan temporalmente los cambios, para luego ser llevados definitivamente al repositorio) digite el comando

git add . El punto indica que va agregar todos los archivos.

```

TERMINAL  PROBLEMS  OUTPUT  GITLENS  DEBUG CONSOLE

PS C:\Laravel-9.0-Projects\personas-app> git add .
warning: LF will be replaced by CRLF in .editorconfig.
The file will have its original line endings in your working directory

```

Puede consultar el estado del Staging Area digitando el comando **git status**, mostrando la información correspondiente al estado de los archivos (**Untracked, New, Modified, Delete**)

```

TERMINAL  PROBLEMS  OUTPUT  GITLENS  DEBUG CONSOLE

PS C:\Laravel-9.0-Projects\personas-app> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   .editorconfig
        new file:   .env.example
        new file:   .gitattributes
        new file:   .gitignore
        new file:   .styleci.yml
        new file:   README.md

```

- 2.13. Para crear una versión del proyecto, con los archivos agregados al **Staging Area** se digita el comando **git commit -m "mensaje para identificar la versión"**. Cree la primera versión de su proyecto: **git commit -m "Initial Project"**

```
TERMINAL  PROBLEMS  OUTPUT  GITLENS  DEBUG CONSOLE
new file:  routes/channels.php
PS C:\Laravel-9.0-Projects\personas-app> git commit -m "Initial Project"
[master (root-commit) 73a694b] Initial Project
84 files changed, 11267 insertions(+)
create mode 100644 .editorconfig
create mode 100644 .env.example
create mode 100644 .gitattributes
create mode 100644 .gitignore
```

- 2.14. Puede consultar el historial de versiones con el comando **git log**. El comando muestra la información de cada uno de los commits realizados: fecha, hora, usuario, correo del usuario y rama.

```
TERMINAL  PROBLEMS  OUTPUT  GITLENS  DEBUG CONSOLE
PS C:\Laravel-9.0-Projects\personas-app> git log
commit 223a7a828e01f397bf93f0ecfb823280bdb45573 (HEAD -> master)
Author: Luis Romo <luis.romo@campuscecep.edu.co>
Date:   Mon Mar 20 17:12:32 2023 -0500


    Initial Project
PS C:\Laravel-9.0-Projects\personas-app> █
```

- 2.15. Para proteger el proyecto o compartirlo con otras personas, se debe crear una cuenta en una plataforma que permita gestionar repositorios remotos, tales como: **Github**, **Bitbucket**, **GitLab**, **SourceForge**, entre otras.

Si no tiene una cuenta en GitHub, crea una accediendo al siguiente enlace-
<https://github.com/>

Una vez creada la cuenta, cree un nuevo repositorio llamado **personas-app**


Owner * Repository name *


 yovaromo-cecep / personas-app ✓

Great repository names are short and memorable. Need inspiration? How about [glowing-happiness?](#)

Description (optional)

Primer proyecto en Laravel

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.


☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▼

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

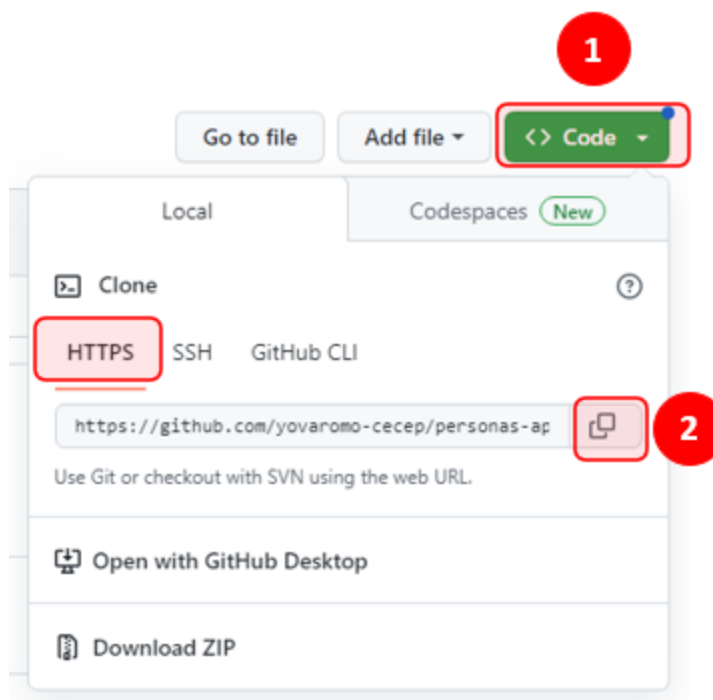
License: None ▼

 You are creating a public repository in your personal account.

Create repository

Ahora que se tiene el repositorio remoto creado, se debe asociar y sincronizar con el repositorio local, utilizando el comando **git remote add origin URL-Proyecto**

Para copiar la dirección del proyecto, presione click en el botón verde <> Code, en la sección HTTPS al frente de la dirección URL presione click en el icono para copiar



Diríjase a la ventana de comandos que tiene abierta en el Visual Studio Code y digite el comando **git remote add origin** y pegue la URL copiada

git remote add origin <https://github.com/yovaromo-cecep/personas-app.git>

```
TERMINAL  PROBLEMS  OUTPUT  GITLENS  DEBUG CONSOLE
PS C:\Laravel-9.0-Projects\personas-app> git remote add origin https://github.com/yovaromo-cecep/personas-app.git
PS C:\Laravel-9.0-Projects\personas-app>
```

Puede consultar los repositorios remotos configurados con el comando **git remote -v**

```
TERMINAL  PROBLEMS  OUTPUT  GITLENS  DEBUG CONSOLE
PS C:\Laravel-9.0-Projects\personas-app> git remote -v
origin https://github.com/yovaromo-cecep/personas-app.git (fetch)
origin https://github.com/yovaromo-cecep/personas-app.git (push)
PS C:\Laravel-9.0-Projects\personas-app>
```

Puede cambiar el nombre de la rama **master** por **main**, puesto que el nombre de la rama principal en GitHub es main, **git branch -M main**

```

TERMINAL  PROBLEMS  OUTPUT  GITLENS  DEBUG CONSOLE

PS C:\Laravel-9.0-Projects\personas-app> git remote -v
origin https://github.com/yovaromo-cecep/personas-app.git (fetch)
origin https://github.com/yovaromo-cecep/personas-app.git (push)
PS C:\Laravel-9.0-Projects\personas-app> git branch -M main
PS C:\Laravel-9.0-Projects\personas-app>

```

Es hora de enviar el contenido del repositorio local al repositorio remoto creado, para ello en la ventana de comando abierta al principio de la práctica digite **git push -u origin main**, si todo va bien, los cambios subirán al repositorio remoto, y el comando lo informará

```

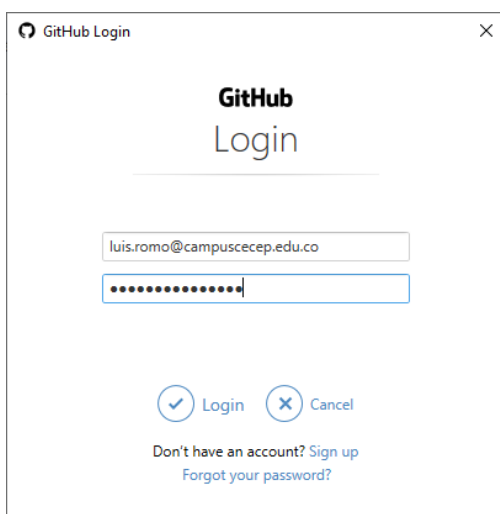
C:\Windows\System32\cmd.exe

C:\Laravel-9.0-Projects\personas-app>git push -u origin main
Enumerating objects: 109, done.
Counting objects: 100% (109/109), done.
Delta compression using up to 4 threads
Compressing objects: 100% (91/91), done.
Writing objects: 100% (109/109), 72.32 KiB | 2.68 MiB/s, done.
Total 109 (delta 7), reused 0 (delta 0)
remote: Resolving deltas: 100% (7/7), done.
To https://github.com/yovaromo-cecep/personas-app.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

C:\Laravel-9.0-Projects\personas-app>

```

Es posible que el comando antes de ejecutar solicite las credenciales de la cuenta de GitHub.

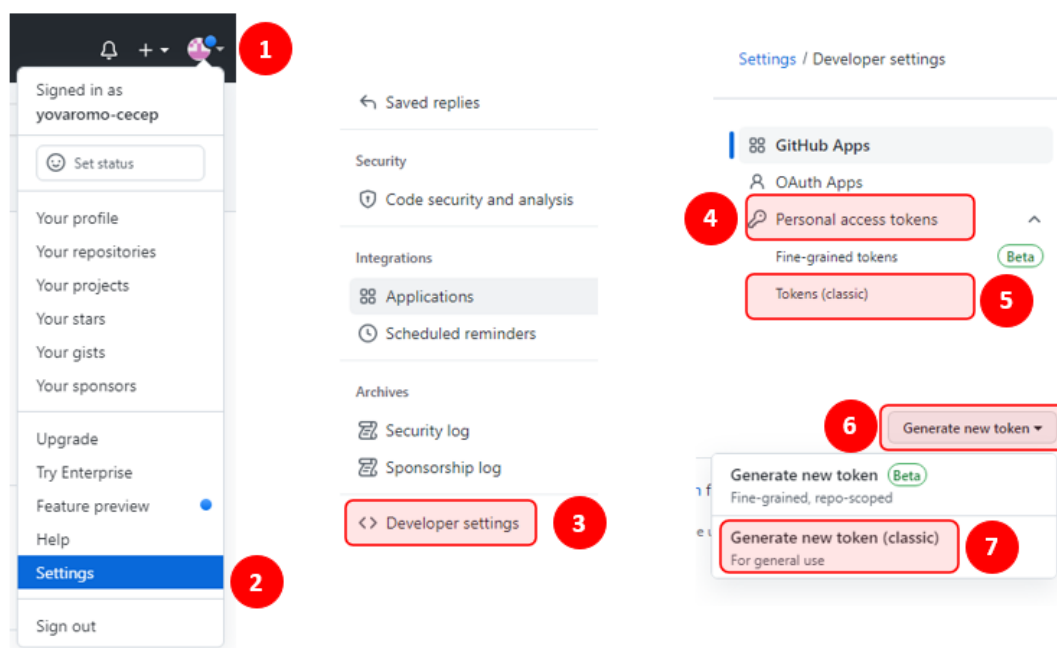


Por seguridad, GitHub para actualizar el repositorio no permite autenticar con las credenciales de la cuenta.

Para ello, ofrece varios mecanismos, entre éstos, crear un **Personal Access Token**.

Ingresa a la cuenta de GitHub y ejecute los siguientes pasos: Clic en el icono de perfil de la cuenta , clic en **Settings**, vaya a la parte inferior izquierda y presione clic en **Developers settings, Persona access**

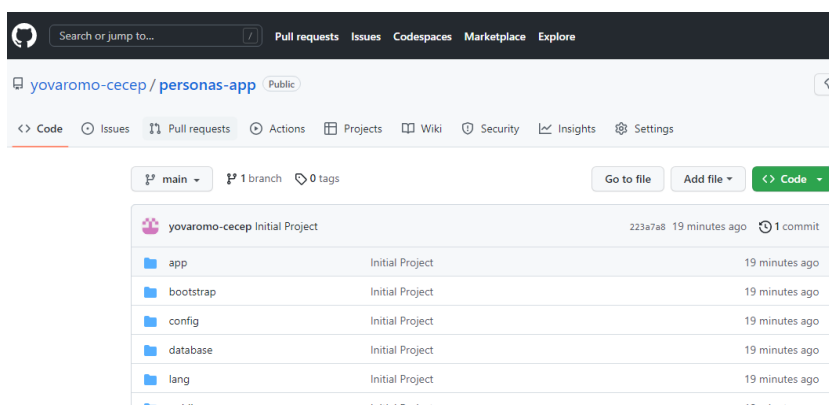
tokens, Tokens (classic), Generate new token, Generate new token (classic)



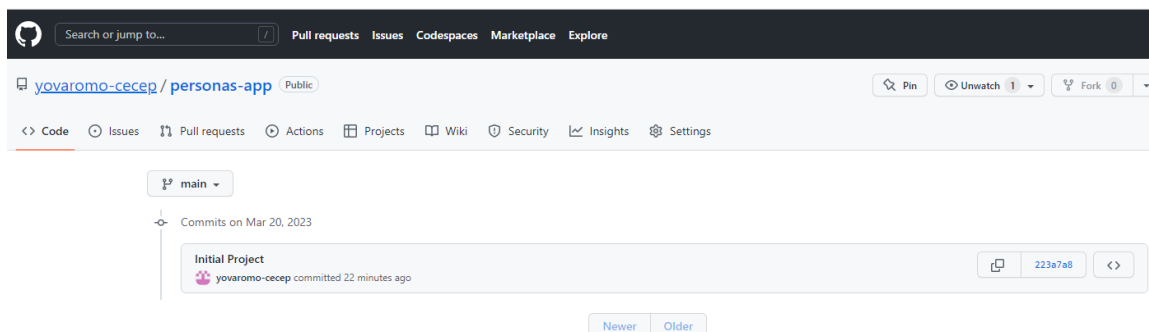
Configure el nombre del Token (note), el tiempo de expiración y los permisos, y finalmente presione click en el botón **Generate Token**. Guarde el password generado, no tendrá forma de volverlo a consultar, lo usará cuando vaya a realizar una actualización al repositorio desde la línea de comandos.

Si no puedo ejecutar el push anterior, intente de nuevo **git push -u origin main**, utilice el token personal creado con su correspondiente password.

Una vez el el push sea exitoso, vaya a GitHub y consulte su repositorio, si aún está en el resultado de creación del proyecto, presione la tecla F5 para actualizar. Podrá ver que se encuentran todos los archivos del repositorio local, y además se puede observar que se ha realizado un commit.

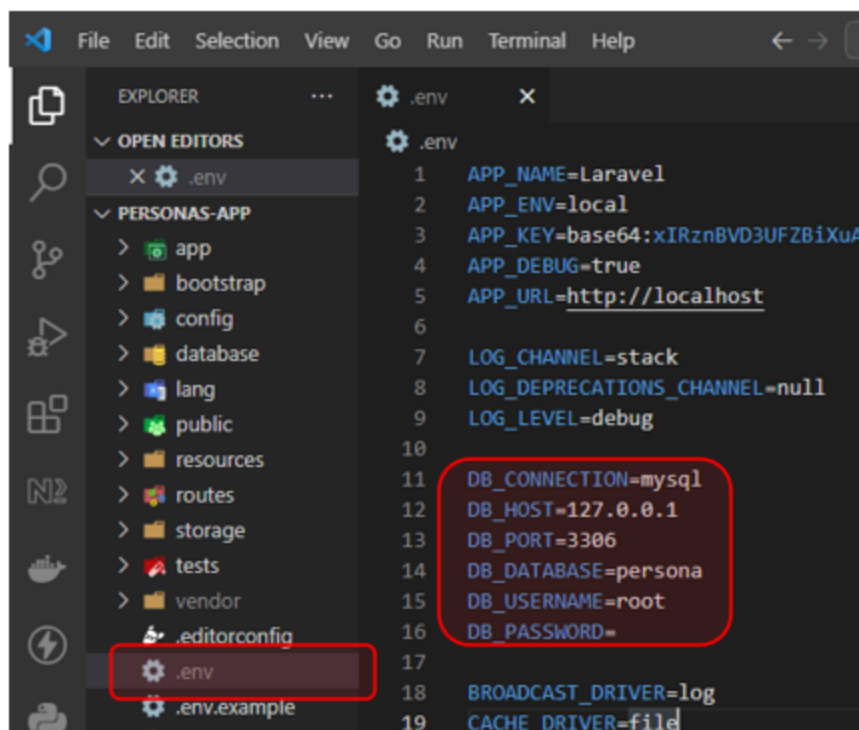


Presione clic en el enlace 1 commit para que observe la información correspondiente a éste.



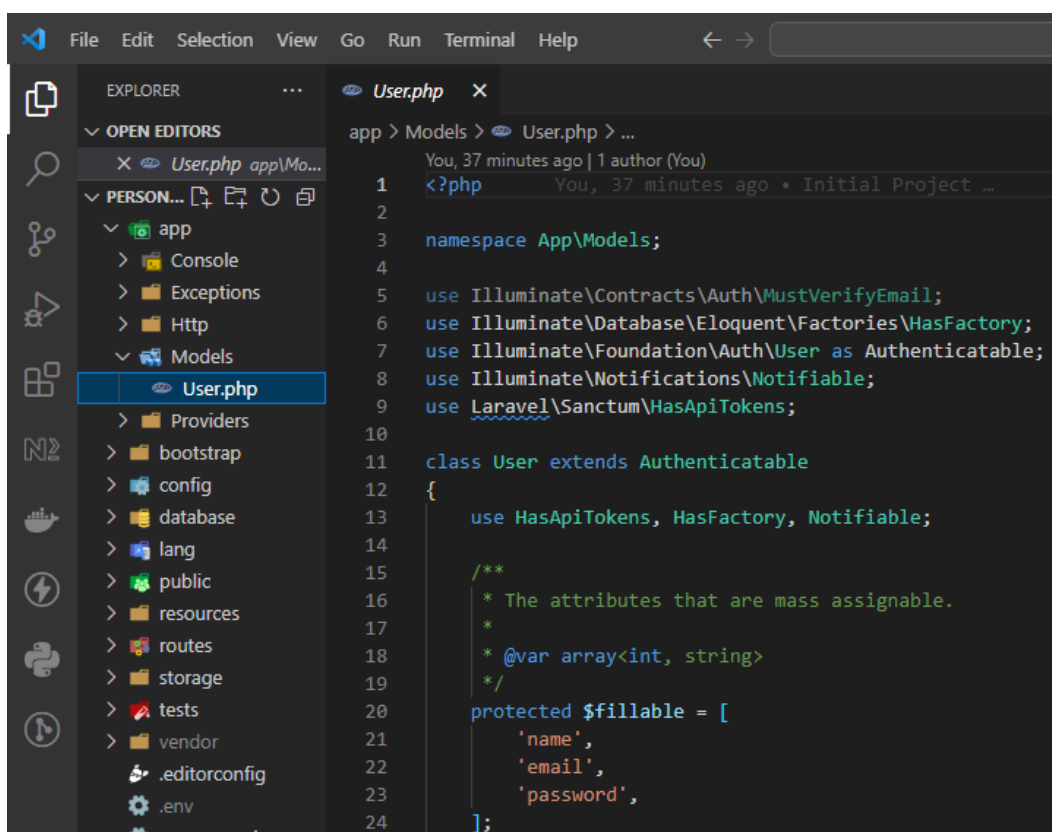
3. Desarrollando el proyecto Personas-app - CRUD para las comunas

- 3.1. **Configurar DB Persona**, el proyecto utilizará la BD persona creada al principio de la práctica. Para configurar la BD en el proyecto se debe abrir el archivo **.env** que se encuentra en la raíz del proyecto. Se debe cambiar la llave de configuración **DB_DATABASE**



3.2. Creando el modelo Comuna, para trabajar con la base de datos se debe crear los Modelos del proyecto, los cuales van a corresponder con las tablas de la base de datos. En un proyecto recién creado Laravel nos ofrece el modelo **User**, que se encuentra en la carpeta **Models** ubicada en la carpeta **app**.

<https://laravel.com/docs/9.x/eloquent#generating-model-classes>



```

1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Contracts\Auth\MustVerifyEmail;
6  use Illuminate\Database\Eloquent\Factories\HasFactory;
7  use Illuminate\Foundation\Auth\User as Authenticatable;
8  use Illuminate\Notifications\Notifiable;
9  use Laravel\Sanctum\HasApiTokens;
10
11 class User extends Authenticatable
12 {
13     use HasApiTokens, HasFactory, Notifiable;
14
15     /**
16      * The attributes that are mass assignable.
17      *
18      * @var array<int, string>
19      */
20     protected $fillable = [
21         'name',
22         'email',
23         'password',
24     ];
  
```

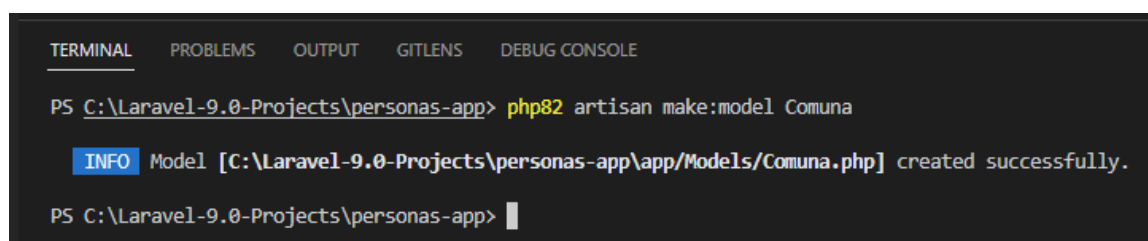
Un modelo en Laravel es una clase que representa una tabla de la base de datos, como nomenclatura **Laravel** espera que el nombre del modelo sea una palabra en singular y que exista una tabla con el mismo nombre en plural. En ese sentido el modelo **User** espera que exista una tabla en la base de datos con el nombre **Users**.

La clase de un modelo hereda de la clase **Model** que ofrecerá un conjunto de métodos para operar los datos, como agregar, consultar, modificar, eliminar entre otros.

Para crear un modelo se ejecuta el comando **php artisan make:model NombreDeModelo**, Laravel creará un archivo con el nombre del modelo y lo

almacenará en la carpeta **Models** que está ubicada en la carpeta **app** del proyecto.

Es hora de crear el modelo para la entidad Comuna, para ello digitar en la ventana de comandos de Visual Studio Code el comando **php artisan make:model Comuna**



```
TERMINAL  PROBLEMS  OUTPUT  GITLENS  DEBUG CONSOLE

PS C:\Laravel-9.0-Projects\personas-app> php82 artisan make:model Comuna

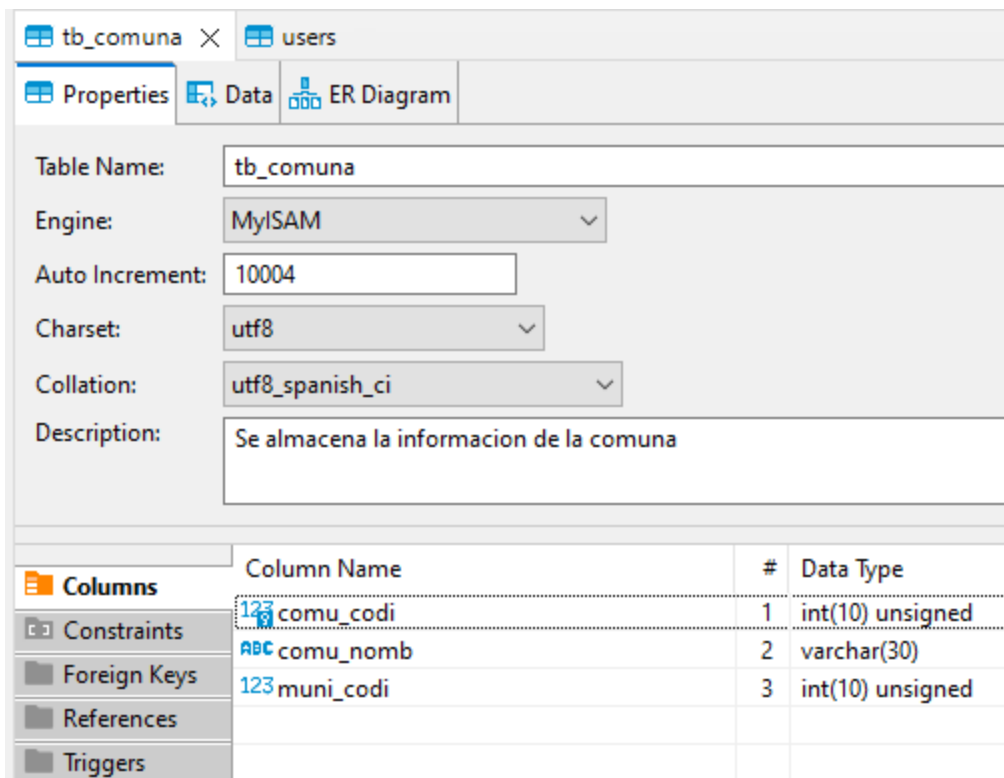
INFO Model [C:\Laravel-9.0-Projects\personas-app\app\Models\Comuna.php] created successfully.

PS C:\Laravel-9.0-Projects\personas-app> 
```

Ahora el proyecto tiene el archivo **Comuna.php** que contiene la clase Comuna que hereda de la clase **Model**. La clase Comuna está lista para gestionar datos.

Laravel también espera que la tabla se llame **Comunas**, que tenga una llave primaria llamada **id** y que tenga dos campos para manejar las fechas creación y actualización de cada registro, llamados **created_at** y **updated_at**.

La tabla para la entidad **Comuna** no cumple con estas características, dado que se llama **tb_comuna**, tiene una llave primaria llamada **comu_codi** y no tiene los campos para las fechas.



Para que el modelo conozca de estas características se le debe informar, haciendo los siguiente cambios a la clase:

```
Comuna.php U X
app > Models > Comuna.php > ...
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Comuna extends Model
9  {
10     use HasFactory;
11     protected $table = 'tb_comuna';
12     protected $primaryKey = 'comu_codi';
13     public $timestamps = false;
14 }
15
```

3.3. Creando la Vista para Comunas.

Las vistas tienen como responsabilidad ofrecer un mecanismo para que el usuario pueda interactuar con la aplicación. Un proyecto Laravel guarda las vistas en la carpeta **views** que se encuentra dentro de la carpeta **resources**.

<https://laravel.com/docs/9.x/views>

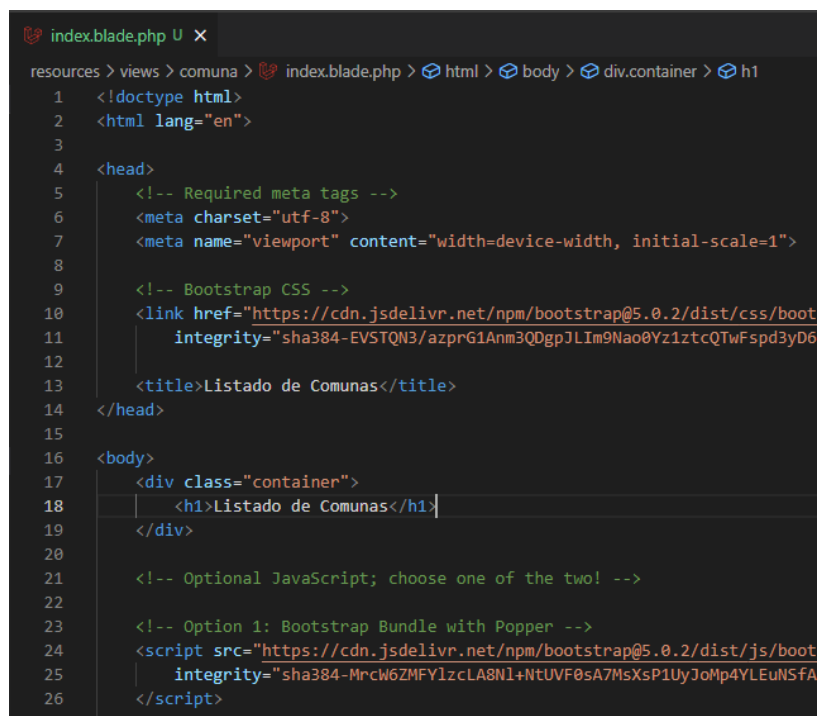
Un proyecto tendrá un conjunto de vistas que se pueden agrupar por algún criterio, para ello se crean carpetas dentro de la carpeta **views**.

Para gestionar las acciones de la entidad Comuna se requiere varias vistas, por esta razón crearemos una carpeta llamada **comuna**, dentro de la carpeta **views**.

Presione clic derecho en el nombre de la carpeta **view** y seleccione **New Folder** y llámela **comuna**.

Ahora cree un archivo dentro de la carpeta comuna y llámelo **index.blade.php**. Clic derecho sobre la carpeta comuna y luego **New File**.

Vaya a la página de bootstrap y copie el código de la sección **Starter Template** <https://getbootstrap.com/docs/5.0/getting-started/introduction/>, peguelo dentro del archivo **index.blade.php** y modifique el contenido como se ve en la imagen.



```

index.blade.php U X
resources > views > comuna > index.blade.php > html > body > div.container > h1
1 <!doctype html>
2 <html lang="en">
3
4 <head>
5 <!-- Required meta tags -->
6 <meta charset="utf-8">
7 <meta name="viewport" content="width=device-width, initial-scale=1">
8
9 <!-- Bootstrap CSS -->
10 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/boot
11     integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfSpd3yD6
12
13 <title>Listado de Comunas</title>
14 </head>
15
16 <body>
17 <div class="container">
18 <h1>Listado de Comunas</h1>
19 </div>
20
21 <!-- Optional JavaScript; choose one of the two! -->
22
23 <!-- Option 1: Bootstrap Bundle with Popper -->
24 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/boot
25     integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfA
26 </script>
27
  
```

En general, las vistas necesitan de los datos almacenados en la base de datos, para la vista index de comuna se necesita los datos de la tabla **tb_comuna** que está asociada al modelo **Comuna**, sin embargo, la **vista** no tiene una conexión directa al **modelo**, por lo que se requiere contar con un **controlador** que gestione las acciones solicitadas por la vista, el controlador le solicita al modelo los datos y finalmente el controlador le pasará a la vista los datos recuperados.

3.4. Creando el controlador para Comuna.

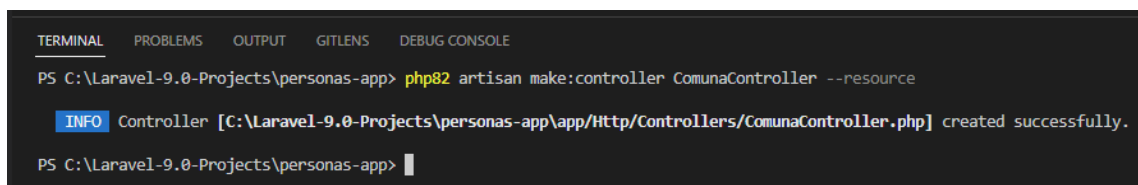
Un controlador es una clase que hereda de la clase **Controller** y tiene como responsabilidad recibir las peticiones realizadas por el usuario desde la vista. El controlador tendrá un conjunto de métodos que se encargaran de ejecutar la lógica de la solicitud. Los métodos estarán asociados a las rutas configuradas en el proyecto. <https://laravel.com/docs/9.x/controllers>

Para crear un controlador se ejecuta el comando **php artisan make:controller NombreController**, este comando creará una clase vacía con el nombre **NombreController** que hereda de la clase **Controller**.

Hay controladores que se encargan de realizar todas las operaciones del CRUD de una entidad, a estos controladores Laravel los denomina Controladores Resources, y se pueden crear automáticamente mediante comando.

Para crear el controlador resource para la entidad comuna se ejecuta el siguiente comando:

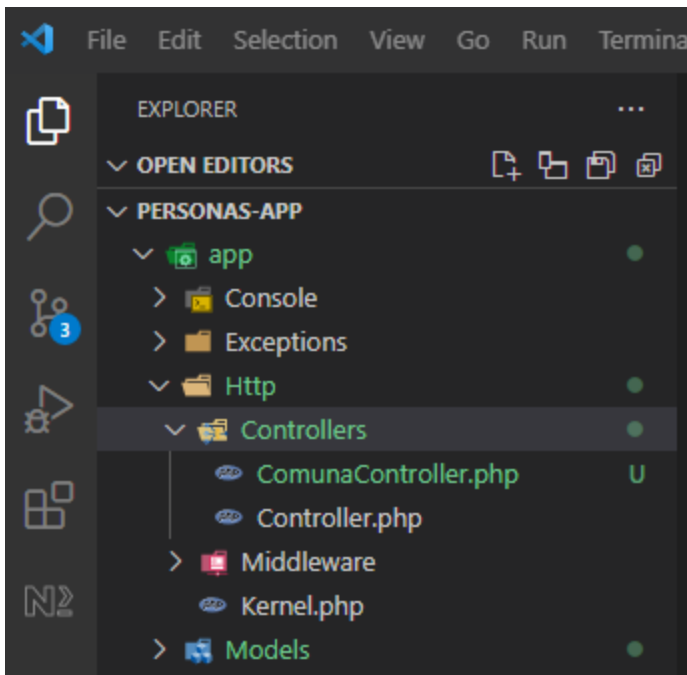
php artisan make:controller ComunaController - - resource



```
TERMINAL  PROBLEMS  OUTPUT  GITLENS  DEBUG CONSOLE
PS C:\Laravel-9.0-Projects\personas-app> php82 artisan make:controller ComunaController --resource

INFO Controller [C:\Laravel-9.0-Projects\personas-app\app\Http\Controllers\ComunaController.php] created successfully.
PS C:\Laravel-9.0-Projects\personas-app> |
```

El comando creará un archivo llamado **ComunaController.php** que se almacena en la ruta **app/Http/Controllers**



El archivo contiene una clase llamada **ComunaController** que hereda de la clase **Controller** y como fue creado de tipo resource tiene 7 métodos: **index, create, store, show, edit, update y destroy**, cada uno para programar la lógica de las solicitudes realizadas.

index: Consulta el modelo y devuelve el listado de registros almacenados en la tabla.

create: Llama una vista que tendrá un formulario con el propósito de capturar los datos para la creación de un registro.

store: Recibe los datos enviados por el formulario para crear un registro y los almacena en la base de datos.

show: Muestra la información de un registro en específico.

edit: Llama una vista que tendrá un formulario con el propósito de editar los datos de un registro, el controlador le deberá enviar los datos actuales a la vista.

update: Recibe los datos enviados por el formulario para editar un registro y los actualiza en la base de datos.

destroy: Elimina un registro específico de la base de datos.

Laravel utiliza **Eloquent**, un ORM (object-relational mapper) que permite realizar las operaciones con la base de datos, generalmente Eloquent se utiliza en mayor proporción en los controladores. <https://laravel.com/docs/9.x/eloquent>

3.5. Creando la lógica para recupera los datos del modelo y enviar a la vista

Abrir el archivo **ComunaController.php** y modificar el método index, como se observa en la siguiente imagen:

```

5  use App\Models\Comuna;
6  use Illuminate\Http\Request;
7
8  class ComunaController extends Controller
9  {
10     /**
11      * Display a listing of the resource.
12      *
13      * @return \Illuminate\Http\Response
14      */
15     public function index()
16     {
17         $comunas = Comuna::all();
18         return view("comunas.index", ["comunas" => $comunas]);
19     }

```

Verificar que se haya importado el modelo Comuna, línea 5 de la imagen, de lo contrario digitarla.

El controlador utiliza **Eloquent** para recuperar todos los registros de la tabla **tb_comuna**, mediante el modelo **Comuna** (línea 17 de la imagen) y las almacena en la variable **\$comunas**. Luego el controlador llama a la vista index que se encuentra en la carpeta comuna y le pasa los datos guardados en la variable **\$comunas** (línea 18)

3.6. Gestionados los datos enviados en la vista

Ajustar la vista del listado de comunas (**index.blade.php**) para que en un elemento table despliegue la información enviada. Para ello vaya a la página de bootstrap y copie el código HTML para construir una tabla, <https://getbootstrap.com/docs/5.0/content/tables/> el primer ejemplo que está

en la sección **Overview**. Pegue el código en el archivo index.php y borre los dos últimos bloques de las etiquetas <tr>, como se observa en la imagen.

```

16 <body>
17   <div class="container">
18     <h1>Listado de Comunas</h1>
19     <table class="table">
20       <thead>
21         <tr>
22           <th scope="col">#</th>
23           <th scope="col">First</th>
24           <th scope="col">Last</th>
25           <th scope="col">Handle</th>
26         </tr>
27       </thead>
28       <tbody>
29         <tr>
30           <th scope="row">1</th>
31           <td>Mark</td>
32           <td>Otto</td>
33           <td>@mdo</td>
34         </tr>
35       </tbody>
36     </table>
37   </div>
38

```

Ahora realice el siguiente ajuste para que en la tabla HTML se observe los registros de la consulta enviada desde el método index de ComunaController , como se observa en la siguiente imagen

```

16 <body>
17   <div class="container"> You, 3 hours ago • Comuna List ...
18     <h1>Commune List</h1>
19     <a href="{{ route('comunas.create') }}" class="btn btn-success">Add</a>
20     <table class="table">
21       <thead>
22         <tr>
23           <th scope="col">Code</th>
24           <th scope="col">Commune</th>
25           <th scope="col">Municipality</th>
26           <th scope="col">Actions</th>
27         </tr>
28       </thead>
29       <tbody>
30         @foreach ($comunas as $comuna)
31           <tr>
32             <th scope="row">{{ $comuna->comu_codi }}</th>
33             <td>{{ $comuna->comu_nomb }}</td>
34             <td>{{ $comuna->muni_nomb }}</td>
35             <td><span> Actions </span></td>
36           </tr>
37         @endforeach
38       </tbody>
39     </table>
40   </div>

```

Laravel utiliza el motor de plantillas Blade, que permite incorporar código PHP en las vistas utilizando diferentes directivas. <https://laravel.com/docs/9.x/blade>

En la línea 29 de la imagen, se utiliza la directiva de blade **@foreach** que permite iterar arreglos. El controlador envió los datos en el arreglo **\$comunas** y la directiva **@foreach** la está iterando, cada fila del array se pasa a la variable **\$comuna**. Por tanto, se generarán tantas etiquetas `<tr>` como registros haya en el arreglo **\$comunas**. Para imprimir el valor de una variable se encierra entre dobles `{{ $variable }}`

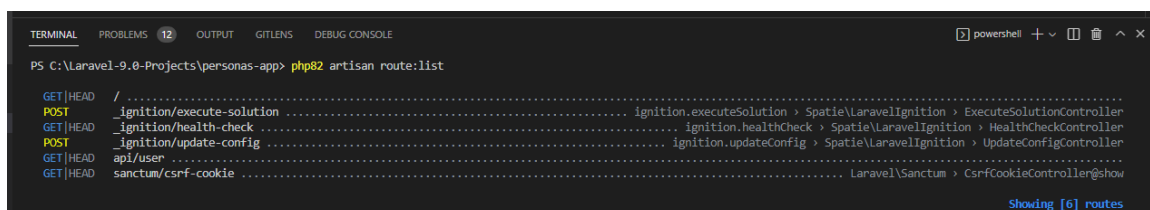
3.7. Configurando rutas

Para acceder a los recursos del proyecto en Laravel se debe configurar las rutas de acceso. Hay varios métodos para acceder a los recuerdos de Laravel, uno de ellos es mediante el navegador web. En Laravel para configurar las rutas web se hacen en el archivo **web.php** que se encuentra en la carpeta **resource**.

<https://laravel.com/docs/9.x/routing>

Cada ruta deberá tener asociado el método de solicitud (get, post put, patch, delete, options) y deberá asociar la acción a realizar: mostrar una vista, realizar una redirección, llamar un método de un controlador.

En la línea de comandos se puede consultar todas las rutas configuradas, mediante el comando **php artisan route:list**



```

PS C:\Laravel-9.0-Projects\personas-app> php82 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 ..... Laravel\Sanctum > CsrfCookieController@show

Showing [6] routes

```

Configuremos una ruta web que acepte la ruta **/comunas** mediante una petición web de tipo **get** y que invoque el método **index** del controlador **ComunaController**

```
ComunaController.php U index.blade.php U web.php 2, M X
routes > web.php > ...
You, 1 minute ago | 1 author (You)
1 <?php
2
3 use App\Http\Controllers\ComunaController;
4 use Illuminate\Support\Facades\Route;
5
6 /*
7 |-----
8 | Web Routes
9 |-----
10 |
11 | Here is where you can register web routes for your application. These
12 | routes are loaded by the RouteServiceProvider within a group which
13 | contains the "web" middleware group. Now create something great!
14 |
15 */
16 You, 3 hours ago • Initial Project ...
17 Route::get('/', function () {
18     return view('welcome');
19 });
20
21 Route::get('/comunas', [ComunaController::class, 'index'])
22
23
```

Verifique que se haya importado el controlador al archivo **web.php**, línea 3 de la imagen, de lo contrario debe digitarla.

Si consulta nuevamente las rutas por la línea de comando, observará la nueva ruta, **php artisan route:list**

```

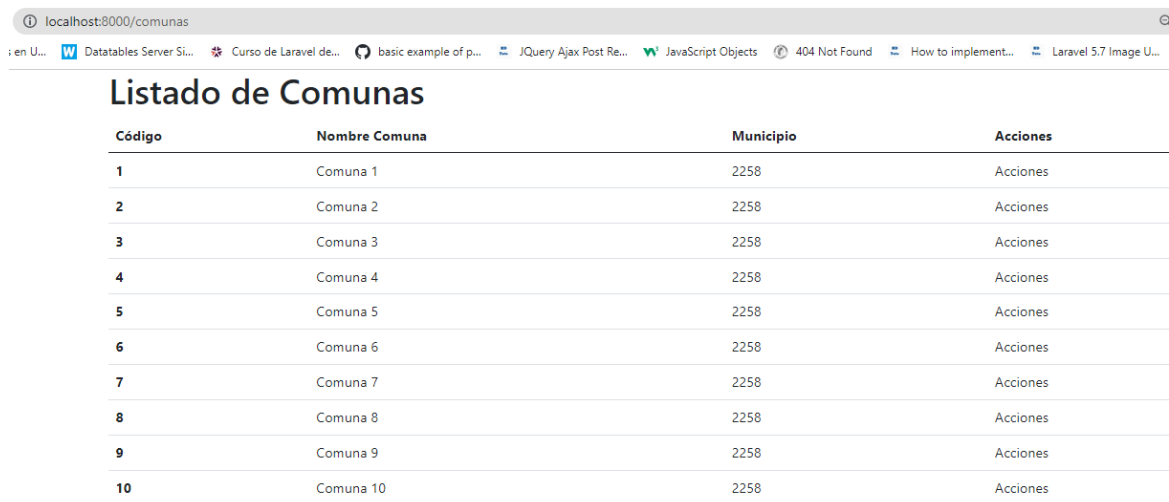
TERMINAL  PROBLEMS 12  OUTPUT  GITLENS  DEBUG CONSOLE
POST      _ignition/execute-solution ..... ignition.executeSolution > Spatie\LaravelIgnition > ExecuteSolutionController
GET /HEAD _ignition/health-check ..... Ignition.healthCheck > Spatie\LaravelIgnition > HealthCheckController
POST      _ignition/update-config ..... ignition.updateConfig > Spatie\LaravelIgnition > UpdateConfigController
GET /HEAD api/user ..... ComunaController@index
GET /HEAD comunas ..... ComunaController@index
GET /HEAD sanctum/csrf-cookie ..... Laravel\Sanctum > CsrfCookieController@show

Showing [7] routes

```

Es momento de ejecutar el proyecto y probar las nuevas funcionalidades, levante el proyecto con el comando **php artisan serve**

Abra un navegador y digite la dirección **http://localhost:8000/comunas**



Código	Nombre Comuna	Municipio	Acciones
1	Comuna 1	2258	Acciones
2	Comuna 2	2258	Acciones
3	Comuna 3	2258	Acciones
4	Comuna 4	2258	Acciones
5	Comuna 5	2258	Acciones
6	Comuna 6	2258	Acciones
7	Comuna 7	2258	Acciones
8	Comuna 8	2258	Acciones
9	Comuna 9	2258	Acciones
10	Comuna 10	2258	Acciones

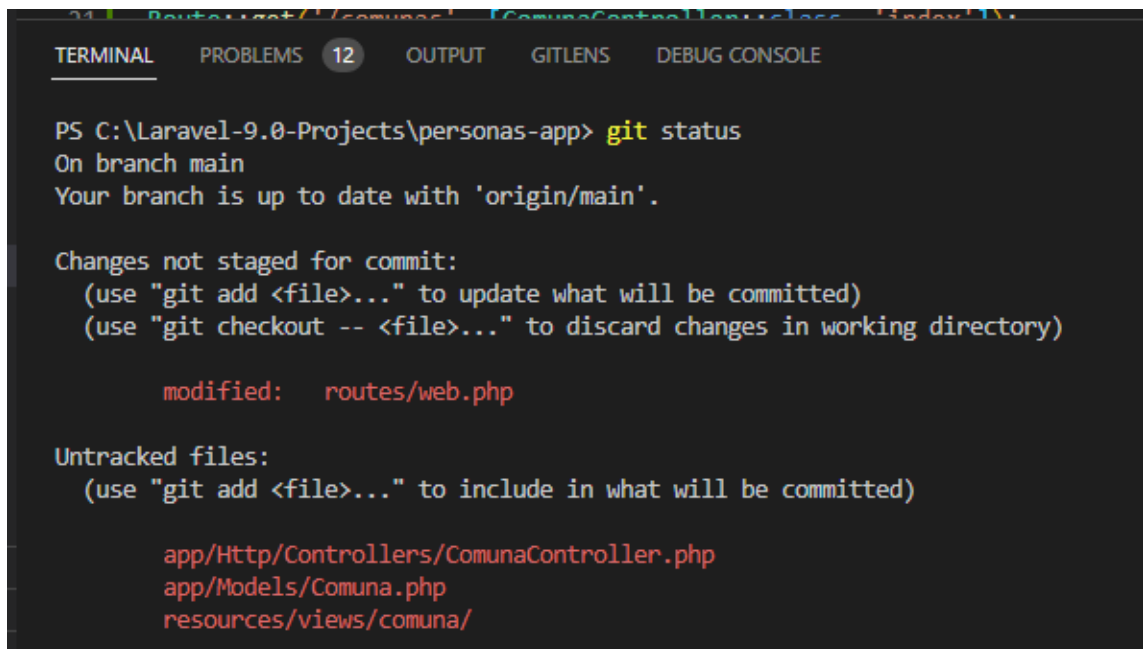
Como se puede observar, la aplicación ha respondido con la vista **index.blade.php** de comunas, con los datos almacenados en la tabla **tb_comuna**.

El flujo de la ejecución es:

- Se llama la ruta **/comunas**, la ruta está configurada en el archivo **web.php** para recibir una petición de tipo **get**.
- La ruta **/comunas** está configurada para llamar el método **index** del controlador **ComunaController**.
- El método **index** usa el modelo de comuna para recuperar todos los registros [**Comuna::all()**] utilizando **Eloquent**.
- El controlador mediante el método **index** llama la vista **comuna.index** pasando un arreglo llamado **\$comunas**
- La vista **index** de comunas, despliega la información

3.8. Generando una nueva versión del proyecto

Hasta el momento se han realizado varios cambios en los archivos del proyecto, se pueden consultar mediante el comando **git status** en la línea de comandos.



```

PS C:\Laravel-9.0-Projects\personas-app> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   routes/web.php

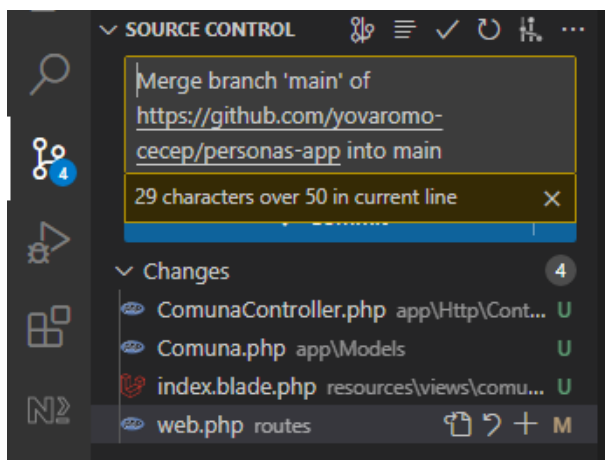
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        app/Http/Controllers/ComunaController.php
        app/Models/Comuna.php
        resources/views/comuna/

```

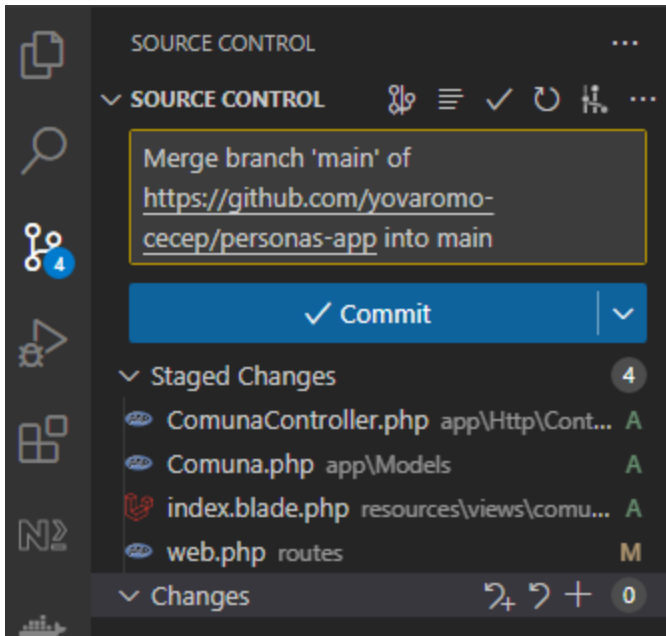
Se ha modificado el archivo **web.php** y se han agregado los archivos **ComunaController.php**, **Comuna.php** y la carpeta **Comuna**, estos últimos aún sin incluir en una próxima versión.

Visual Studio Code también muestra los cambios realizados en el proyecto, en el icono de source control, para nuestro proyecto informa que 4 archivos han cambiado, dando clic en el icono se puede observar los archivos que han cambiado con respecto de la última versión.



En la sección **changes** lista los archivos que han cambiado, la letra al final del nombre del archivo indica el tipo de cambio (U: untracked, M: Modified)

Procedamos a realizar el proceso para generar una nueva versión, la primera acción es incluir al Staging Area los nuevos archivos de la versión, para ello en la ventana de comandos digitar **git add** .



Si consultamos de nuevo el Source Control de Visual Studio Code vemos que los archivos ahora están en el Staged Changes, listos para una nueva versión.

Esta información también la podemos ver con el comando `git status` desde la línea de comandos.

Se puede observar que es la misma información del Control Source.

```

TERMINAL  PROBLEMS  12  OUTPUT  GITLENS  DEBUG CONSOLE

PS C:\Laravel-9.0-Projects\personas-app> git status
On branch main
Your branch is up to date with 'origin/main'.

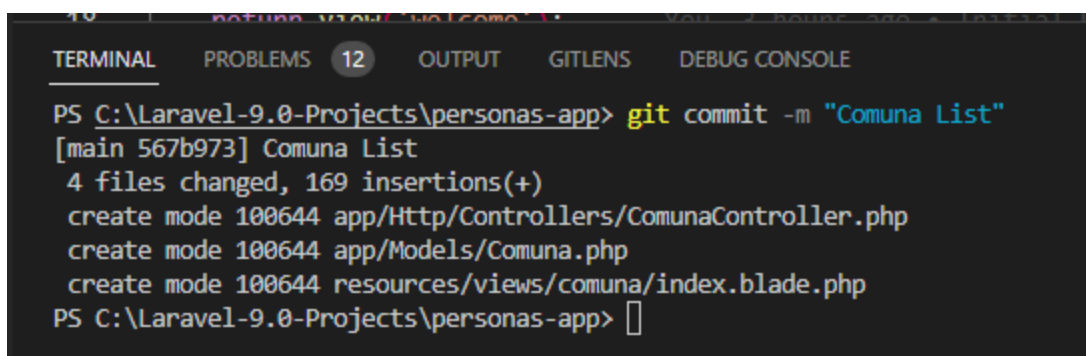
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   app/Http/Controllers/ComunaController.php
    new file:   app/Models/Comuna.php
    new file:   resources/views/comuna/index.blade.php
    modified:   routes/web.php

PS C:\Laravel-9.0-Projects\personas-app>

```

Para generar la versión definitiva realizamos el commit, con el siguiente comando: **git commit -m "Comuna List"**

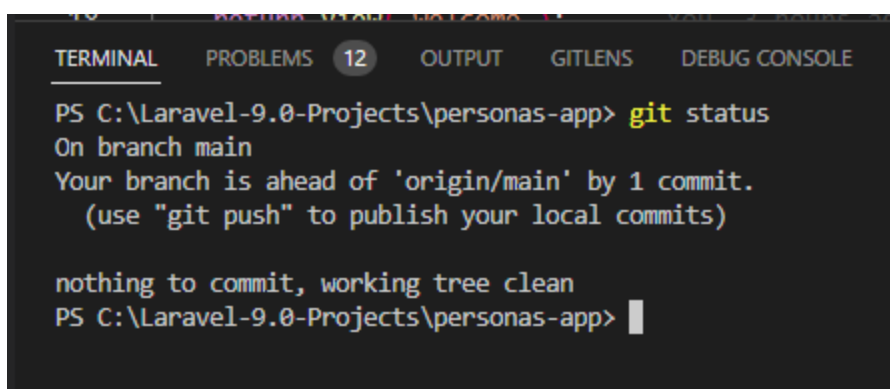


```

TERMINAL  PROBLEMS  12  OUTPUT  GITLENS  DEBUG CONSOLE
PS C:\Laravel-9.0-Projects\personas-app> git commit -m "Comuna List"
[main 567b973] Comuna List
4 files changed, 169 insertions(+)
create mode 100644 app/Http/Controllers/ComunaController.php
create mode 100644 app/Models/Comuna.php
create mode 100644 resources/views/comuna/index.blade.php
PS C:\Laravel-9.0-Projects\personas-app>

```

Git nos informa acerca de lo modificado en el nuevo versionamiento, puede observar que en Source Control de Visual Studio Code ya no aparecen los archivos en la zona de **Staged Changes**, tambien se puede consultar con el comando **git status**



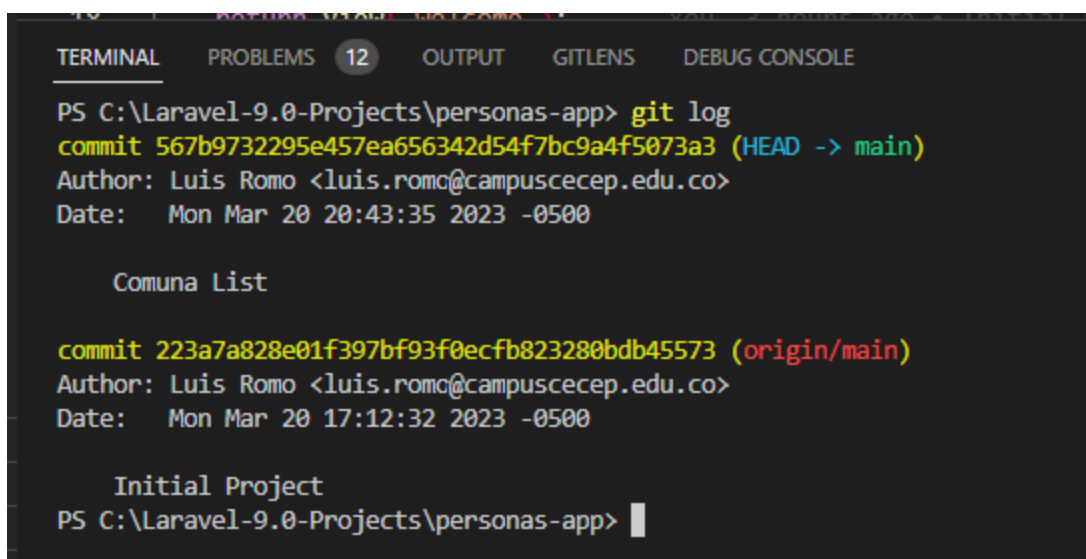
```

TERMINAL  PROBLEMS  12  OUTPUT  GITLENS  DEBUG CONSOLE
PS C:\Laravel-9.0-Projects\personas-app> git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
PS C:\Laravel-9.0-Projects\personas-app>

```

También podemos consultar el historial de versionamiento (commits) mediante el comando **git log**



```

TERMINAL  PROBLEMS  12  OUTPUT  GITLENS  DEBUG CONSOLE
PS C:\Laravel-9.0-Projects\personas-app> git log
commit 567b9732295e457ea656342d54f7bc9a4f5073a3 (HEAD -> main)
Author: Luis Romo <luis.romo@campuscecep.edu.co>
Date: Mon Mar 20 20:43:35 2023 -0500

    Comuna List

commit 223a7a828e01f397bf93f0ecfb823280bdb45573 (origin/main)
Author: Luis Romo <luis.romo@campuscecep.edu.co>
Date: Mon Mar 20 17:12:32 2023 -0500

    Initial Project
PS C:\Laravel-9.0-Projects\personas-app>

```

Se puede observar que hay dos versiones del proyecto (commits), muestra la hora, la fecha y el usuario que la realizó. También muestra que la rama **main** del repositorio **local** ahora es diferente que la rama **main** del **repositorio remoto** (origin).

Es hora de enviar los cambios al repositorio remoto, mediante el comando **git push**.

```
TERMINAL  PROBLEMS  12  OUTPUT  GITLENS  DEBUG CONSOLE

PS C:\Laravel-9.0-Projects\personas-app> git push origin
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 4 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (14/14), 2.58 KiB | 882.00 KiB/s, done.
Total 14 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/yovaromo-cecep/personas-app.git
   223a7a8..567b973  main -> main
PS C:\Laravel-9.0-Projects\personas-app>
```

El comando git informa acerca de la actualización de repositorio remoto. Al consultar de nuevo el historial de commits se observa que los dos repositorios (loca/remoto[origin]) están en la misma versión.

```
TERMINAL  PROBLEMS  12  OUTPUT  GITLENS  DEBUG CONSOLE

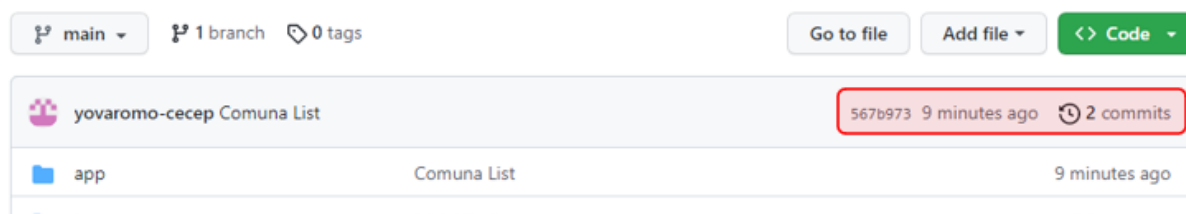
PS C:\Laravel-9.0-Projects\personas-app> git log
commit 567b9732295e457ea656342d54f7bc9a4f5073a3 (HEAD -> main, origin/main)
Author: Luis Romo <luis.romo@campuscecep.edu.co>
Date:   Mon Mar 20 20:43:35 2023 -0500

    Comuna List

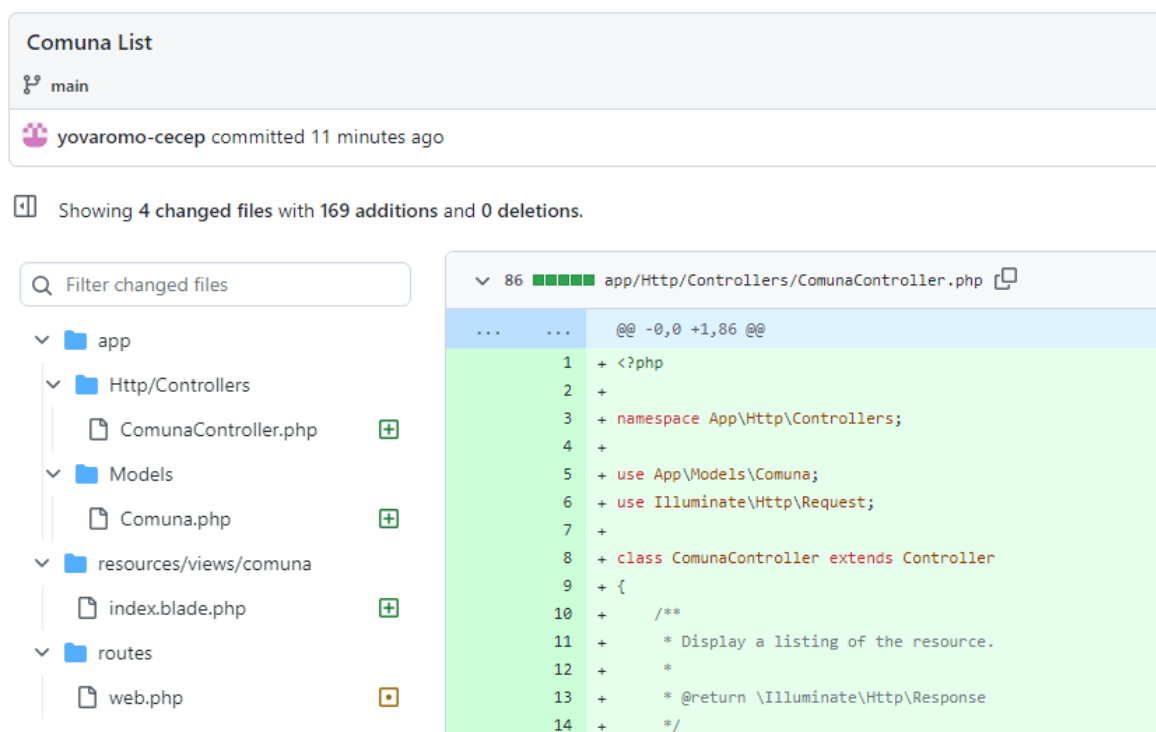
commit 223a7a828e01f397bf93f0ecfb823280bdb45573
Author: Luis Romo <luis.romo@campuscecep.edu.co>
Date:   Mon Mar 20 17:12:32 2023 -0500

    Initial Project
PS C:\Laravel-9.0-Projects\personas-app>
```


También puede consultar en **GitHub** qué archivos modificados y adicionados están en el último commit. Observe que cada cambio tiene la observación que se puso en el comando commit como mensaje, en **-m "mensaje"**



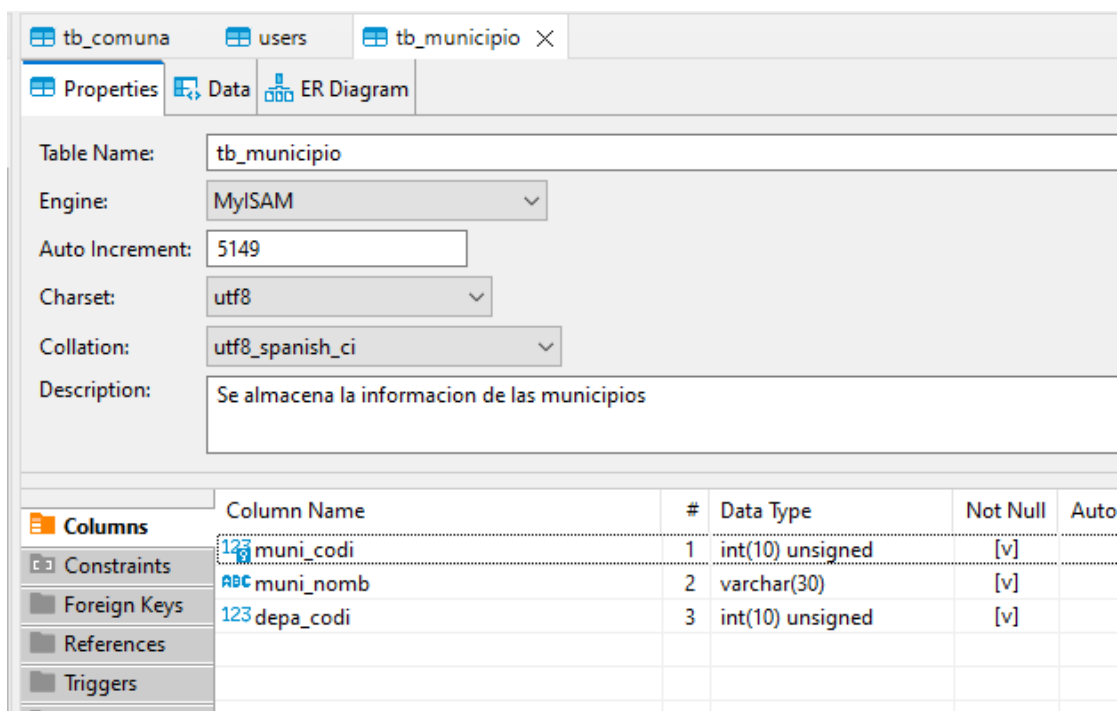
Al presionar clic en el enlace commits podrá observar el historial de commits, si presiona clic en uno de los commits podrá ver los cambios que ocurrieron en éste, y los cambios realizados en cada archivo.



3.9. Ajustando la vista del listado de Comunas

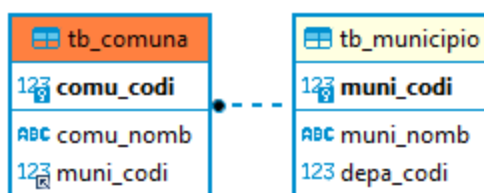
Como puede observar, el listado muestra el código del municipio, se debería mostrar el nombre del municipio. Sin embargo, en la tabla **tb_comuna** sólo tiene el código del municipio que corresponde a la llave foránea, el nombre del municipio se encuentra en la tabla **tb_municipio**.

La tabla **tb_municipio** tiene la estructura que se observa en la imagen



Column Name	#	Data Type	Not Null	Auto
muni_codi	1	int(10) unsigned	[v]	
muni_nomb	2	varchar(30)	[v]	
depa_codi	3	int(10) unsigned	[v]	

Por tanto, para obtener el nombre del municipio se deberá realizar una consulta que relacione las tablas **tb_comuna** y **tb_municipio**, dado que la lógica del negocio así lo determina.



Esto exige que se deba cambiar la forma como se consulta la información de las comunas en el método index del controlador comuna.

Laravel además de Eloquent nos brinda Query Builder para realizar consultas, podríamos ajustar la consulta mediante Eloquent o por medio de Query Builder.

Si se efectúa con Eloquent, se debe configurar las relaciones entre las tablas de acuerdo al tipo de relación (1:1, 1:M, M:M)

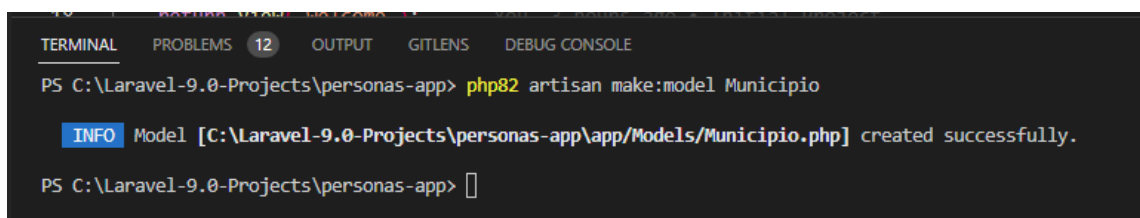
<https://laravel.com/docs/10.x/eloquent-relationships>

Para nuestro caso, la relación entre las tablas `tb_comuna` y `tb_municipio` es de 1 a muchos, un municipio tiene muchas comunas, o varias comunas pertenecen a un municipio. Esta asociación se debería configurar en los modelos de Laravel.

Por el momento, vamos a resolver el requerimiento por medio de Query Builder <https://laravel.com/docs/10.x/queries> que nos permite sostener las relaciones por medio de comandos, **joins**.

Para modificar el método `index` del controlador **ComunaController**, se debe crear primero el modelo de **Municipio**.

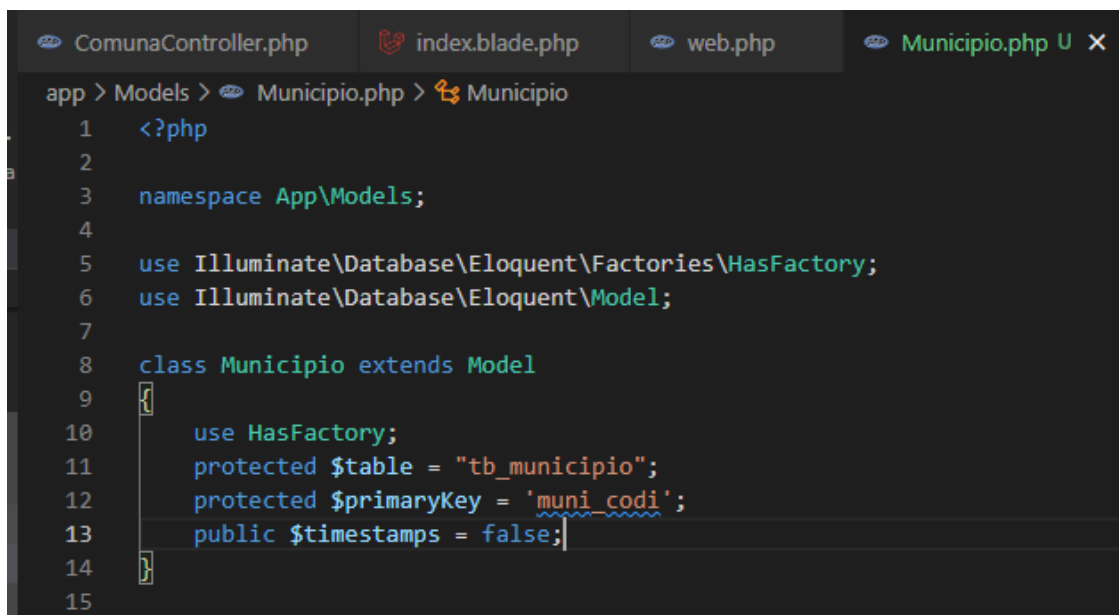
Para crear el modelo **municipio** utilizamos el comando, **php artisan make:model Municipio**



```
TERMINAL  PROBLEMS 12  OUTPUT  GITLENS  DEBUG CONSOLE
PS C:\Laravel-9.0-Projects\personas-app> php82 artisan make:model Municipio

INFO Model [C:\Laravel-9.0-Projects\personas-app\app\Models\Municipio.php] created successfully.
PS C:\Laravel-9.0-Projects\personas-app> 
```

Se debe ajustar el modelo con las características de la tabla `tb_municipio`, tal como se muestra en la siguiente imagen:



```
ComunaController.php  index.blade.php  web.php  Municipio.php U X
app > Models > Municipio.php > Municipio
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Municipio extends Model
9  {
10     use HasFactory;
11     protected $table = "tb_municipio";
12     protected $primaryKey = 'muni_codi';
13     public $timestamps = false;
14 }
15
```

Ahora se puede modificar el método **index** del controlador **ComunaController**, utilizando **Query Builder** para recuperar los datos relacionando las tablas **tb_comuna** y **tb_municipio**, como se observa en la siguiente imagen:

```

12      /**
13       * Display a listing of the resource.
14       *
15       * @return \Illuminate\Http\Response
16       */
17      public function index()
18      {
19          //
20          //$comunas = Comuna::all();
21          $comunas = DB::table('tb_comuna')
22              ->join('tb_municipio', 'tb_comuna.muni_codi', '=', 'tb_municipio.muni_codi')
23              ->select('tb_comuna.*', "tb_municipio.muni_nomb")
24              ->get();
25          return view('comuna.index', ['comunas' => $comunas]);
26      }
27

```

Verifique que al inicio de la clase se haya importado el Facade DB, sino lo deberá digitar, línea 7 de la siguiente imagen.

```

ComunaController.php ...\api  ComunaController.php ...\Controllers
app > Http > Controllers > ComunaController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\Comuna;
6  use Illuminate\Http\Request;
7  use Illuminate\Support\Facades\DB;
8
9  class ComunaController extends Controller
10 {
11     /**

```

Como se observa en la imagen, Query Builder ofrece un conjunto de opciones (métodos) para realizar la consulta, parecidas a las cláusulas que se utilizan en el lenguaje SQL.

Ahora se debe ajustar la vista en el dato que muestra el nombre del municipio en lugar del código del municipio.

```

28 <tbody>
29   @foreach ($comunas as $comuna)
30     <tr>
31       <th scope="row">{{ $comuna->comu_codi }}</th>
32       <td>{{ $comuna->comu_nomb }}</td>
33       <td>{{ $comuna->muni_nomb }}</td>
34       <td><span> Acciones </span></td>
35     </tr>
36   @endforeach
37 </tbody>

```

Si vuelve al navegador y recarga la página del proyecto en la ruta /comunas, puede observar que ahora se muestra el nombre del municipio en lugar del código del municipio.

localhost:8000/comunas

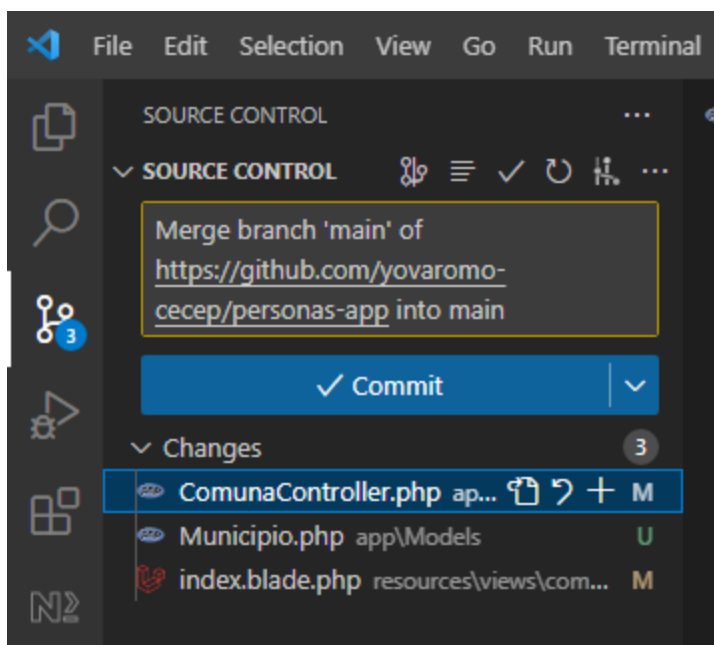
Listado de Comunas

Código	Nombre Comuna	Municipio	Acciones
1	Comuna 1	Cali	Acciones
2	Comuna 2	Cali	Acciones
3	Comuna 3	Cali	Acciones
4	Comuna 4	Cali	Acciones
5	Comuna 5	Cali	Acciones
6	Comuna 6	Cali	Acciones
7	Comuna 7	Cali	Acciones

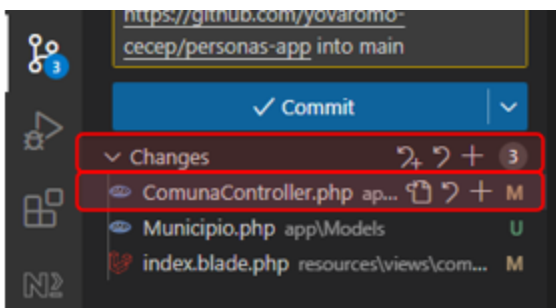
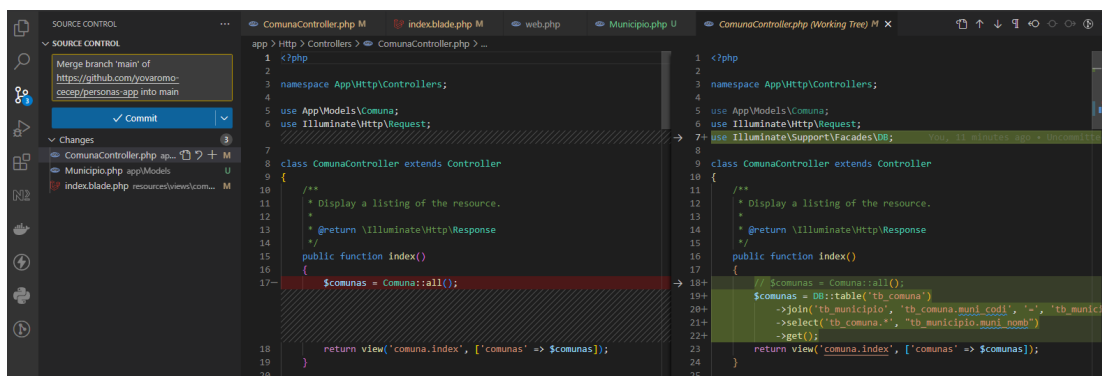
3.10. Generando una nueva versión del proyecto

Se han realizado cambios importantes, es hora de crear una nueva versión del proyecto, recuerden: agregar archivos al Staging Area y luego realizar el commit. Están en Visual Studio Code, estas acciones las podemos realizar por la interfaz gráfica.

De clic en el icono de Source Control, allí mostrará los archivos modificados, son tres: **ComunaController.php** (M), **Municipio.php** (U) e **index.blade.php** (M).



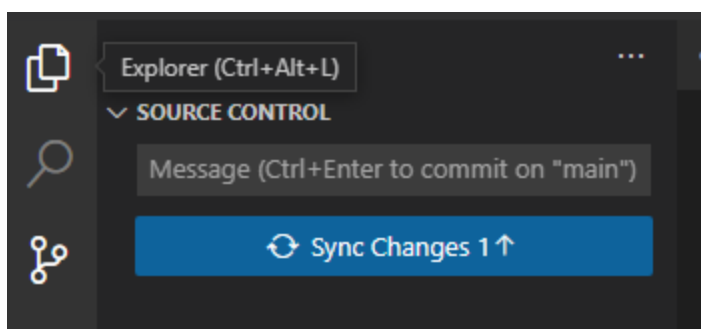
Si presiona click en el nombre del archivo, se abre una ventana en donde se evidencia los cambios efectuados al archivo, por ejemplo, en la siguiente imagen se puede observar los cambios realizados al archivo ComunaController.php.



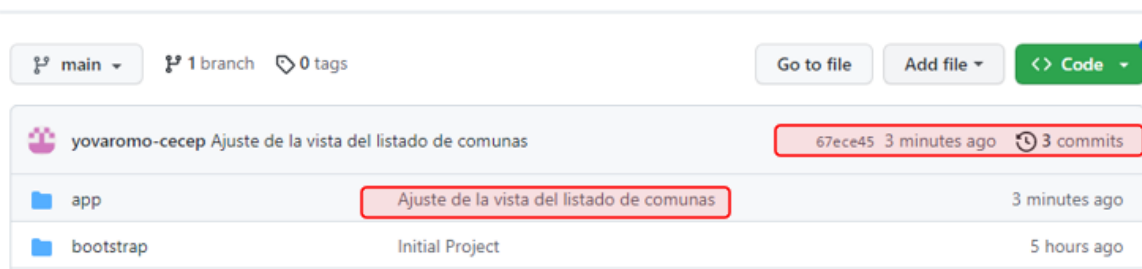
Para agregar archivo por archivo al Staged Area, se presiona clic en el Icono + que aparece al final de cada nombre de archivo, o si prefiere agregar todos los archivos, presiona clic en el icono + que aparece al final del título Changes.

Presionamos clic en el icono + que aparece al final del título Changes, para que todos los archivos sean agregados a Stage Area, la cual se visualizará en el Source Control. Ahora en la caja de texto que aparece arriba del botón Commit, borre el texto que contiene y escriba el mensaje de observación del Commit, escribamos **“Ajuste de la vista del listado de comunas”**, por último, presione clic en el botón **Commit**.

Luego de realizar el Commit, el botón se titula Sync Changes 1 ↑, el cual al presionarlo, enviará los cambios al repositorio remoto. Solicitará confirmación para enviar los cambios, presione clic en OK



Presione clic en el botón, luego consulte el log con el comando **git log**, o puede ver que el repositorio remoto ahora aparecen 3 commits, quizá deba refrescar la ventana del navegador.



3.11. Ajustando el proyecto para que permita grabar una nueva Comuna

Crear la vista que contenga un formulario que permita capturar los datos de la comuna, dentro de la carpeta **comuna** que está en la carpeta **views**, cree un archivo llamado **new.blade.php**

Vaya a la página de bootstrap y copie la plantilla Starter Template y la pega en el archivo **new.blade.php**

<https://getbootstrap.com/docs/5.0/getting-started/introduction/>

```

new.blade.php U X
resources > views > comuna > new.blade.php > ...
1 <!doctype html>
2 <html lang="en">
3
4 <head>
5     <!-- Required meta tags -->
6     <meta charset="utf-8">
7     <meta name="viewport" content="width=device-width, initial-scale=1">
8
9     <!-- Bootstrap CSS -->
10    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"
11        integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
12
13    <title>Hello, world!</title>
14 </head>
15
16 <body>
17     <h1>Hello, world!</h1>
18
19     <!-- Optional JavaScript; choose one of the two! -->
20
21     <!-- Option 1: Bootstrap Bundle with Popper -->
22     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
23         integrity="sha384-MrcW6ZMFY1zclA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous">
24     </script>
25
26     <!-- Option 2: Separate Popper and Bootstrap JS -->
27     <!--
28     <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"
29         integrity="sha384-IQsoLX15PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iWwAwPtgFTxbJ8NT4GN1R8p" crossorigin="anonymous">
30     </script>
31     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js"
32         integrity="sha384-cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQF" crossorigin="anonymous">
33     </script>

```

Realice los cambio resaltados en la siguiente imagen

```

new.blade.php U X
resources > views > comuna > new.blade.php > html > body > div.container
1 <!doctype html>
2 <html lang="en">
3
4 <head>
5     <!-- Required meta tags -->
6     <meta charset="utf-8">
7     <meta name="viewport" content="width=device-width, initial-scale=1">
8
9     <!-- Bootstrap CSS -->
10    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"
11        integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
12
13    <title>Add Comuna</title>
14 </head>
15
16 <body>
17     <div class="container">
18         <h1>Add Comuna</h1>
19     </div>
20
21     <!-- Optional JavaScript; choose one of the two! -->
22
23     <!-- Option 1: Bootstrap Bundle with Popper -->
24    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
25        integrity="sha384-MrcW6ZMFY1zclA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous">
26    </script>
27
28     <!-- Option 2: Separate Popper and Bootstrap JS -->
29     <!--
30    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"
31        integrity="sha384-IQsoLX15PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iWwAwPtgFTxbJ8NT4GN1R8p" crossorigin="anonymous">
32    </script>

```


Ahora vaya a la sección de formularios de la página oficial de Bootstrap, y copie el código de la sección overview y lo pega debajo de la etiqueta h1.

<https://getbootstrap.com/docs/5.0/forms/overview/>

```

16 <body>
17   <div class="container">
18     <h1>Add Comuna</h1>
19     <form>
20       <div class="mb-3">
21         <label for="exampleInputEmail1" class="form-label">Email address</label>
22         <input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp">
23         <div id="emailHelp" class="form-text">We'll never share your email with anyone else.</div>
24       </div>
25       <div class="mb-3">
26         <label for="exampleInputPassword1" class="form-label">Password</label>
27         <input type="password" class="form-control" id="exampleInputPassword1">
28       </div>
29       <div class="mb-3 form-check">
30         <input type="checkbox" class="form-check-input" id="exampleCheck1">
31         <label class="form-check-label" for="exampleCheck1">Check me out</label>
32       </div>
33       <button type="submit" class="btn btn-primary">Submit</button>
34     </form>
35   </div>
36

```

Ahora vamos a ajustar el formulario de acuerdo a los datos que se deben capturar:

```

16 <body>
17   <div class="container">
18     <h1>Add Comuna</h1>
19     <form method="POST" action="{ route('comunas.store') }">
20       @csrf
21       <div class="mb-3">
22         <label for="id" class="form-label">Code</label>
23         <input type="text" class="form-control" id="id" aria-describedby="idHelp" name="id"
24           disabled="disabled">
25         <div id="idHelp" class="form-text">Commune code</div>
26       </div>
27       <div class="mb-3">
28         <label for="name" class="form-label">Commune</label>
29         <input type="text" required class="form-control" id="name" aria-describedby="nameHelp"
30           name="name" placeholder="Comuna name.">
31       </div>
32
33       <label for="municipality">Municipality:</label>
34       <select class="form-select" id="municipality" name="code" required>
35         <option selected disabled value="">Choose one...</option>
36         @foreach ($municipios as $municipio)
37           <option value="{ $municipio->muni_codi }" "{ $municipio->muni_nomb }</option>
38         @endforeach
39       </select>
40       <div class="mt-3">
41         <button type="submit" class="btn btn-primary">Save</button>
42         <a href="{ route('comunas.index') }" class="btn btn-warning">Cancel</a>
43       </div>
44     </form>
45   </div>

```

En la línea 19 se configura el formulario con método de envío **POST** y el acción será una ruta con nombre llamado **comuna.store**, el función **route** permite invocar una ruta por su nombre.

En la línea 20 se utiliza la directiva de blade **@csrf** la cual genera un token oculto en el formulario para indicar a la petición que es una petición válida de nuestro sitio. <https://laravel.com/docs/9.x/blade#csrf-field>

En la línea 36 se está utilizando la directiva de blade **@foreach** para iterare el arreglo **\$municipios** que deberán envían al momento de llamar la vista. Desde la línea 34 hasta la 39 se está construyendo un combo box con los datos de los municipios, del cual el usuario deberá seleccionar uno.

En la línea 42 se está utilizando un enlace para llamar la ruta nombrada **comunas.index** que utiliza la función **route**

Dado que en el formulario se están utilizando rutas nombradas vamos ajustar la ruta **/comunas** (get) que tendrá el nombre **comunas.index** y vamos a crear la ruta **/comunas** con método **POST** que invoque el método **store** de **ComunaController** y cuya ruta se llamará **comunas.store**.



```
new.blade.php U web.php M ●
routes > web.php > ...
You, 1 second ago | 1 author (You)
1 <?php
2
3 use App\Http\Controllers\ComunaController;
4 use Illuminate\Support\Facades\Route;
5
6 Route::get('/', function () {
7     return view('welcome');
8 });
9
10 Route::get('/comunas', [ComunaController::class, 'index'])->name('comunas.index');
11 Route::post('/comunas', [ComunaController::class, 'store'])->name('comunas.store');
12
13
14
```

La vista **new.blade.php** se llamará en el método **create** de **ComunaController**, el cual deberá recuperar los datos de los municipios y enviarlos a la vista.

Modifique el método **create** como se observa en la imagen:

```

26      /**
27       * Show the form for creating a new resource.
28       *
29       * @return \Illuminate\Http\Response
30       */
31      public function create()
32      {
33          $municipios = DB::table('tb_municipio')
34              ->orderBy('muni_nomb')
35              ->get();
36          return view('comuna.new', ['municipios' => $municipios]);
37      }
38  
```

El formulario creado en la vista **new** llamará a la ruta **comunas.store** la cual está asociada al método **store** de **ComunaController**. El método **store** se encargará de recibir los datos enviados por el formulario y utilizará el modelo para grabar los datos.

Modifique el método **store** como se observa en la imagen

```

39      /**
40       * Store a newly created resource in storage.
41       *
42       * @param \Illuminate\Http\Request $request
43       * @return \Illuminate\Http\Response
44       */
45      public function store(Request $request)
46      {
47          $comuna = new Comuna();
48          // $comuna->comu_codi = $request->id;
49          // El código de comuna es auto incremental
50          $comuna->comu_nomb = $request->name;
51          $comuna->muni_codi = $request->code;
52          $comuna->save();
53
54          $comunas = DB::table('tb_comuna')
55              ->join('tb_municipio', 'tb_comuna.muni_codi', '=', 'tb_municipio.muni_codi')
56              ->select('tb_comuna.*', "tb_municipio.muni_nomb")
57              ->get();
58          return view('comuna.index', ['comunas' => $comunas]);
59      }
60  
```

En el método se inyecta la clase Request que contiene toda la información de la petición.

En la línea 47 se crea un objeto del modelo Comuna y se guarda en la variable \$comuna.

En las líneas 50 y 51 se le asigna al modelo creado los datos recibidos de la petición.

En la línea 52 se utiliza el método save del modelo para grabar los datos en la base de datos.

En la línea 54 se consulta los datos de la columna utilizando Query Builder.

En la línea 58 se llama la vista **index** pasándole los datos de las comunas.

Ahora se debe agregar un enlace en la vista **index** de **comuna** para que pueda llamar la vista **new**, mediante una ruta **/comunas/create (get)** que llame al método create de **ComunaController**, la ruta tendrá el nombre **comunas.create**

Primero configure la nueva ruta en el archivo **web.php**



```
new.blade.php U  web.php M  ComunaController.php M
routes > web.php > ...
You, now | 1 author (You)
1  <?php
2
3  use App\Http\Controllers\ComunaController;
4  use Illuminate\Support\Facades\Route;
5
6  Route::get('/', function () {
7      return view('welcome');
8  });
9
10 Route::get('/comunas', [ComunaController::class, 'index'])->name('comunas.index');
11 Route::post('/comunas', [ComunaController::class, 'store'])->name('comunas.store');
12 Route::get('/comunas/create', [ComunaController::class, 'create'])->name('comunas.create');
13
14
```

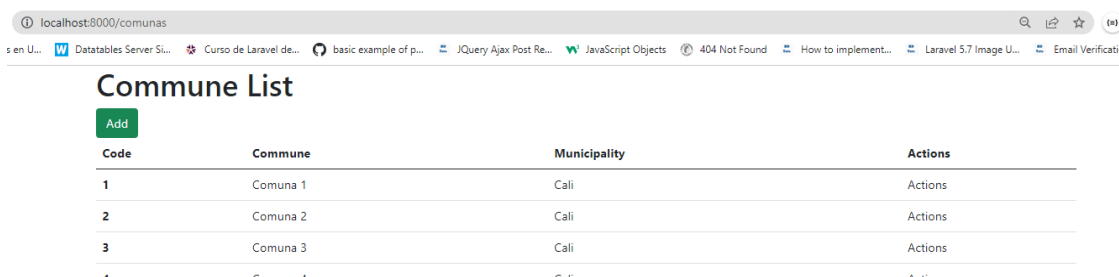
Abra el archivo **index.blade.php** y agregue el enlace para llamar a la vista **new**

```

16 <body>
17 <div class="container">
18 <h1>Listado de Comunas</h1>
19 <a href="{{ route('comunas.create') }}" class="btn btn-success">Add</a>
20 <table class="table">
21 <thead>

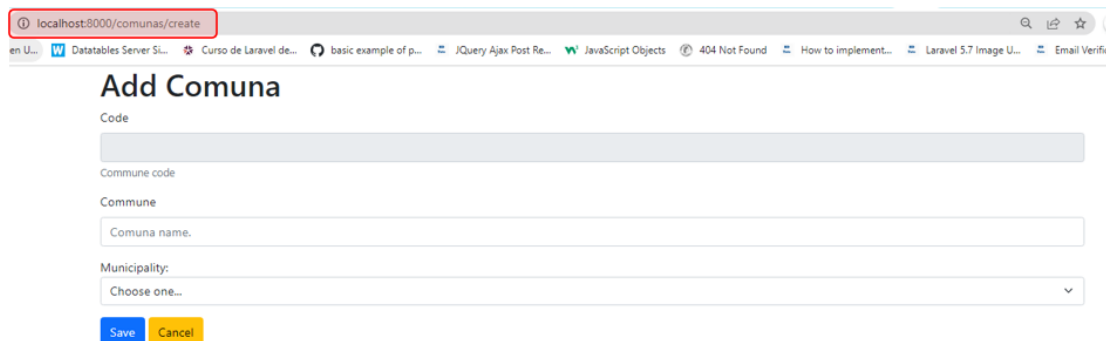
```

Es hora de probar que se puede grabar nuevas comunas, vaya al navegador y llame la ruta <http://localhost:8000/comunas>, ahora la vista tiene el botón **Add**.



Code	Commune	Municipality	Actions
1	Comuna 1	Cali	Actions
2	Comuna 2	Cali	Actions
3	Comuna 3	Cali	Actions
4	Comuna 4	Cali	Actions

Presione clic en el botón para llamar el formulario y crear una nueva Comuna, observe la ruta que levantó. Diligencie los datos y presione clic en el botón **Save**



3.12. Generando una nueva versión del proyecto

Genere una nueva versión de su proyecto, con el mensaje **"Commune Add"** y actualice su repositorio remoto.

3.13. Ajustando el proyecto para que permita borrar una Comuna

Abra el archivo de **ComunaController.php** y modifique el método **destroy** como se observa en la imagen:

```

106      /**
107       * Remove the specified resource from storage.
108       *
109       * @param int $id
110       * @return \Illuminate\Http\Response
111       */
112      public function destroy($id)
113      {
114          $comuna = Comuna::find($id);
115          $comuna->delete();
116
117          $comunas = DB::table('tb_comuna')
118              ->join('tb_municipio', 'tb_comuna.muni_codi', '=', 'tb_municipio.muni_codi')
119              ->select('tb_comuna.*', "tb_municipio.muni_nomb")
120              ->get();
121
122          return view('comuna.index', ['comunas' => $comunas]);
123      }

```

El método **destroy** recibe el **id** de la comuna que desea borrar, mediante **Eloquent** usando el modelo y el método **find** busca la comuna con dicho **id** y lo guarda en la variable **\$comuna**, luego lo borra utilizando el método **delete** del modelo.

Posteriormente almacena en la variable **\$comunas** todos los registros regresados por la consulta de Query Builder.

Ahora, abra el archivo **index.blade.php** y realice las modificaciones que se observan en la imagen:

```

30      @foreach ($comunas as $comuna)
31          <tr>
32              <th scope="row">{{ $comuna->comu_codi }}</th>
33              <td>{{ $comuna->comu_nomb }}</td>
34              <td>{{ $comuna->muni_nomb }}</td>
35              <td>
36                  <form action="{{ route('comunas.destroy', ['comuna' => $comuna->comu_codi]) }}"
37                      method="POST" style="display: inline-block">
38                      @method('delete')
39                      @csrf
40                      <input class="btn btn-danger" type="submit" value="Delete">
41                  </form>
42              </td>
43          </tr>
44      @endforeach

```

Se incluye un formulario que llama la ruta **comunas.destroy**, método de envío **POST** y pasándole como parámetro el código de la comuna. Se utiliza la directiva **@method('delete')** para modificar el método de envío al formulario, esto dado que el parámetro action de form no acepta el método **delete**.

Ahora se debe configurar la nueva ruta, abrir el archivo **web.php** y agregar la nueva ruta.

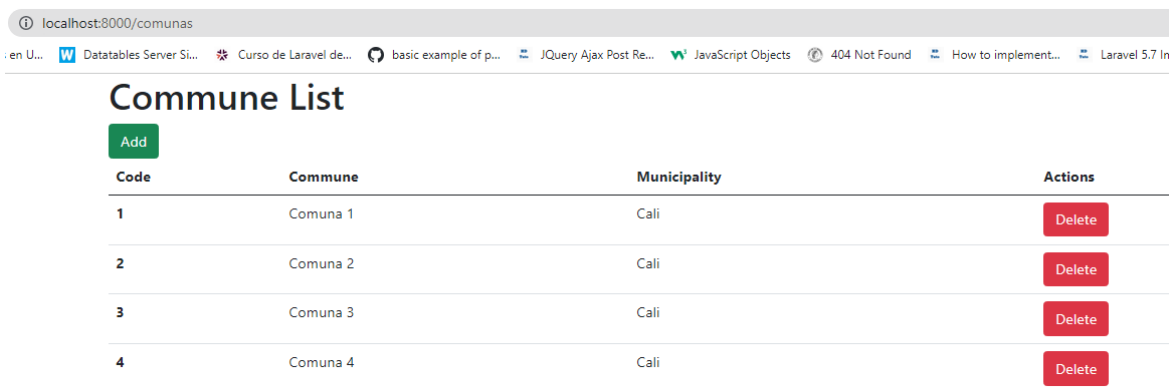


```

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

```

Vaya al navegador y actualice la ruta <http://localhost:8000/comunas> presione clic en el botón delete de una comuna y verifique que la ha eliminado.



Code	Commune	Municipality	Actions
1	Comuna 1	Cali	Delete
2	Comuna 2	Cali	Delete
3	Comuna 3	Cali	Delete
4	Comuna 4	Cali	Delete

3.14. Generando una nueva versión del proyecto

Genere una nueva versión de su proyecto, con el mensaje **"Commune Delete"** y actualice su repositorio remoto.

3.15. Ajustando el proyecto para que permita editar una Comuna

Cree un archivo **edit.blade.php** en la carpeta **comuna de vistas** y copie en este el contenido que tiene el archivo **new.blade.php**, y realice los cambios que se observan en la siguiente imagen:

```

17 <div class="container">
18 <h1>Edit Commune</h1>
19 <form method="POST" action="{{ route('comunas.update', ['comuna' => $comuna->comu_codi]) }}">
20     @method('put')
21     @csrf
22     <div class="mb-3">
23         <label for="codigo" class="form-label">Id</label>
24         <input type="text" class="form-control" id="id" aria-describedby="codigoHelp" name="id"
25             disabled="disabled" value="{{ $comuna->comu_codi }}">
26         <div id="codigoHelp" class="form-text">Commune Id.</div>
27     </div>
28     <div class="mb-3">
29         <label for="name" class="form-label">Commune</label>
30         <input type="text" required class="form-control" id="name" placeholder="Commune name"
31             name="name" value="{{ $comuna->comu_nomb }}">
32     </div>
33
34     <label for="municipality">Municipality:</label>
35     <select class="form-select" id="municipality" name="code" required>
36         <option selected disabled value="">Choose one...</option>
37         @foreach ($municipios as $municipio)
38             @if ($municipio->muni_codi == $comuna->muni_codi)
39                 <option selected value="{{ $municipio->muni_codi }}">{{ $municipio->muni_nomb }}</option>
40             @else
41                 <option value="{{ $municipio->muni_codi }}">{{ $municipio->muni_nomb }}</option>
42             @endif
43         @endforeach
44     </select>
45     <div class="mt-3">
46         <button type="submit" class="btn btn-primary">Update</button>
47         <a href="{{ route('comunas.index') }}" class="btn btn-warning">Cancel</a>
48     </div>
49 </form>
50 </div>

```

Se incluye un formulario que llama la ruta **comunas.update**, método de envío **POST** y pasándole como parámetro el código de la comuna. Se utiliza la directiva **@method('put')** para modificar el método de envío al formulario, esto dado que el parámetro action de form no acepta el método **put**.

Ahora se debe configurar la nueva ruta, abrir el archivo **web.php** y agregar la nueva ruta.


```

edit.blade.php U  web.php M
routes > web.php > ...
You, 42 seconds ago | 1 author (You)
1  <?php
2
3  use App\Http\Controllers\ComunaController;
4  use Illuminate\Support\Facades\Route;
5
6  Route::get('/', function () {
7      return view('welcome');
8  });
9
10 Route::get('/comunas', [ComunaController::class, 'index'])->name('comunas.index');
11 Route::post('/comunas', [ComunaController::class, 'store'])->name('comunas.store');
12 Route::get('/comunas/create', [ComunaController::class, 'create'])->name('comunas.create');
13 Route::delete('/comunas/{comuna}', [ComunaController::class, 'destroy'])->name('comunas.destroy');
14 Route::put('/comunas/{comuna}', [ComunaController::class, 'update'])->name('comunas.update');
15

```

Abra el archivo **ComunaController.php** y ajuste el método **update**

```

90 public function update(Request $request, $id)
91 {
92     $comuna = Comuna::find($id);
93
94     $comuna->comu_nomb = $request->name;
95     $comuna->muni_codi = $request->code;
96     $comuna->save();
97
98     $comunas = DB::table('tb_comuna')
99         ->join('tb_municipio', 'tb_comuna.muni_codi', '=', 'tb_municipio.muni_codi')
100         ->select('tb_comuna.*', "tb_municipio.muni_nomb")
101         ->get();
102
103     return view('comuna.index', ['comunas' => $comunas]);
104 }
105

```

El método **update** inyecta la clase **Request** y recibe el **id** de la comuna que desea editar, mediante **Eloquent** usando el modelo y el método **find** busca la comuna con dicho **id** y lo guarda en la variable **\$comuna**, cambia los datos del objeto por los datos recibidos en la petición..

Posteriormente almacena en la variable **\$comunas** todos los registros regresados por la consulta de Query Builder.

Abra el archivo **index.blade.php** y realice la modificación que se observa en la imagen

```

30         @foreach ($comunas as $comuna)
31             <tr>
32                 <th scope="row">{{ $comuna->comu_codi }}</th>
33                 <td>{{ $comuna->comu_nomb }}</td>
34                 <td>{{ $comuna->muni_nomb }}</td>
35                 <td>
36                     <a href="{{ route('comunas.edit', ['comuna' => $comuna->comu_codi]) }}"
37                         class="btn btn-info"> Edit </a></td>
38
39                     <form action="{{ route('comunas.destroy', ['comuna' => $comuna->comu_codi]) }}"
40                         method="POST" style="display: inline-block">
41                         @method('delete')
42                         @csrf
43                         <input class="btn btn-danger" type="submit" value="Delete">
44                     </form>
45                 </td>
46             </tr>
47         @endforeach

```

Se agrega un enlace que llama la ruta **comunas.edit**, pasando como parámetro el código de la comuna. La ruta **comuna.edit**, hace tiene el patrón **/comunas/{comuna}/edit** y llama al método **edit** del controlador asociado.

Abra el archivo **web.php** y adicione la ruta

```

edit.blade.php U  web.php M  ComunaController.php M  index.blade.php M
routes > web.php > ...
You, now | 1 author (You)
<?php
1
2
3 use App\Http\Controllers\ComunaController;
4 use Illuminate\Support\Facades\Route;
5
6 Route::get('/', function () {
7     return view('welcome');
8 });
9
10 Route::get('/comunas', [ComunaController::class, 'index'])->name('comunas.index');
11 Route::post('/comunas', [ComunaController::class, 'store'])->name('comunas.store');
12 Route::get('/comunas/create', [ComunaController::class, 'create'])->name('comunas.create');
13 Route::delete('/comunas/{comuna}', [ComunaController::class, 'destroy'])->name('comunas.destroy');
14 Route::put('/comunas/{comuna}', [ComunaController::class, 'update'])->name('comunas.update');
15 Route::get('/comunas/{comuna}/edit', [ComunaController::class, 'edit'])->name('comunas.edit');
16
17

```

Modifique el método **edit** de **ComunaController**, como se observa en la imagen

```

74  /**
75   * Show the form for editing the specified resource.
76   *
77   * @param int $id
78   * @return \Illuminate\Http\Response
79   */
80  public function edit($id)
81  {
82      //
83      $comuna = Comuna::find($id);
84      $municipios = DB::table('tb_municipio')
85          ->orderBy('muni_nomb')
86          ->get();
87      return view('comuna.edit', ['comuna' => $comuna, 'municipios' => $municipios]);
88  }
89

```

Vaya al navegador y actualice la ruta <http://localhost:8000/comunas> presione clic en el botón edit de una comuna y modifique sus datos.

localhost:8000/comunas

Commune List

Add

Code	Commune	Municipality	Actions
1	Comuna 1	Cali	Edit Delete
2	Comuna 2	Cali	Edit Delete
3	Comuna 3	Cali	Edit Delete

localhost:8000/comunas/1/edit

Edit Commune

Id
1

Commune Id.

Commune
Comuna 1111

Municipality:
Cali

Update Cancel

localhost:8000/comunas/1

Commune List

Add

Code	Commune	Municipality	Actions
1	Comuna 1111	Cali	Edit Delete
2	Comuna 2	Cali	Edit Delete

3.16. Generando una nueva versión del proyecto

Genere una nueva versión de su proyecto, con el mensaje **"Commune Edit"** y actualice su repositorio remoto.

3.17. Ahora crea el CRUD para las entidades: **municipios, departamentos y país.**