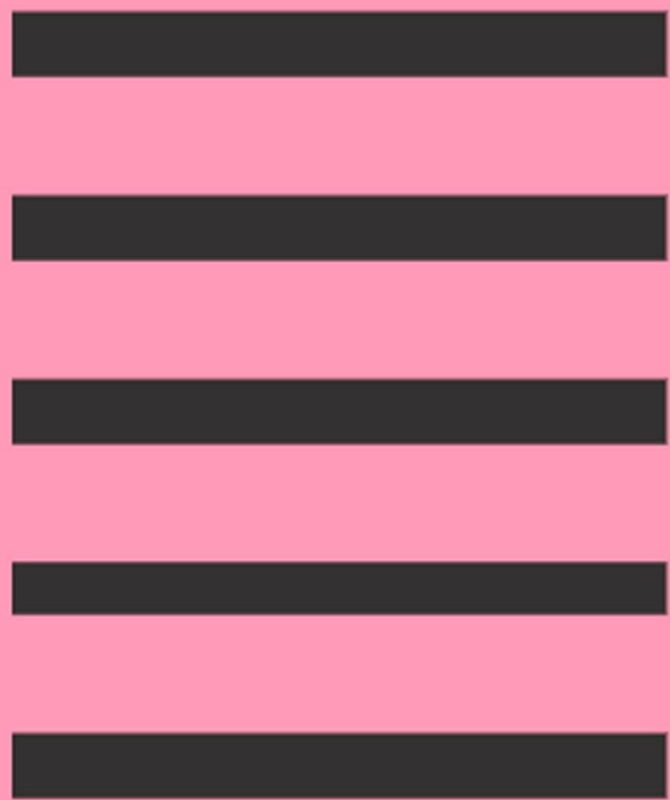


# MANUAL DEL PROGRAMADOR

APLICACIONES WEB INTERACTIVAS

---

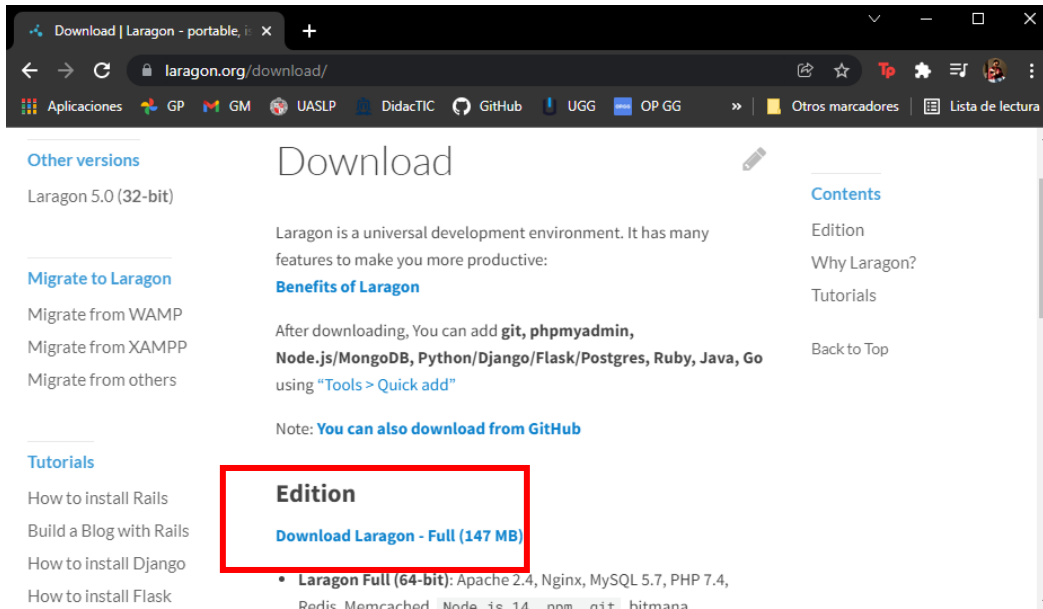
## SHALALA



# DESCARGAS

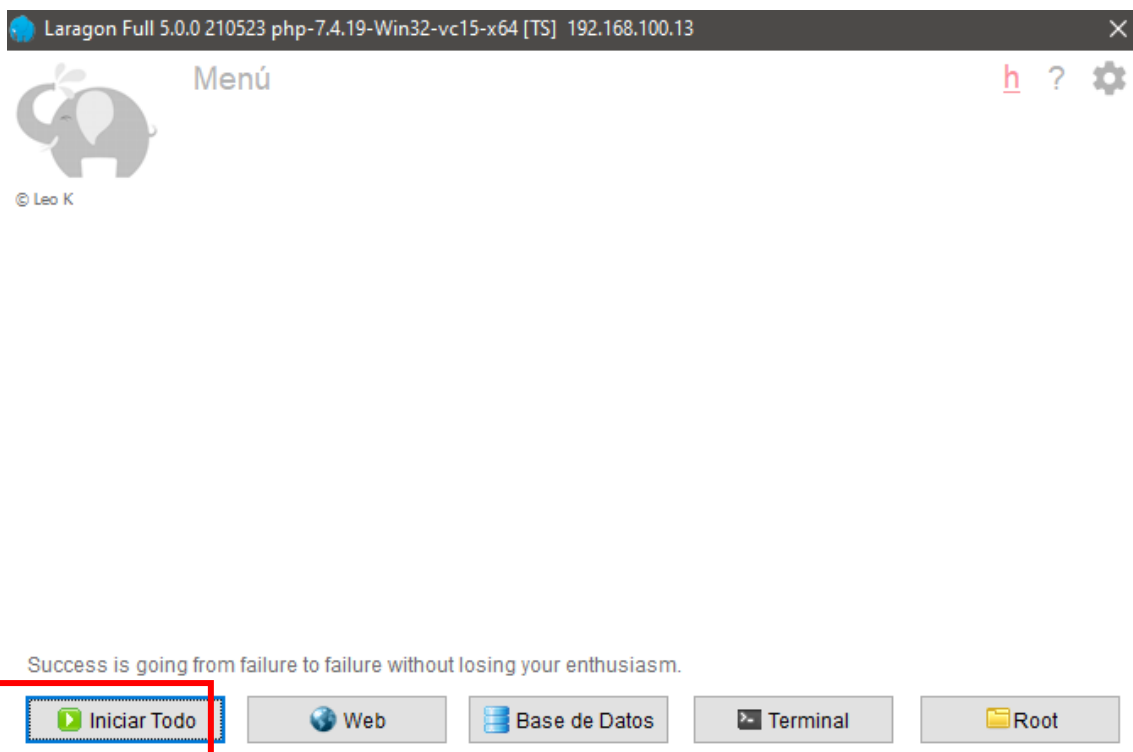
Laragon

<https://laragon.org/download/>

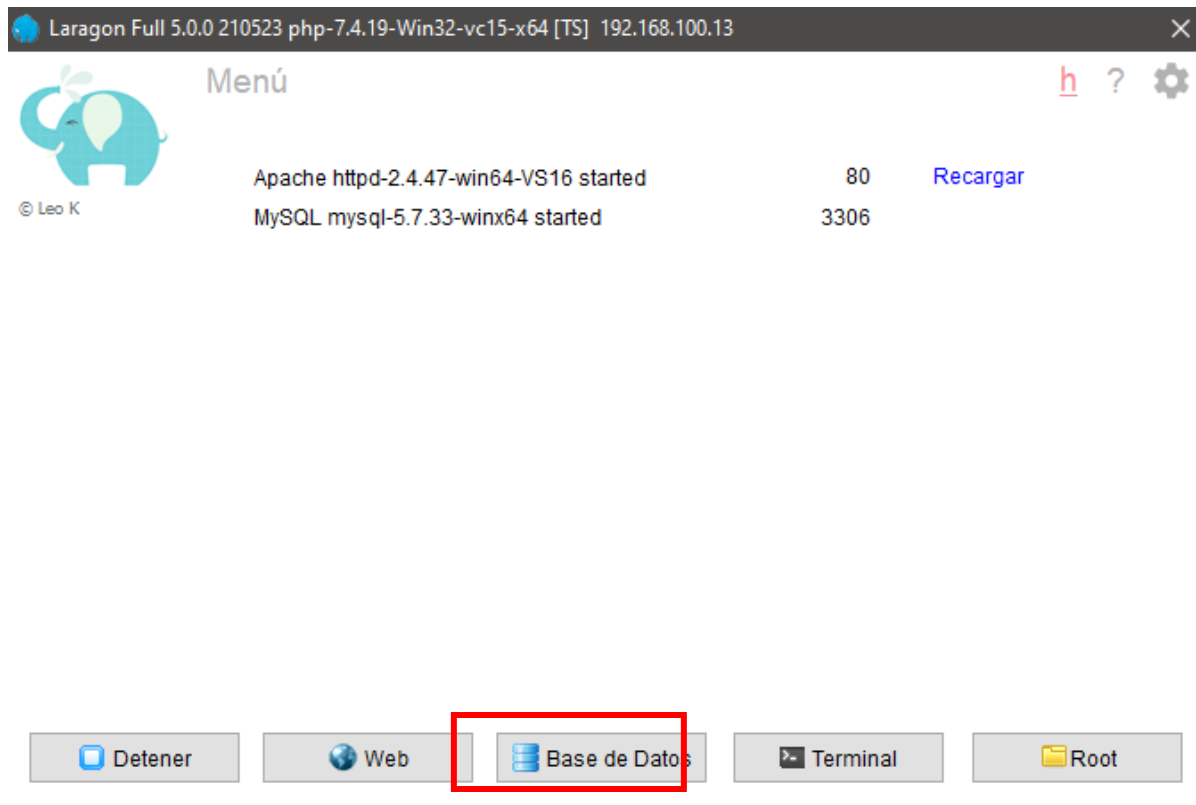


# MONTAJE DEL SISTEMA

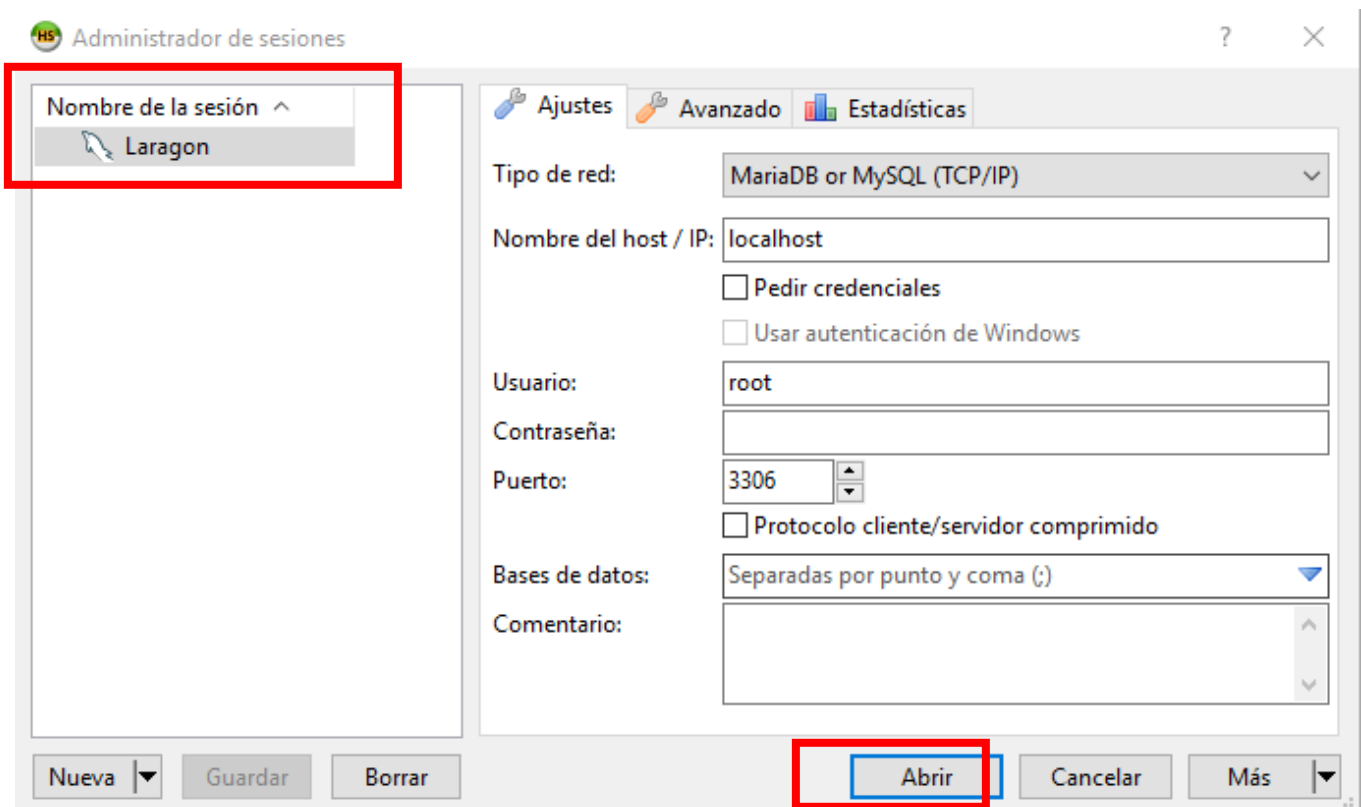
Ya instalado el software de laragon, lo abrimos y presionamos el botón.



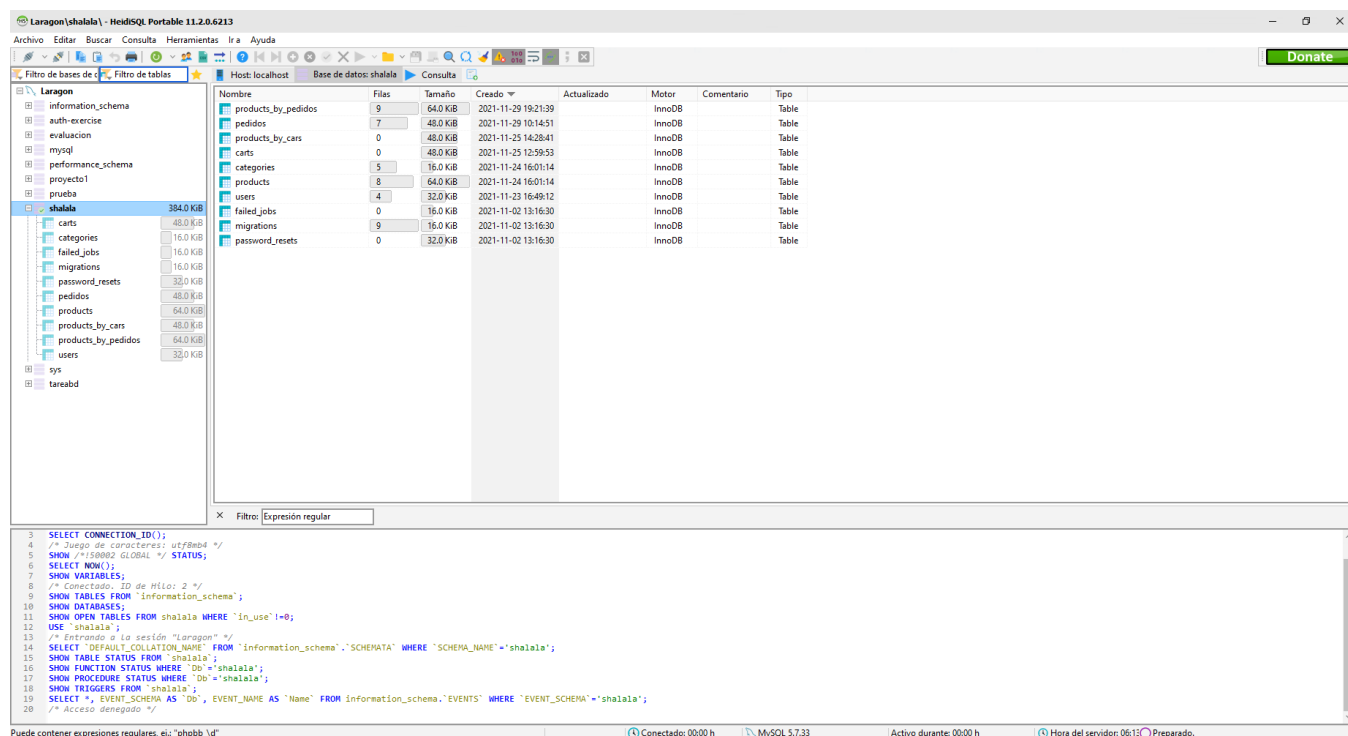
Después de presionar el botón de Iniciar Todo, así lucirá nuestro programa.



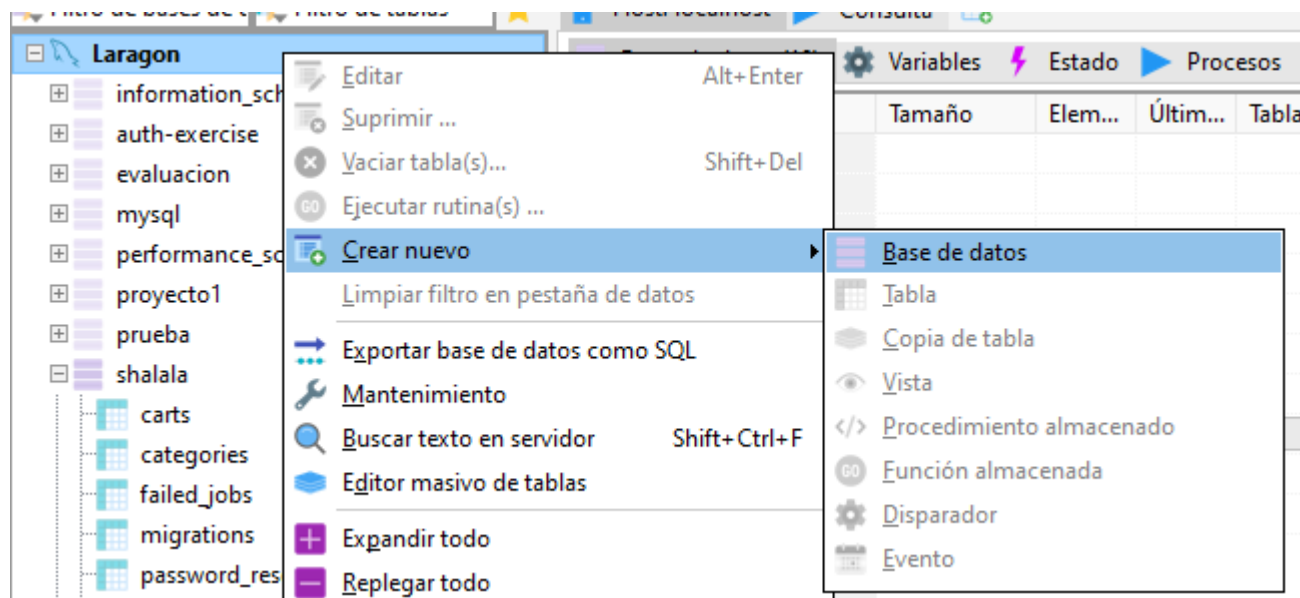
Iremos al botón **Base de Datos**, damos click y nos aparecerá algo así.



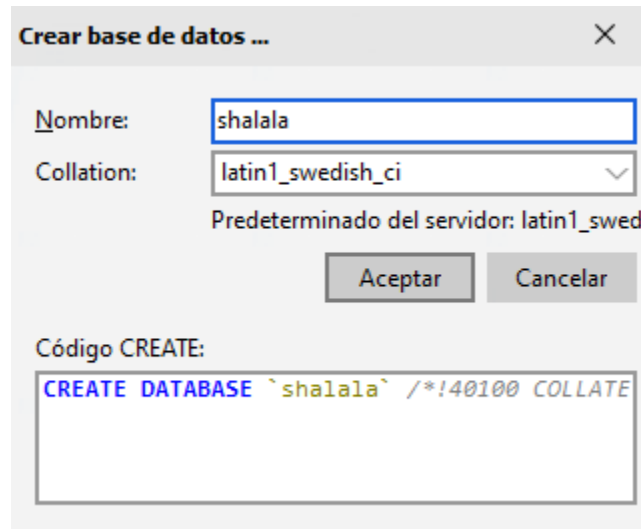
Si no te aparece una base de datos como esta, créala con el mismo nombre **Laragon**.



Ya dentro tendremos algo como esto, nos iremos a **Laragon** y crearemos una nueva base de datos, con click derecho, crear nuevo, base de datos.



A la nueva base de datos la llamaremos **shalala**.



Aceptamos y listo, ya montamos la **base de datos**.

## Link del proyecto

<https://github.com/Aplicaciones-web-interactivas/proyecto-Cordova-Serrano>

Vamos a una terminal, ya sea CMD, PowerShell o GitBash, para clonar el repositorio a nuestra carpeta de **Laragon > www**.

```
MINGW64: c:/laragon/www

SVMPLE@DESKTOP-3027B6B MINGW64 ~
$ cd ..

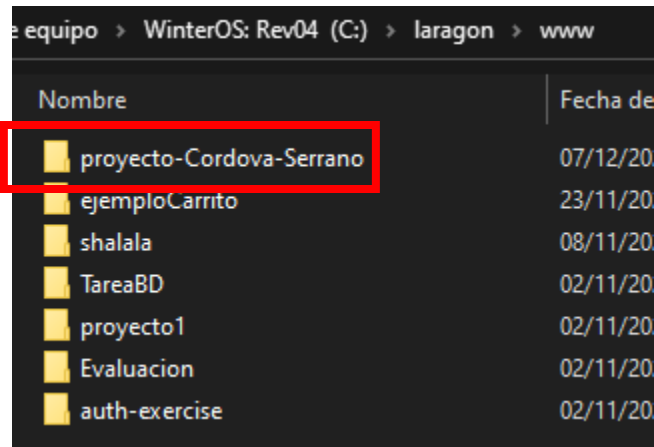
SVMPLE@DESKTOP-3027B6B MINGW64 /c/Users
$ cd ..

SVMPLE@DESKTOP-3027B6B MINGW64 /c
$ cd laragon

SVMPLE@DESKTOP-3027B6B MINGW64 /c/laragon
$ cd www

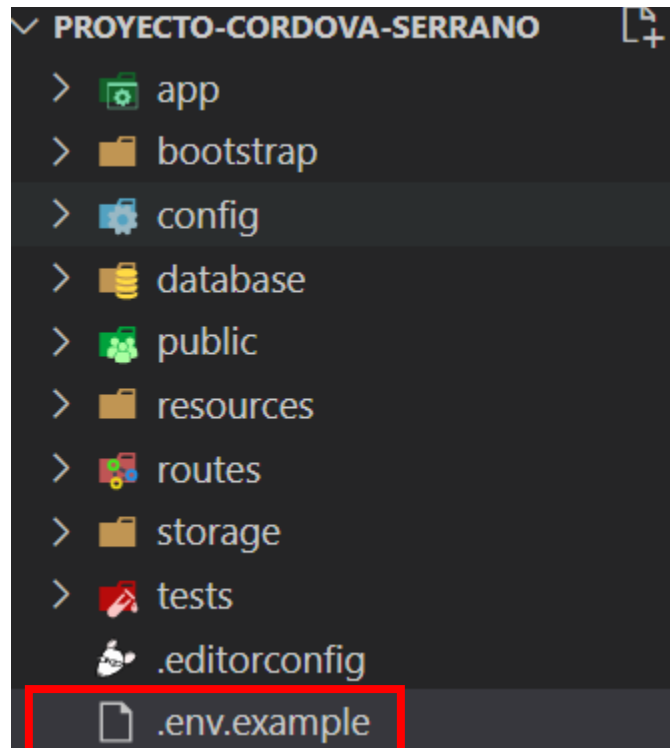
SVMPLE@DESKTOP-3027B6B MINGW64 /c/laragon/www
$ git clone https://github.com/Aplicaciones-web-interactivas/proyecto-Cordova-Serrano.git
Cloning into 'proyecto-Cordova-Serrano'...
remote: Enumerating objects: 574, done.
remote: Total 574 (delta 0), reused 0 (delta 0), pack-reused 574
Receiving objects: 100% (574/574), 708.42 MiB | 7.66 MiB/s, done.
Resolving deltas: 100% (265/265), done.
Updating files: 100% (285/285), done.

SVMPLE@DESKTOP-3027B6B MINGW64 /c/laragon/www
$
```



Ya teniendo ubicado que el proyecto se clono correctamente, editaremos un archivo para que conecte correctamente con la base de datos.

Abrimos la carpeta con **Visual Studio Code**, o cualquier **IDE** de su preferencia.



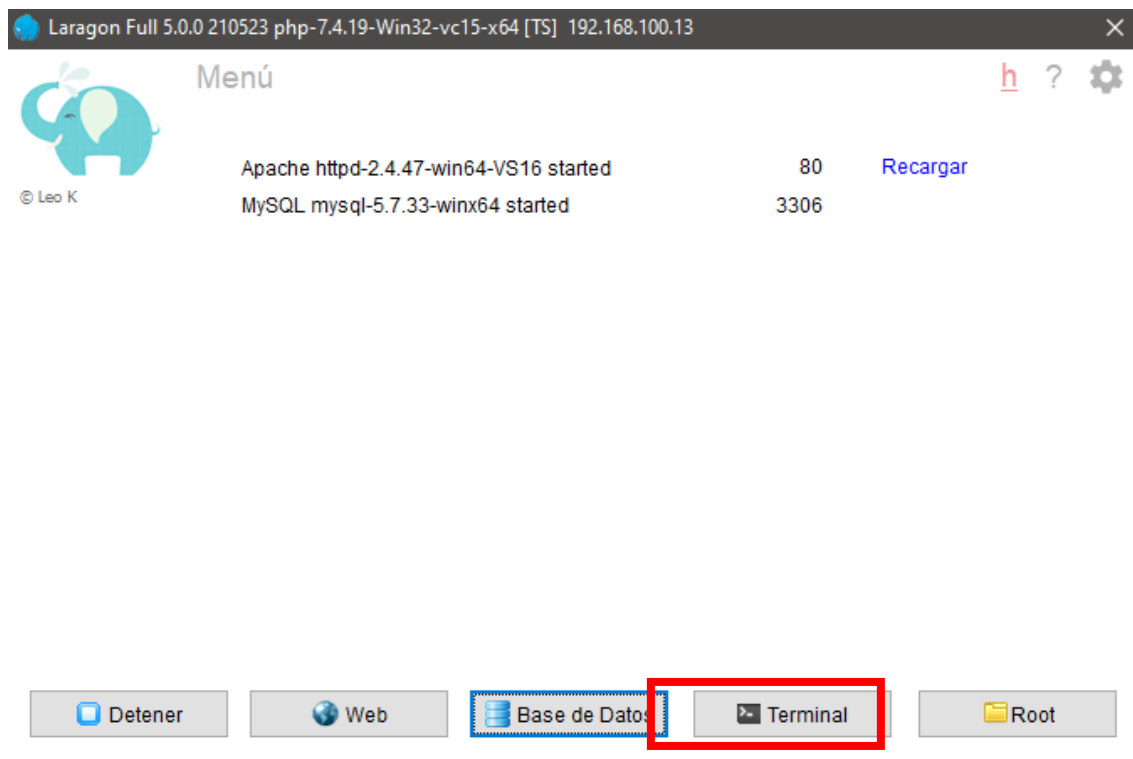
Ya en el IDE, editamos ese archivo **.env.example**, le cambiamos el nombre a **.env**

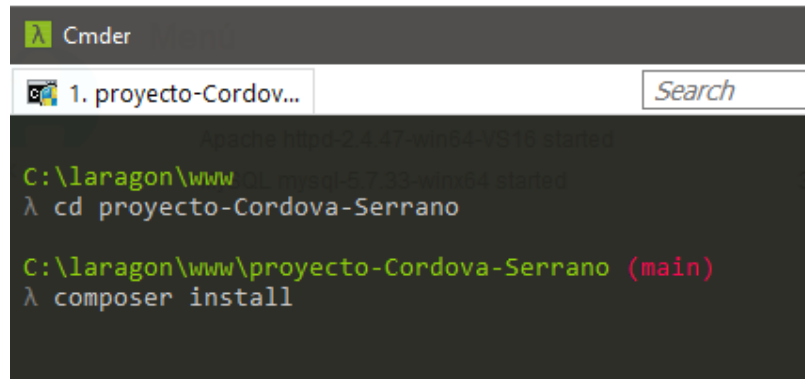
```
EXPLORADOR
✓ EDITORES ABIERTOS
  X .env
✓ PROYECTO-CORDOVA-SERRANO
  > app
  > bootstrap
  > config
  > database
  > public
  > resources
  > routes
  > storage
  > tests
  .editorconfig
  .env

.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8
9 DB_CONNECTION=mysql
10 DB_HOST=127.0.0.1
11 DB_PORT=3306
12 DB_DATABASE=laravel
13 DB_USERNAME=root
14 DB_PASSWORD=
15
```

Ya editado el archivo, editamos el valor de **DB\_DATABASE = laravel**, por **DB\_DATABASE = shalala**, y eso conectará nuestro programa a la base de datos correctamente.

Ahora dentro de laragon, abrimos una terminal.

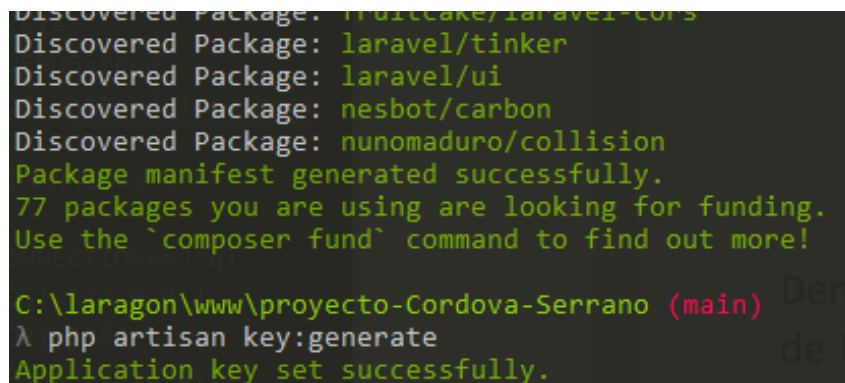




```
Cmder
1. proyecto-Cordov... Search
Apache httpd-2.4.47-win64-VS16 started
C:\laragon\www
mysql-5.7.33-win64 started
λ cd proyecto-Cordova-Serrano
C:\laragon\www\proyecto-Cordova-Serrano (main)
λ composer install
```

Dentro de la terminal, nos ubicaremos en la carpeta del proyecto, ya dentro de la carpeta del proyecto instalaremos los paquetes necesarios para su correcto funcionamiento.

- **composer install**
- **php artisan key:generate**
- **php artisan migrate:fresh --seed**
- **php artisan storage:link**

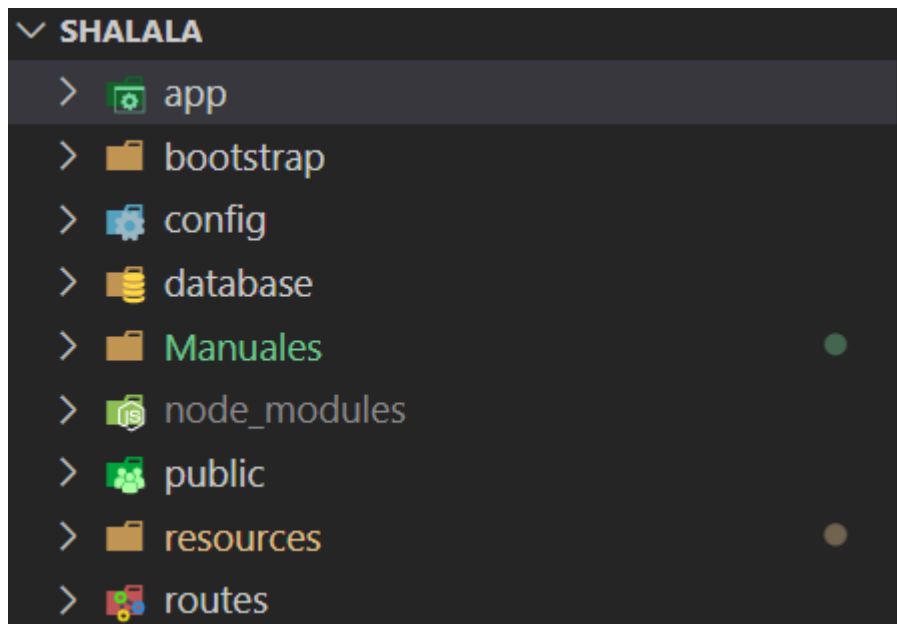


```
Discovered Package: fruitcake/laravel-cors
Discovered Package: laravel/tinker
Discovered Package: laravel/ui
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
77 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

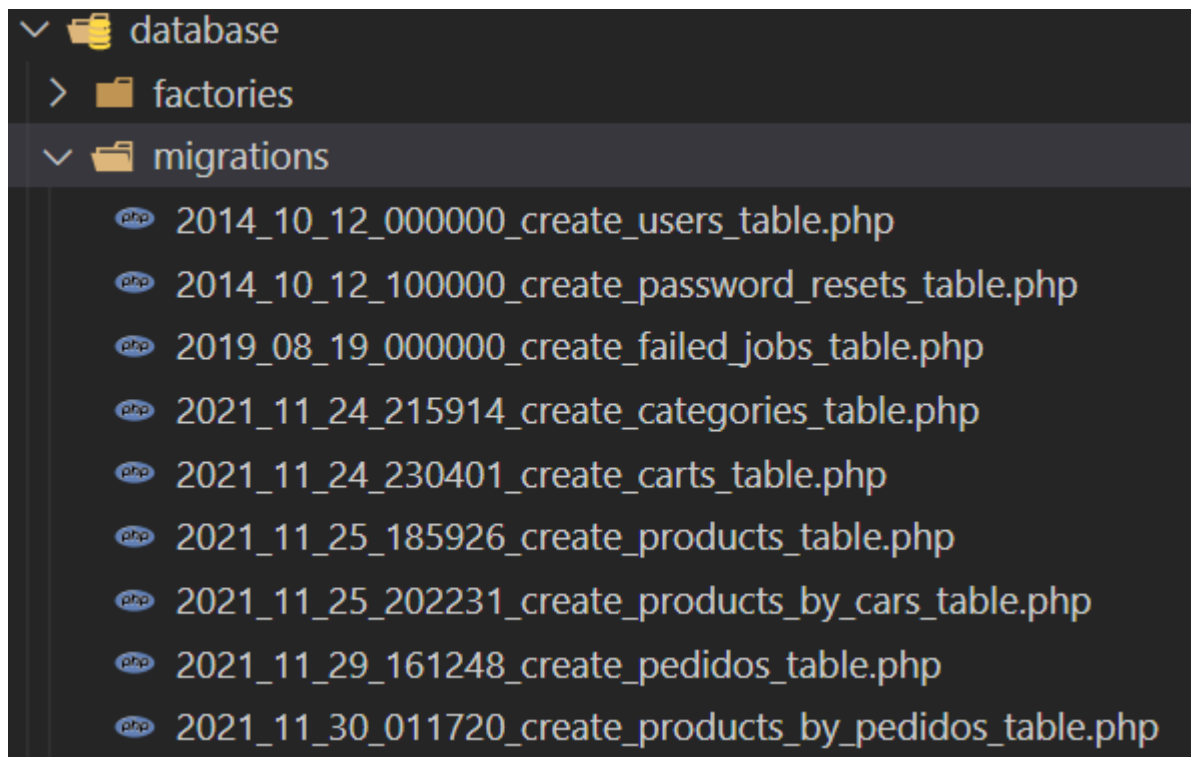
C:\laragon\www\proyecto-Cordova-Serrano (main)
λ php artisan key:generate
Application key set successfully.
```



# DESARROLLADOR

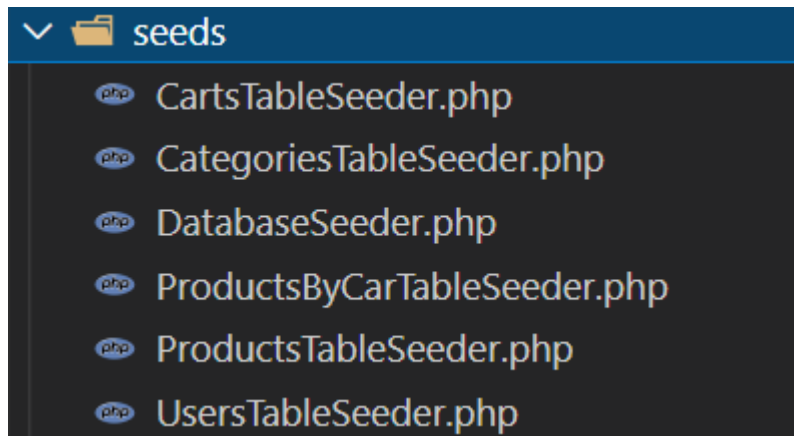


## Database: Migrations



En las migraciones nos encontramos las estructuras de los **Schemas** de cada una de las tablas de la base de datos.

## Database: Seeds

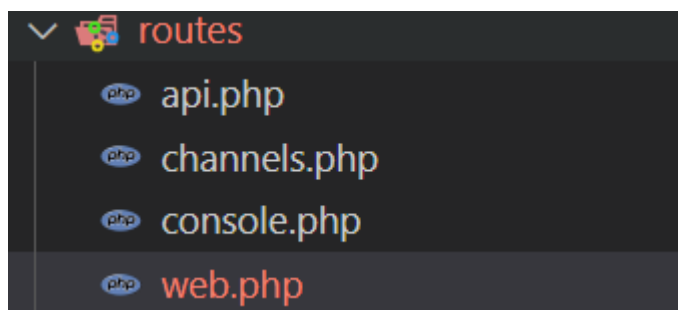


En las semillas nos encontramos los archivos para la creación de información de prueba que rellene la base de datos. Por eso anteriormente usamos este comando:

- **php artisan migrate:fresh --seed**

Este comando se utiliza para correr las migraciones junto a las semillas.

## Routes: web.php



En este archivo se encuentran las rutas del sistema, junto a sus funciones y controladores, para su correcto funcionamiento.

## Ejemplo:

```
Auth::routes();

Route::get('/home', 'HomeController@index')->name('home');

Route::get('/productos', 'ProductController@showProducts');
Route::get('/producto/{category}/{id}', 'ProductController@showOneProduct');
Route::get('/carrito/{id}', 'ProductsByCarController@show');
Route::get('/admin', 'HomeController@admin')->middleware('auth');

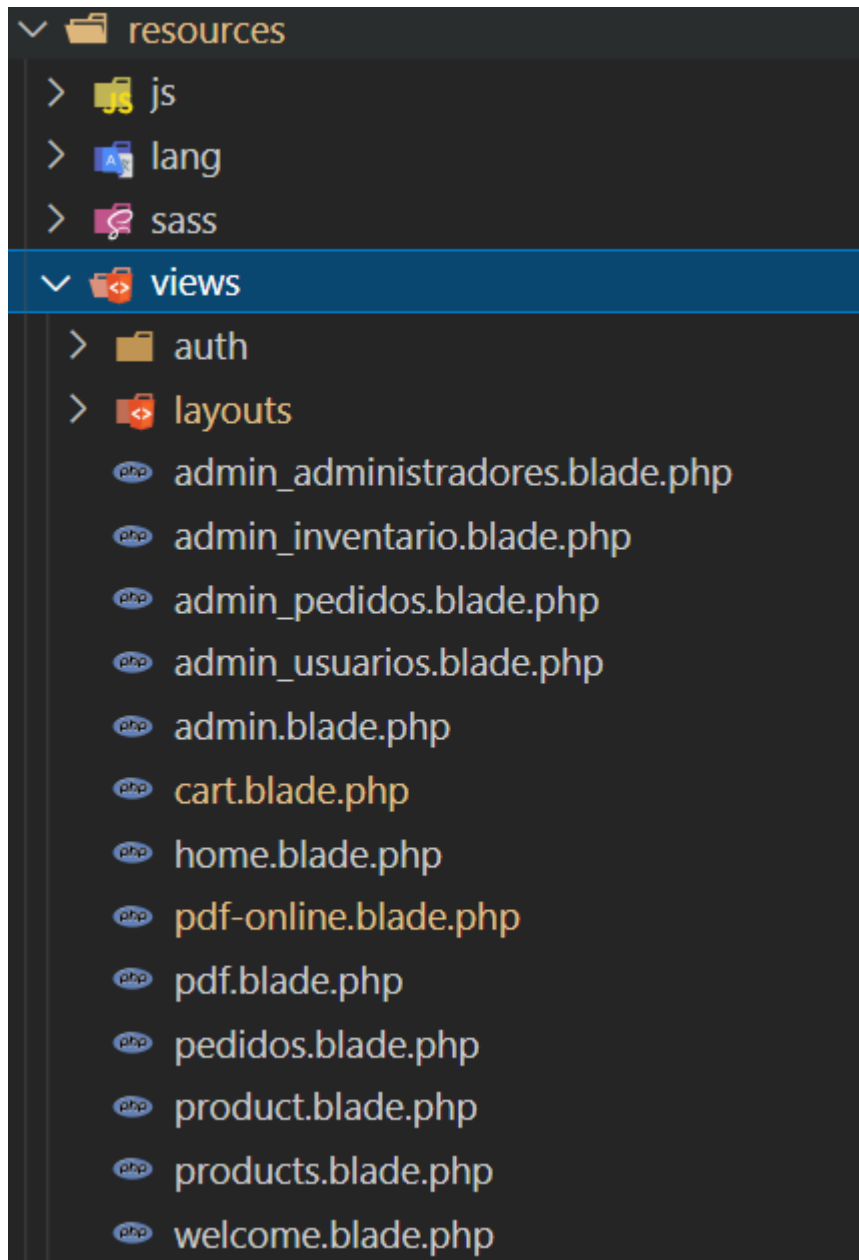
Route::get('/admin/inventario', 'ProductController@show')->middleware('auth');

Route::get('/admin/pedidos', 'PedidosController@show')->middleware('auth');
Route::get('/admin/usuarios', 'UsuariosController@showUsuarios')->middleware('auth');
Route::get('/admin/administradores', 'UsuariosController@showAdmins')->middleware('auth');
Route::get('/user/{id}', 'UsuariosController@usuariosValidation');
Route::get('/delete/{id}', 'ProductController@delete');
Route::get('/deleteFromCart/{id}', 'ProductsByCarController@delete');
Route::get('/pedidos/{id}', 'PedidosController@showById');

Route::post('/guardaUsuario', 'RegisterController@store');
Route::post('/guardaNuevaCategoria', 'CategoryController@store');
Route::post('/storeProduct', 'ProductController@store');
Route::post('/editProduct', 'ProductController@edit');
Route::post('/addToCart', 'ProductsByCarController@addToCart');
Route::post('/buyCart', 'PedidosController@buyCart');|

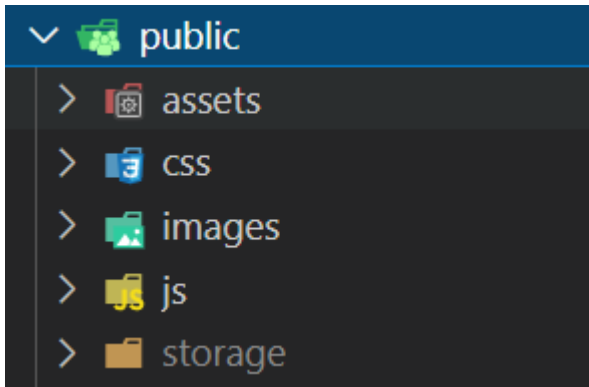
//PDF Rutas
Route::get('/ver-compra/{user_id}/{pedido_id}', 'PedidosController@verPDF'); //Ver PDF
Route::get('/downloadPDF/{user_id}/{pedido_id}', 'PedidosController@createPDF'); //Creacion del Ticket de Compra
```

## Resources: Views



En esta carpeta nos concentraremos en la parte de vistas que ya son la parte visual del sistema, donde contamos con **layouts**, que se utilizan en ciertas vistas.

## Public



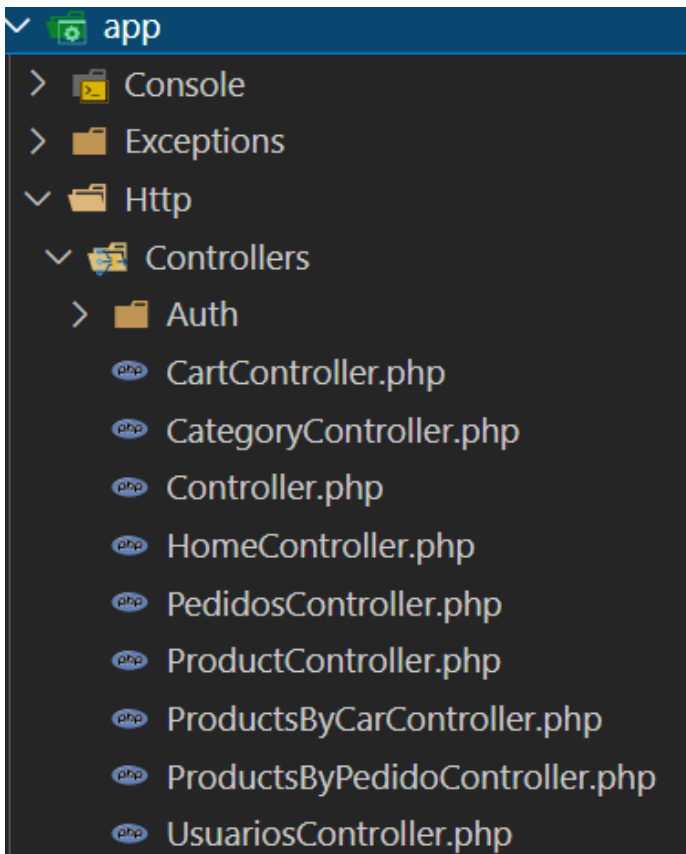
En esta carpeta de **Public**, nos encontramos con **assets**, donde guardaremos cualquier imagen.

La carpeta de estilos **css**. La carpeta de **imágenes**. La carpeta de funciones de **javascript**, **js**. Y como anteriormente utilizamos un comando:

- **php artisan storage:link**

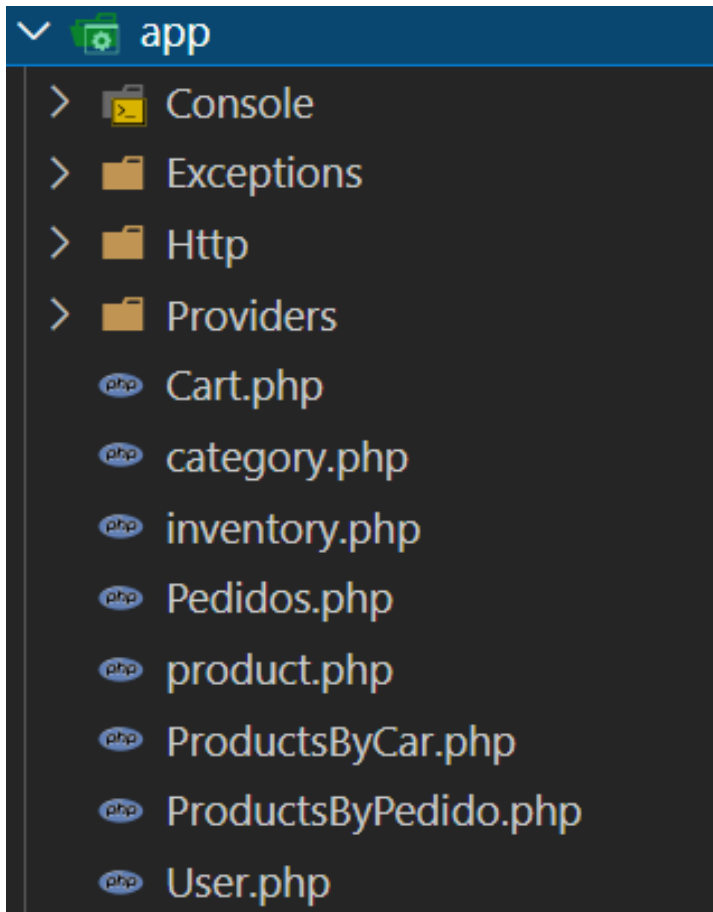
Este comando funciona para ligar el **storage** con nuestra carpeta **public** y así poder acceder a los elementos que estén guardados en **storage**.

## App: Http



En esta carpeta se encuentran los archivos **Controladores** del sistema, el cual está basado en **Modelo Vista Controlador**.

## App: Models



En esta carpeta se encuentran los **Modelos** del sistema.