

Aprendiendo a programar en Laravel

Por Alejandro Jiménez Sosa

Prerrequisitos



Hacemos posible laravel en composer

- Nos situamos en /xampp/htdocs y ejecutamos el siguiente comando

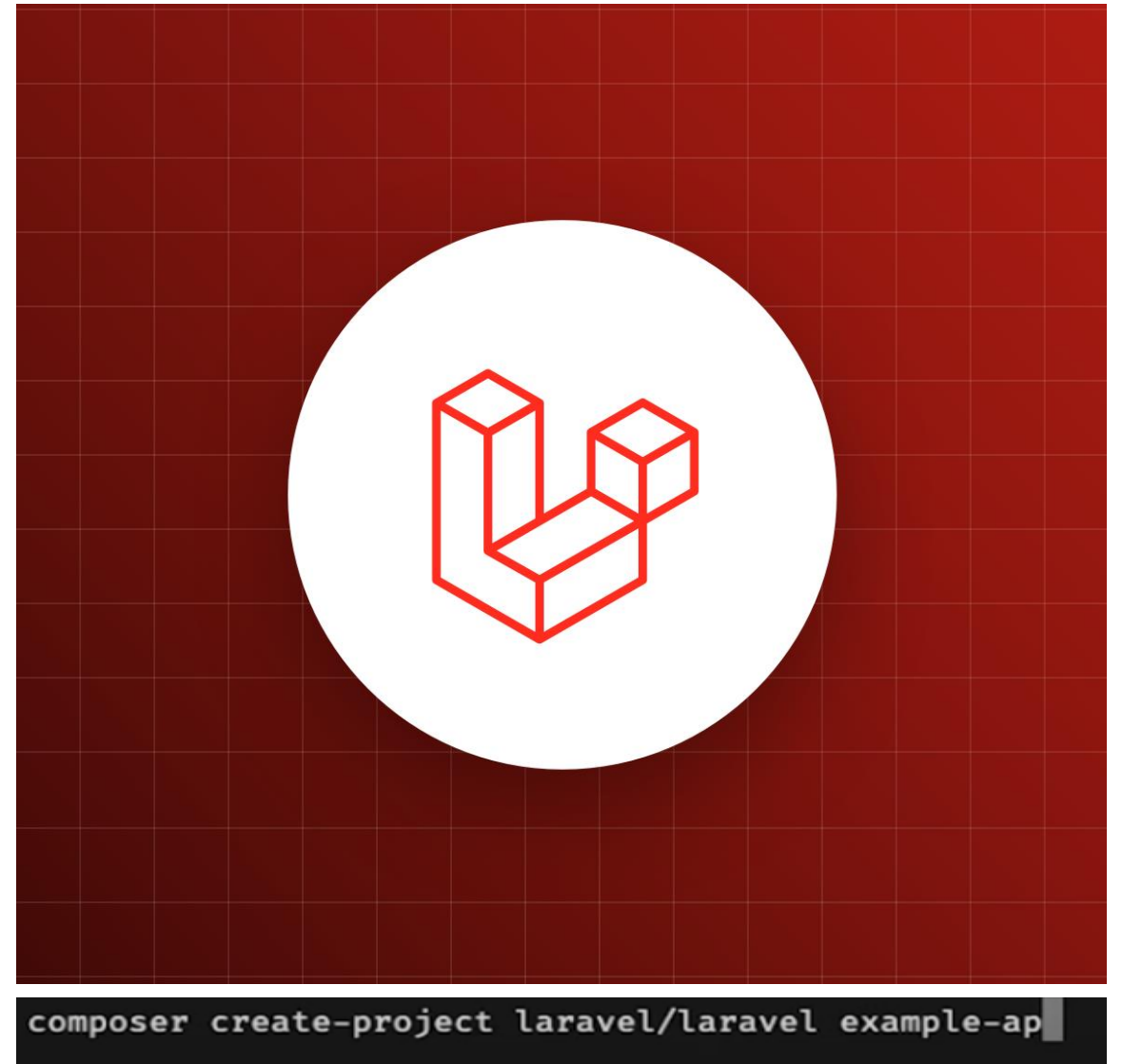
```
C:\xampp\htdocs>composer global require laravel/installer
Changed current directory to C:/Users/Usuario/AppData/Roaming/Composer
Using version ^4.2 for laravel/installer
./composer.json has been created
Running composer update laravel/installer
```

Ahora
crearemos
un nuevo
proyecto de
laravel

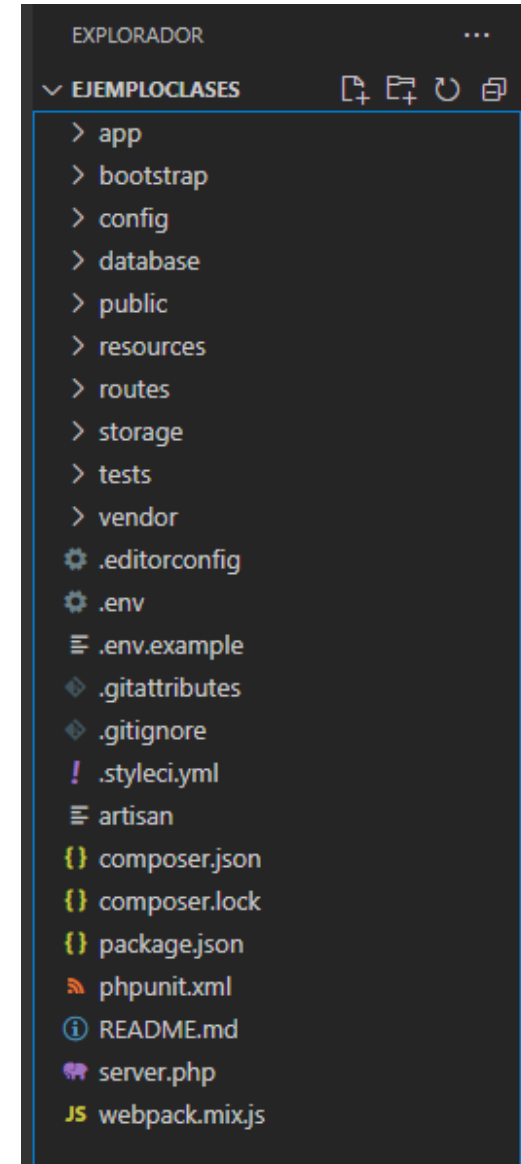


```
C:\xampp\htdocs>laravel new ejemploclases
```

Otra forma de
hacer los dos
pasos anteriores
en solo una línea
una vez creado el
proyecto iremos a
su carpeta



Como verás se
habrá creado
una gran lista
de archivos y
directorios



Comando basico
para levantar
nuestro servidor
laravel y la
pagina de
bienvenida



[Documentation](#)

Laravel has wonderful, thorough documentation covering every aspect of the framework. Whether you are new to the framework or have previous experience with Laravel, we recommend reading all of the documentation from beginning to end.



[Laracasts](#)

Laracasts offers thousands of video tutorials on Laravel, PHP, and JavaScript development. Check them out, see for yourself, and massively level up your development skills in the process.



[Laravel News](#)

Laravel News is a community driven portal and newsletter aggregating all of the latest and most important news in the Laravel ecosystem, including new package releases and tutorials.



[Vibrant Ecosystem](#)

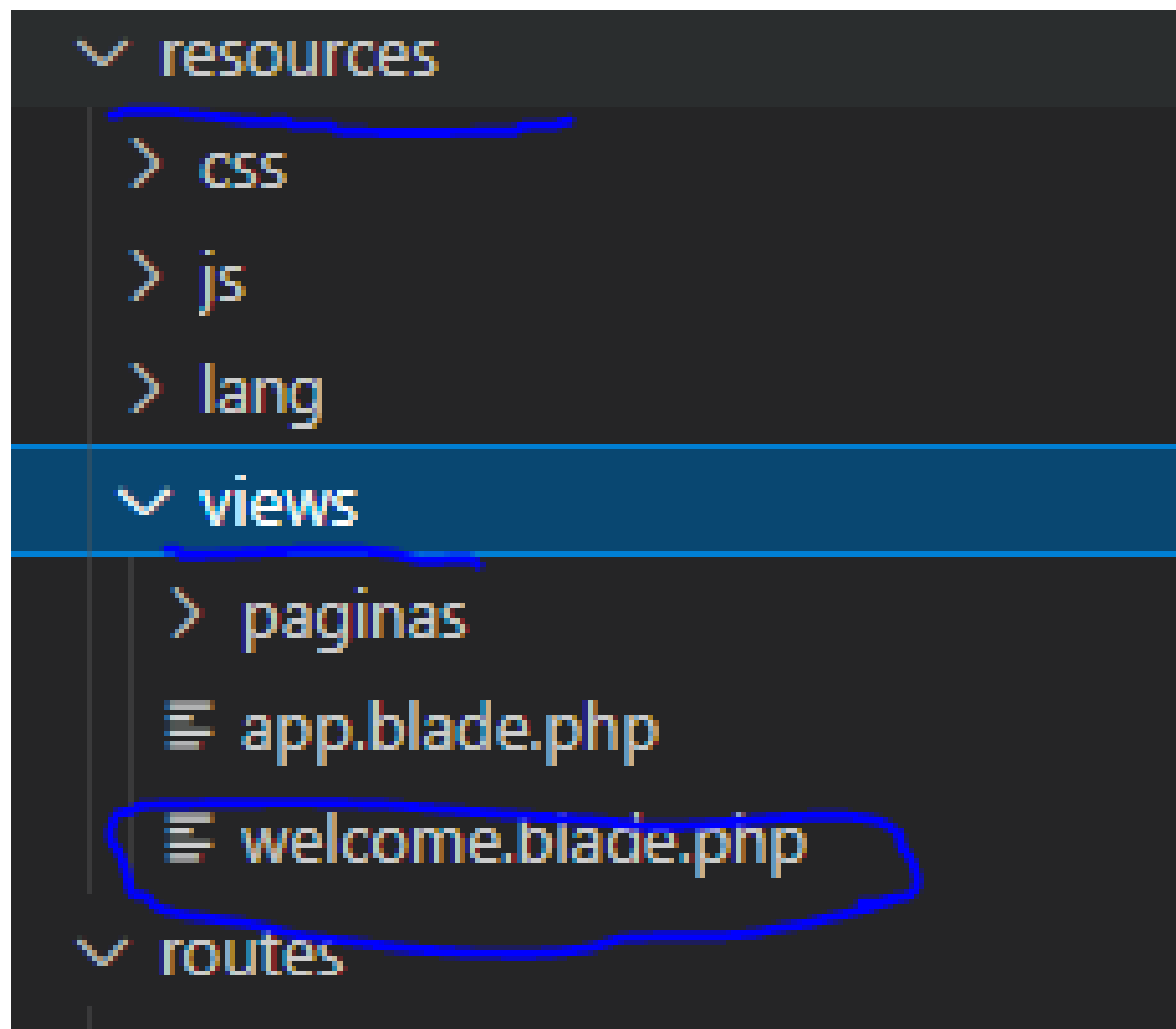
Laravel's robust library of first-party tools and libraries, such as [Forge](#), [Vapor](#), [Nova](#), and [Envoyer](#) help you take your projects to the next level. Pair them with powerful open source libraries like [Cashier](#), [Dusk](#), [Echo](#), [Horizon](#), [Sanctum](#), [Telescope](#), and more.

[Shop](#) [Sponsor](#)

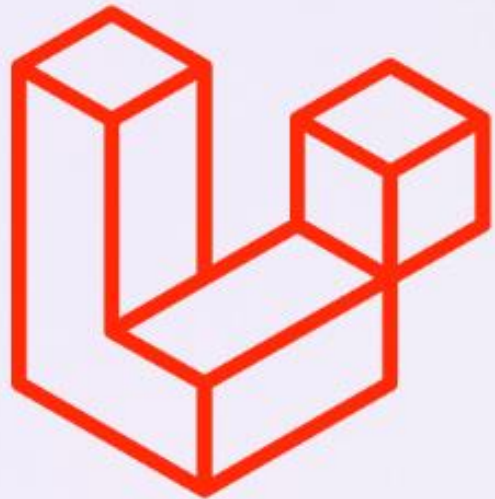
Laravel v8.83.2 (PHP v7.3.30)

```
C:\xampp\htdocs\ejemploclases>php artisan serve  
Starting Laravel development server: http://127.0.0.1:8000
```

La página
por defecto
estará
ubicada en:

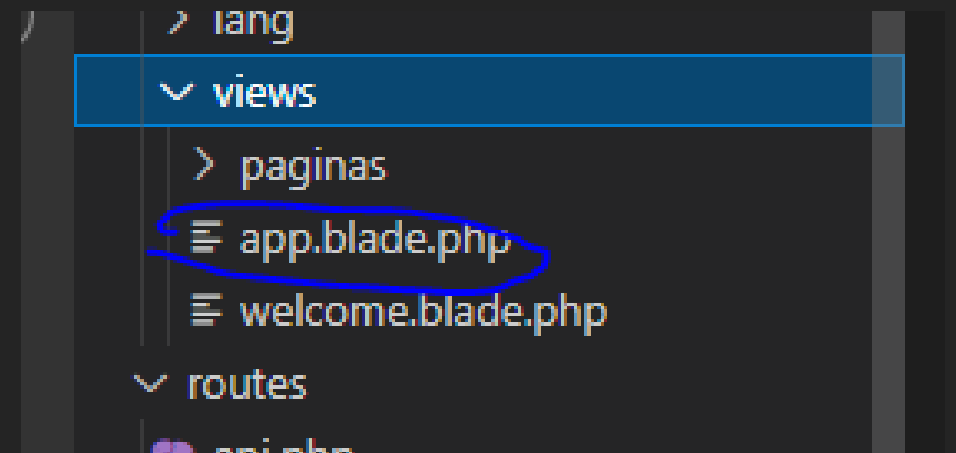


Blande php



Sintaxis
BLADE

En views creamos la que va a ser la plantilla que compartira parte de nuestro programa o nuestro programa



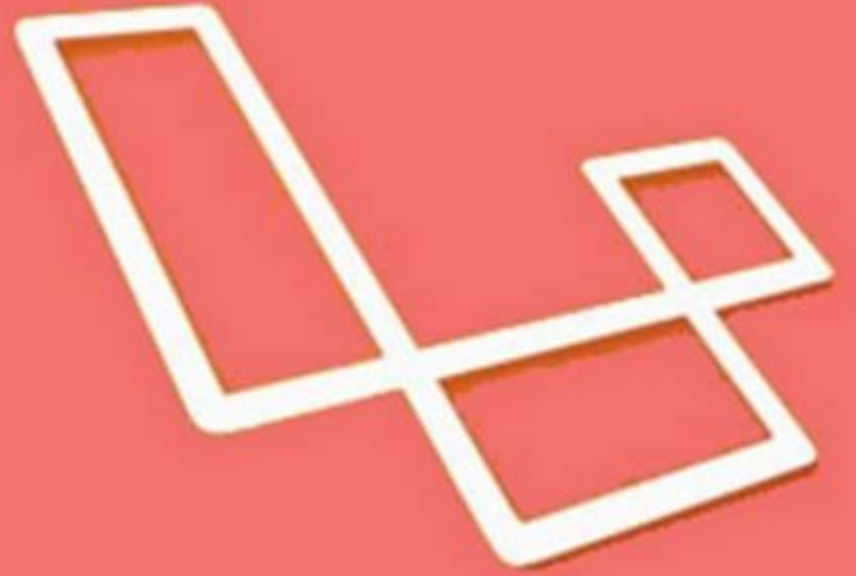
sources > views > app.blade.php > html > body

```
4 <div class="container-fluid">
5   <a class="navbar-brand" href="#">Inicio</a>
6   <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" ari
7     <span class="navbar-toggler-icon"></span>
8   </button>
9   <div class="collapse navbar-collapse" id="navbarNav">
10    <ul class="navbar-nav">
11      <li class="nav-item">
12        <a class="nav-link" href="#">Segunda pagina</a>
13      </li>
14      <li class="nav-item">
15        <a class="nav-link" href="#">Tercera pagina</a>
16      </li>
17    </ul>
18  </div>
19 </div>
20 </nav>
21 @yield('content')
22 </body>
23 </html>
```

Con esa directiva podremos
crear una plantilla para
futuras páginas que hereden
de esta

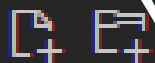
Laravel

Enrutamiento



BÁSICO

✓ EJEMPLOCLASES



> app

> bootstrap

> config

> database

> public

> resources

✓ routes

🐘 api.php

🐘 channels.php

🐘 console.php

🐘 web.php

> storage

> tests

> vendor

⚙ .editorconfig

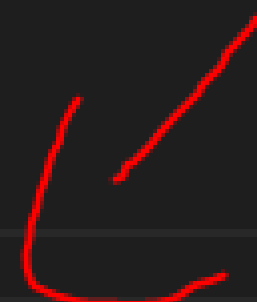
⚙ .env

≡ .env.example

Si nos vamos a
/routes/web.php
podremos crear el
enrutamiento de
nuestro sitio

Por defecto nos
encontraremos
lo siguiente

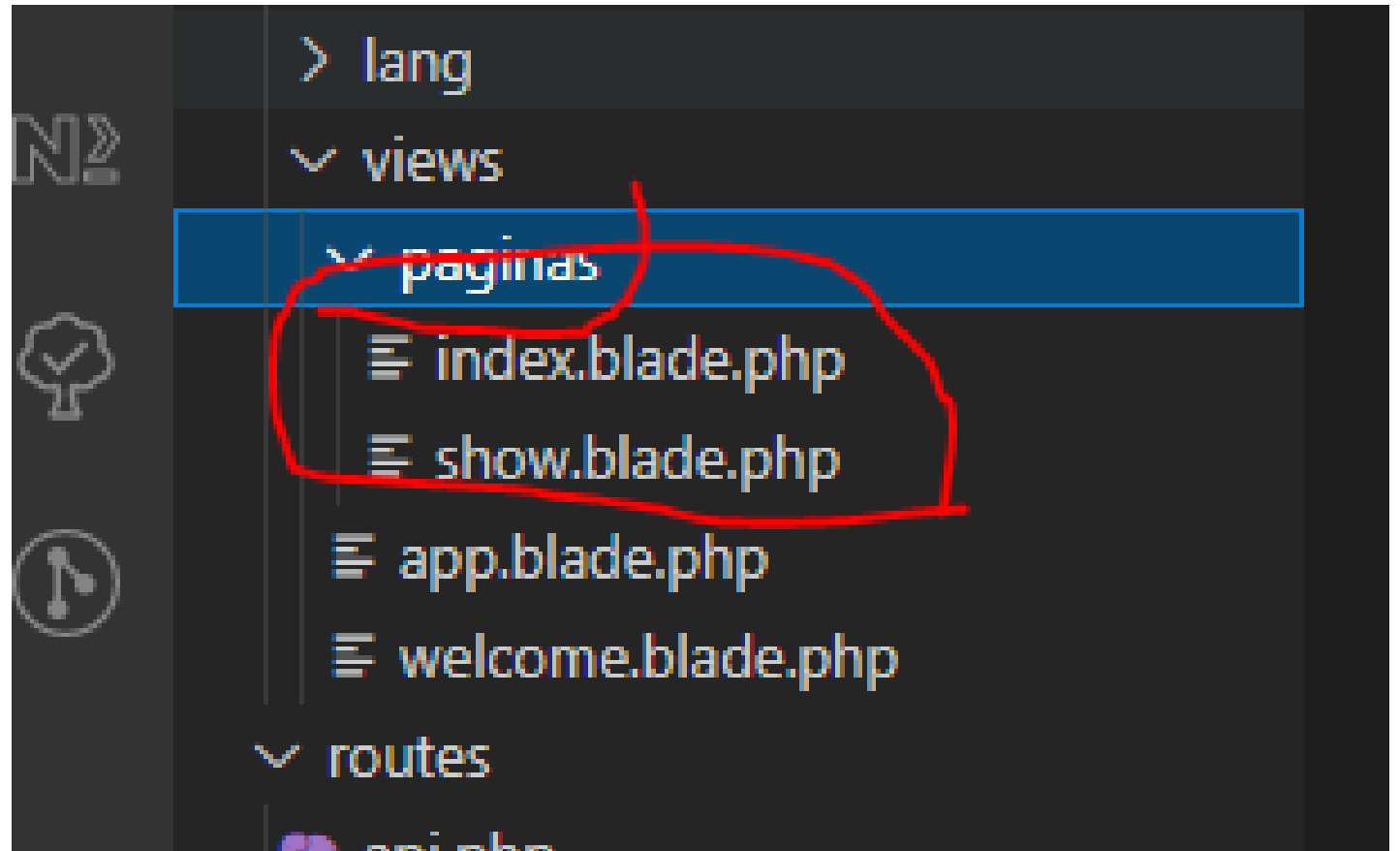
```
| routes are loaded by the RouteServiceProvider  
| contains the "web" middleware group. Now crea  
|  
*/  
  
Route::get('/', function () {  
    return view('welcome');  
});
```



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

Si queremos que por ejemplo nuestra plantilla sea la pagina principal lo editaremos cambiando el welcome por el nombre de nuestra plantilla blade en este caso app

Ahora vamos a
crear la carpeta
y los archivos
que heredaran
de app



Una vez dentro de nuestra pagina que hereda de app exportaremos y crearemos su contenido

```
utur  terminal  Ayuda
≡ app.blade.php  ≡ index.blade.php X  2022_02_24_18
resources > views > paginas > ≡ index.blade.php > ...
1  @extends('app')
2
3  @section('content')
4      _____
5  @endsection
6      <div class="container w-25 border p-4 mt
```

Codigo

Volvemos a
las rutas y la
cambiaremos
para que salga
la página hija

```
Archivo  Editar  Selección  Ver  Ir  Ejecutar  Terminal  Ayuda  web.php - ejemploclases - Visual Studio Code

EXPLORADOR
  EJEMPLOCASES
    > app
    > bootstrap
    > config
    > database
    > public
    > resources
      > css
      > js
      > lang
    > views
      > paginas
        index.blade.php
        show.blade.php
      app.blade.php
      welcome.blade.php
    routes
      api.php
      channels.php
      console.php
      web.php
    > storage
    > tests
    > vendor
    .editorconfig
    .env
    .env.example
    .gitattributes
    .gitignore
    .styleci.yml
    artisan

routes > web.php
6  |-----
7  | Web Routes
8  |-----
9  |
10 | Here is where you can register web routes for your application. These
11 | routes are loaded by the RouteServiceProvider within a group which
12 | contains the "web" middleware group. Now create something great!
13 |
14 | */
15 |
16 | Route::get('/auxi', function () {
17 |     return view('welcome');
18 | });
19 |
20 | Route::get('/', function () {
21 |     return view('paginas.index');
22 | })->name('acceso');
23 |
24 |
25 | Route::get('/', [UsuariosController::class, 'index'])->name('acceso');
26 |
27 | Route::post('/', [UsuariosController::class, 'store'])->name('acceso');
28 |
29 |
30 | Route::get('/{id}', [UsuariosController::class, 'show'])->name('acceso-edit');
31 | Route::patch('/{id}', [UsuariosController::class, 'update'])->name('acceso-update');
32 | Route::delete('/{id}', [UsuariosController::class, 'destroy'])->name('acceso-destroy');
33 |
```

Ahora crearemos
un formulario con
un input y
un submit en mi
caso utilice
bootstrap

```
@extends('app')

@section('content')

    <div class="container w-25 border p-4 mt-4">
        <form action="{{ route('acceso') }}" method="POST">
            @csrf <!--toquen para seguridad de laravel-->
            @if (session('success'))
                <h6 class="alert alert-success">{{ session('success') }}</h6>
            @endif

            @error('nombre')
                <h6 class="alert alert-danger">{{ $message }}</h6>
            @enderror

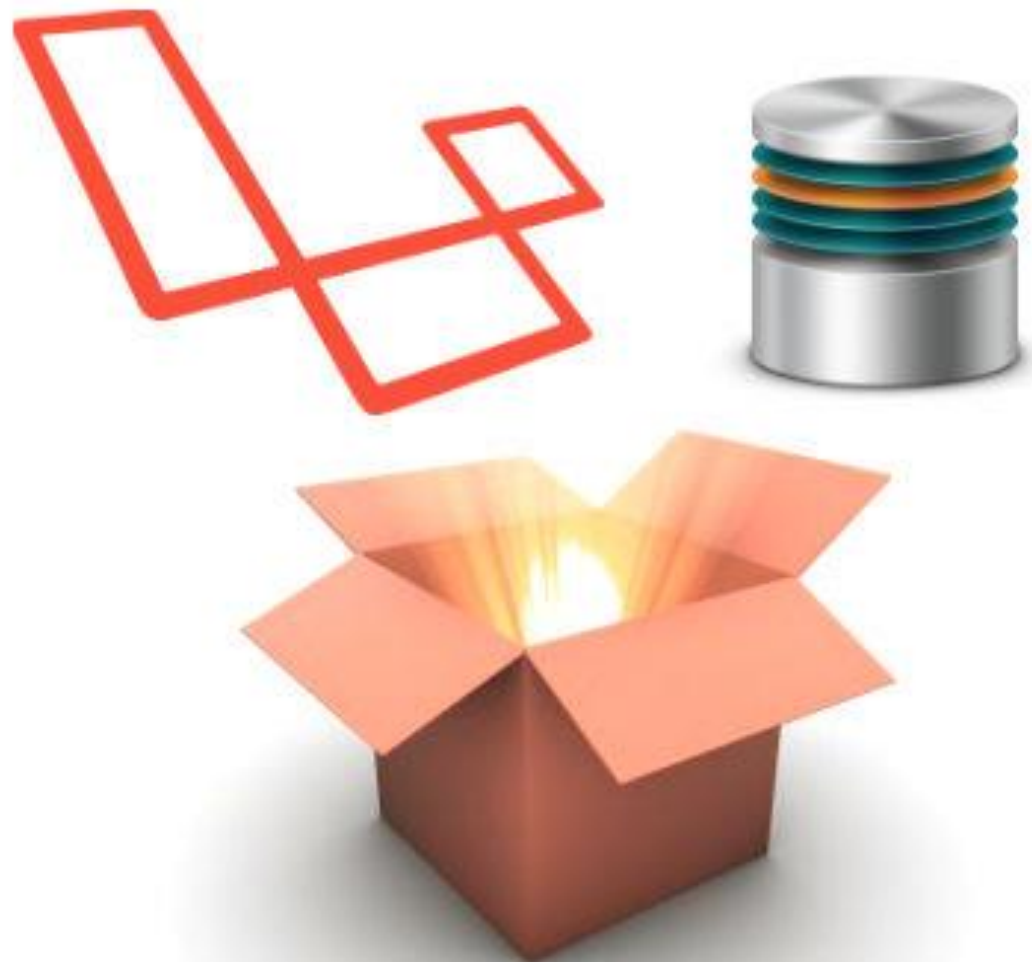
            <div class="mb-3">
                <label for="name" class="form-label">Usuario</label>
                <input type="text" name="nombre" class="form-control">
                <div id="emailHelp" class="form-text">No reveles tu nombre de usuario a nadie.</div>
            </div>

            <button type="submit" class="btn btn-primary">Enviar</button>
        </form>
    </div>
```



No le hagan caso a los métodos
los explicaré más adelante

Migraciones

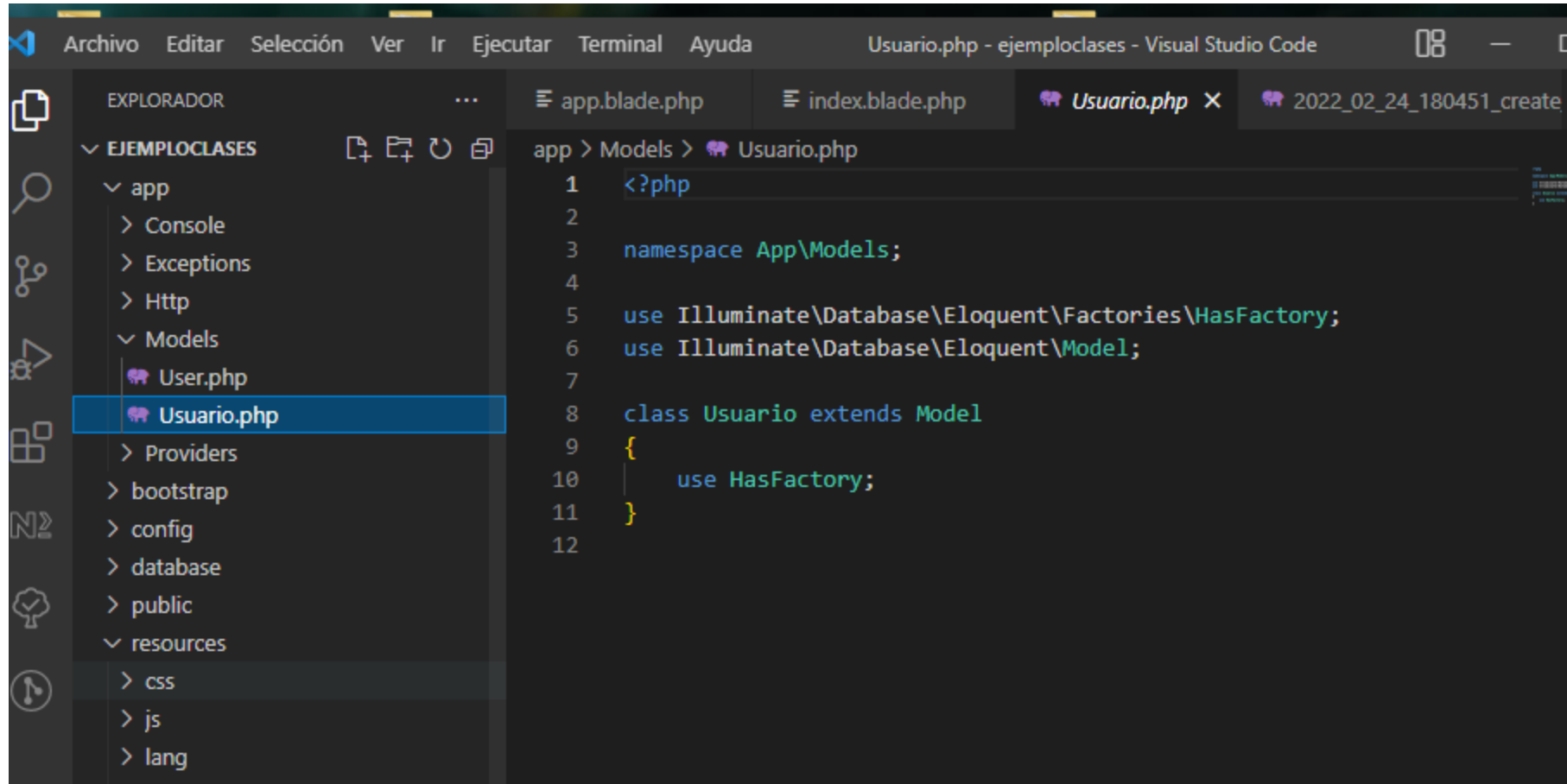


```
C:\xampp\htdocs\ejemploclases>php artisan make:model Usuario -m
Model created successfully.
Created Migration: 2022_02_24_180451_create_usuarios_table

C:\xampp\htdocs\ejemploclases>
```

Vamos a crear una tabla con artisan

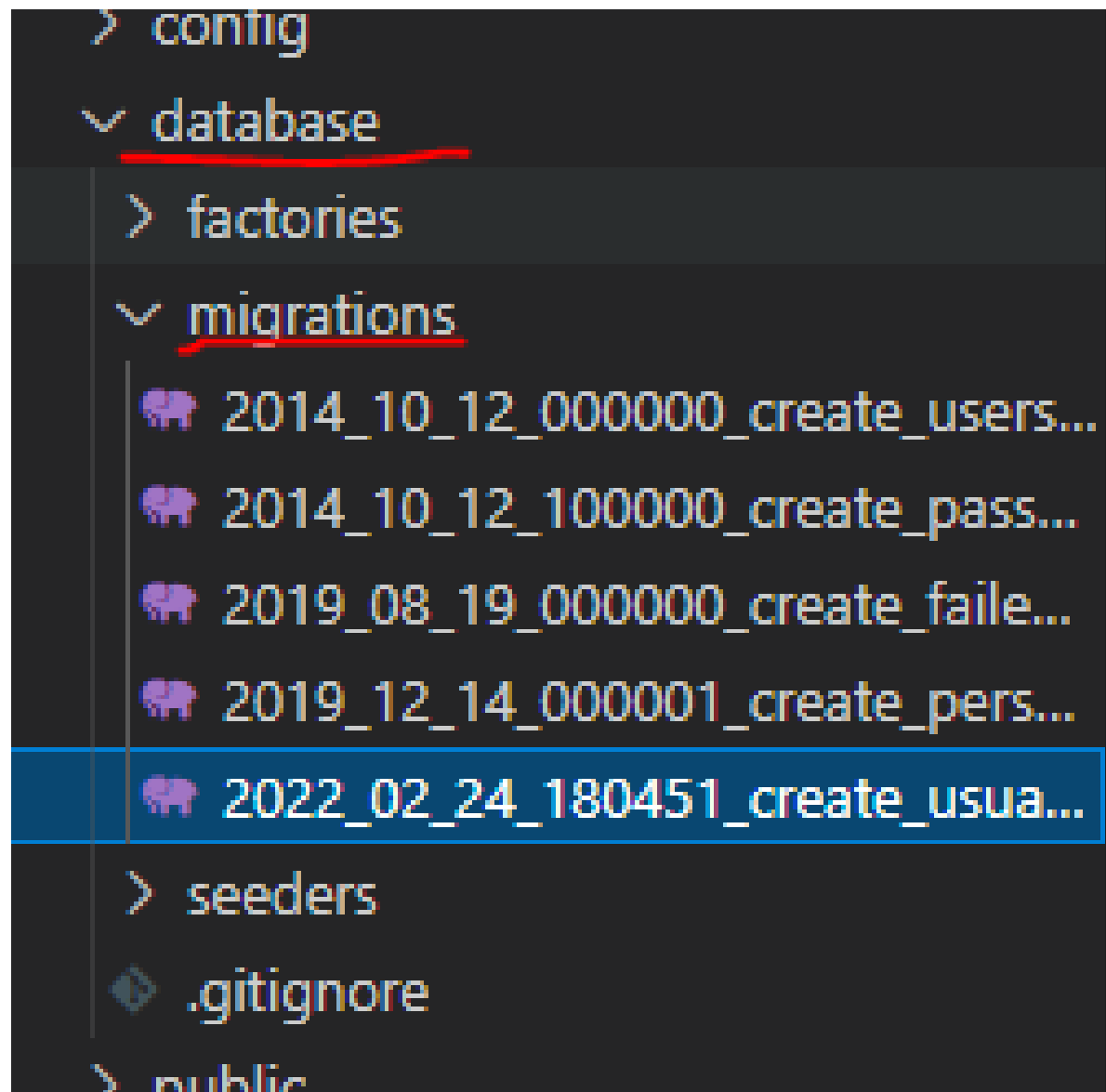
Nos acaba de crear nuestro modelo para eloquent



The screenshot shows the Visual Studio Code interface with the Explorer panel on the left and the Editor panel on the right. The Explorer panel shows the file structure of a project named 'EJEMPLOCLASES'. The 'app' directory is expanded, showing subdirectories like 'Console', 'Exceptions', 'Http', 'Models', 'Providers', 'bootstrap', 'config', 'database', 'public', and 'resources'. The 'Models' directory is further expanded, showing 'User.php' and 'Usuario.php'. The 'Usuario.php' file is selected and highlighted. The Editor panel shows the code for 'Usuario.php', which is a PHP class extending 'Illuminate\Database\Eloquent\Model' and using 'Illuminate\Database\Eloquent\Factories\HasFactory'.

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

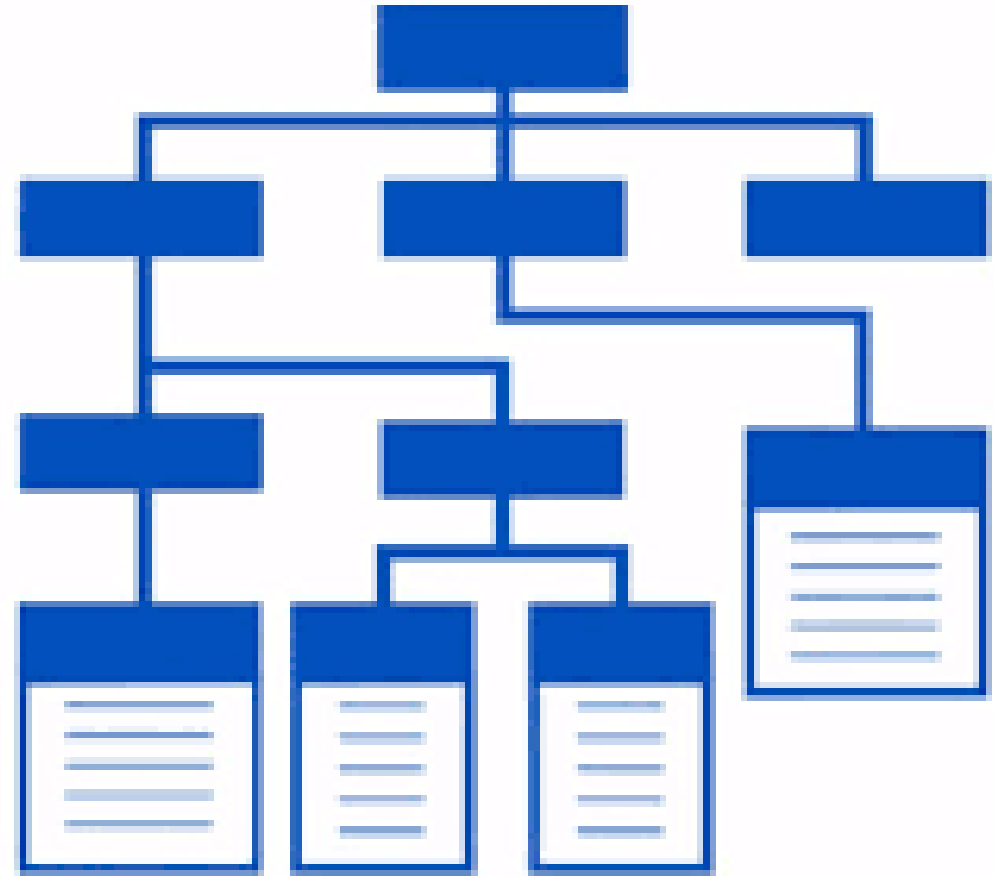
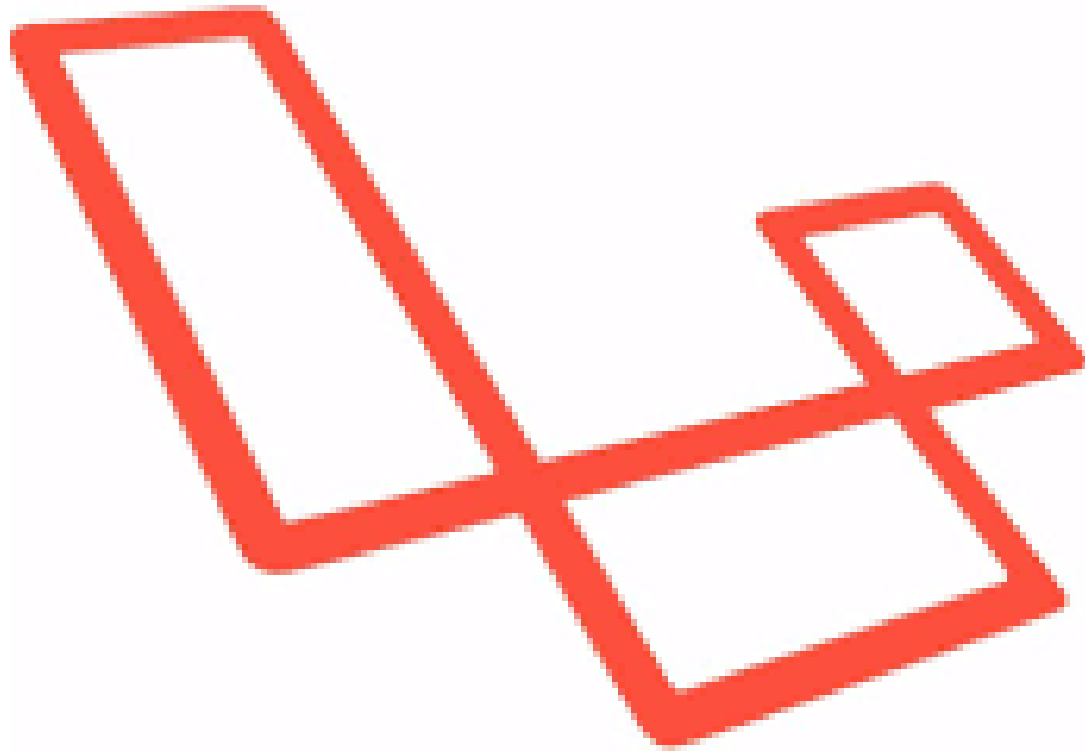
Para crear nuestra tabla iremos a Databases\migations y por defecto el archivo se habrá generado en el ultimo lugar, los demas archivos son ejemplos de laravel



Dentro
crearemos
nuestra tabla,
la estructura ya
estará creada

```
database > migrations > 2022_02_24_180451_create_usuarios_table.php
6
7 class CreateUsuariosTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('usuarios', function (Blueprint $table) {
17             $table->id();
18             $table->string('nombre');
19             $table->timestamps();
20         });
21     }
22
23     /**
24      * Reverse the migrations.
25      *
26      * @return void
27      */
28     public function down()
29     {
30         Schema::dropIfExists('usuarios');
31     }
32 }
33
```

Dentro hay dos funciones a destacar, up para crear la tabla y down para eliminarla

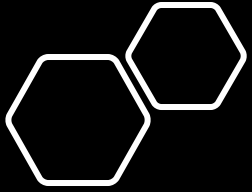


Ahora desde el xampp creamos la base de datos que vayamos a usar

```
MariaDB [(none)]> create database usuarios;
```

- > views
- > routes
- > storage
- > tests
- > vendor
- ⚙ .editorconfig
- ⚙ .env
- ≡ .env.example
- 💎 .gitattributes
- 💎 .gitignore
- ! .styleci.yml
- ≡ artisan
- ⚡ composer.json

Ahora
conectaremos
Laravel con nuestra
base de datos



Lo importante de este fichero es ponerle la base de datos a utilizar, es lo equivalente al new mysqli de php nativo

```
0
1 DB_CONNECTION=mysql
2 DB_HOST=127.0.0.1
3 DB_PORT=3306
4 DB_DATABASE=usuario
5 DB_USERNAME=root
6 DB_PASSWORD=
7
```


Ahora con migrate
crearemos la tabla en la
base de datos: fresh es
para eliminar la tabla
por si has hecho algo
mal y si no lo pones
pues creas la tabla

```
Git CMD
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (23.49ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (25.75ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (20.99ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated: 2019_12_14_000001_create_personal_access_tokens_table (23.10ms)
Migrating: 2022_02_24_180451_create_usuarios_table
Migrated: 2022_02_24_180451_create_usuarios_table (11.74ms)
C:\xampp\htdocs\ejemploclases>php artisan migrate:fresh
Dropped all tables successfully.
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (23.31ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (21.15ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (17.70ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated: 2019_12_14_000001_create_personal_access_tokens_table (23.91ms)
Migrating: 2022_02_24_180451_create_usuarios_table
Migrated: 2022_02_24_180451_create_usuarios_table (11.18ms)
C:\xampp\htdocs\ejemploclases>php artisan migrate
Nothing to migrate.
C:\xampp\htdocs\ejemploclases>
```

server.php PROBLEMAS SALIDA TERMINAL CONSOLA DE DEPURACIÓN powershell

Ahora si vamos a la base de datos podemos ver como se ha creado la tabla

```
MariaDB [(none)]> use usuario;  
Database changed  
MariaDB [usuario]> describe usuario;  
ERROR 1146 (42S02): Table 'usuario.usuario' doesn't exist  
MariaDB [usuario]> show tables;  
+-----+  
| Tables_in_usuario |  
+-----+  
| failed_jobs  
| migrations  
| password_resets  
| personal_access_tokens  
| users  
| usuarios  
+-----+  
6 rows in set (0.000 sec)  
MariaDB [usuario]> describe usuarios;  
+-----+-----+-----+-----+-----+-----+  
| Field | Type | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| id | bigint(20) unsigned | NO | PRI | NULL | auto_increment |  
| nombre | varchar(255) | NO | | NULL | |  
| created_at | timestamp | YES | | NULL | |  
| updated_at | timestamp | YES | | NULL | |  
+-----+-----+-----+-----+-----+-----+  
4 rows in set (0.007 sec)
```



Si la has liado
con algo en la
última
migracion

En vez de hacer un fresh
tambien puedes hacer un
rollback con php artisan
migrate:rollback

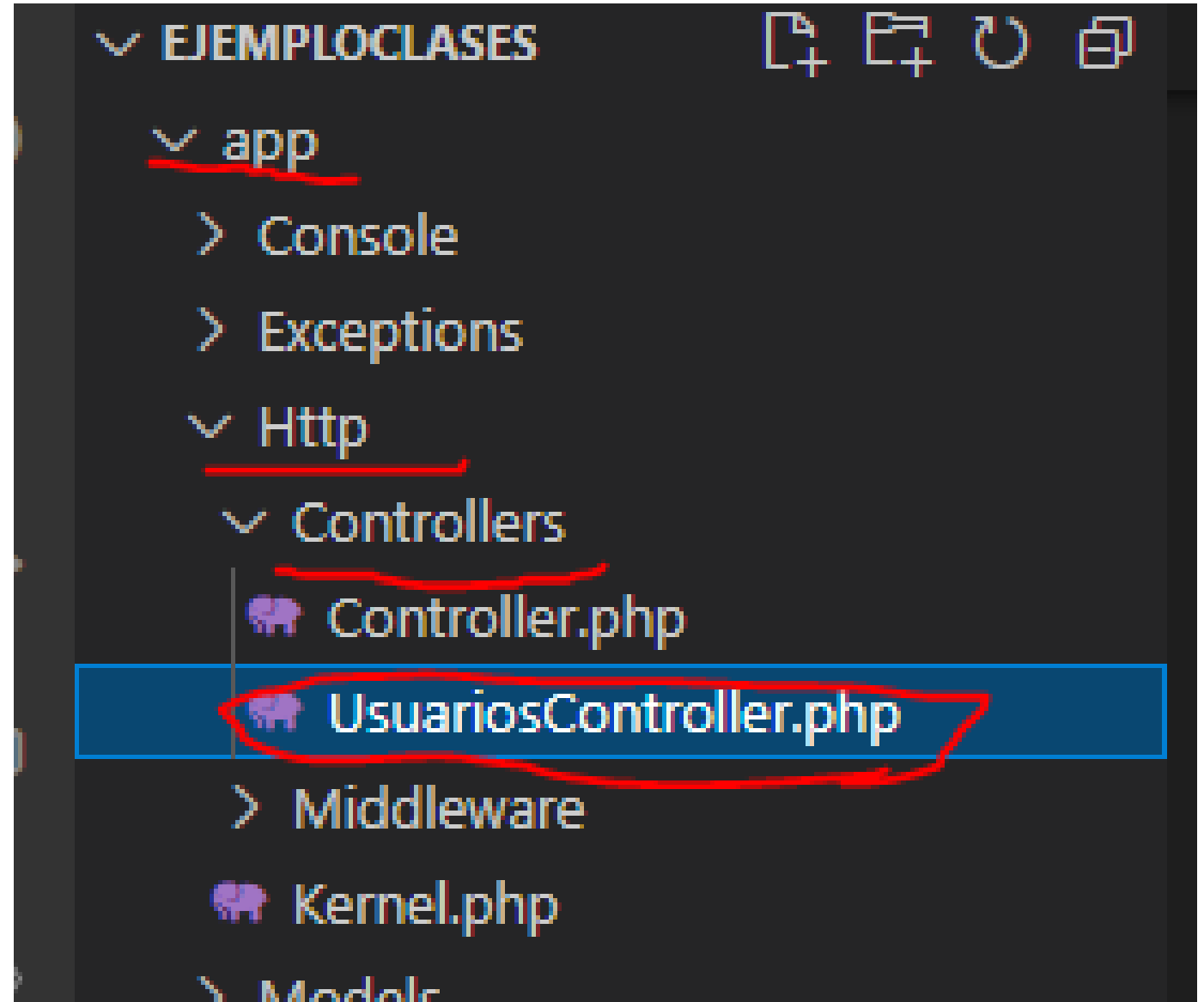

```
C:\xampp\htdocs\ejemploclases>php artisan migrate  
Nothing to migrate.
```

```
C:\xampp\htdocs\ejemploclases>php artisan make:controller UsuariosController  
Controller created successfully.
```

```
C:\xampp\htdocs\ejemploclases>
```

Ahora crearemos un controlador

Su ruta será
la siguiente



Ahora vamos a crear
nuestros metodos de
insercion en la base de
datos, editar etc...
Por lo general en laravel
se suelen utilizar estos
nombres

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\Models\Usuario;
7
8  class UsuariosController extends Controller
9  {
10     /*
11     * index para mostrar todos los usuarios
12     * store para guardar un usuario
13     * update para actualizar un usuario
14     * destroy para eliminar un usuario
15     * edit para mostrar el formulario de edicion
16     */
17
```

Creamos nuestro método de inserción

```
* store para guardar un usuario
* update para actualizar un usuario
* destroy para eliminar un usuario
* edit para mostrar el formulario de edicion
*/

public function store(Request $request){
    $request->validate([
        'nombre' => 'required|min:3'
    ]);
    $usuario = new Usuario;
    $usuario->nombre = $request->nombre;
    $usuario->save();
    return redirect()->route('acceso')->with('success', 'Usuario insertado correctamente' );
}
```

Nombre entrante

Validar usuario

Lo creamos

Importante hay que poner el modelo que estamos utilizando.

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\Models\Usuario;
7
8  class UsuariosController extends Controller
```

```
$usuario->save();  
return redirect()->route('acceso')->with('success', 'Usuario insertado correctamente' );  
}  
public function index(){  
    $usuarios = Usuario::all();
```

Mandaremos al apartado de rutas el mensaje de usuario creado

Creamos la ruta
para la inserción

- ->name() es para darle un alias de esta forma si cambiamos la ruta siempre tendra el mismo nombre para el uso en el html

```
Route::post('/', [UsuariosController::class, 'store'])->name('acceso');
```

Para conectar nuestro formulario con la
ruta insertaremos la ruta en el action

```
class="container w-25 border p-4 mt-4">
<form action="{{ route('acceso') }}" method="POST">
    @csrf <!--toquen para seguridad de laravel-->
    @if (session('success'))
        <h6 class="alert alert-success">{{ session('succ
```


Por motivos de seguridad laravel no te dejara insetar desde el formulario sin la siguiente directiva

- Dicho de forma resumida esta directiva sirve para darle una id a una sesion para evitar ataques de spam

```
orm action="{{ route('acceso') }}" method="POST">  
  @csrf <!--toquen para seguridad de laravel-->
```

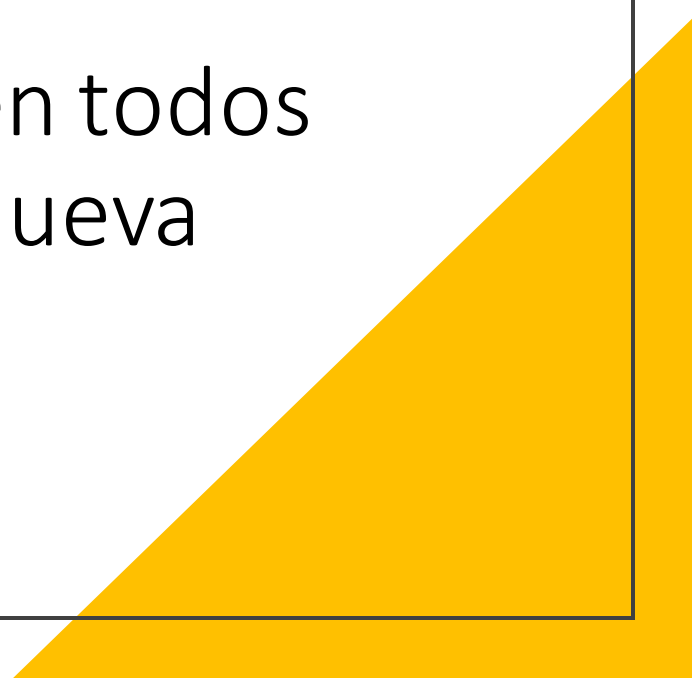
Ahora si comprobamos
nuestra base de datos
los datos deberían
haberse insertado

```
@if (session('success'))  
|   <h6 class="alert alert-success">{{ session('success') }}</h6>  
@endif  
  
@error('nombre')  
<h6 class="alert alert-danger">{{ $message }}</h6>  
@enderror
```

El siguiente if funcionara para saber si el usuario ha sido creado correctamente

```
public function index(){  
    $usuarios = Usuario::all();  
    return view('paginas.index', ['usuarios' => $usuarios]);  
}
```

Ahora vamos a hacer que se muestren todos los valores para ello crearemos una nueva funcion en el controlador



Como anteriormente
iremos a rutas y
crearemos una ruta para
lo que vayamos a mostrar



```
Route::post('/', [UsuariosController::class, 'store'])->name('acceso');
```

Ahora crearemos un foreach para mostrar todos los campos, aquí solo nos fijaremos en el segundo div en la parte de \$usuario->nombre que lo que hara será mostrar el nombre de usuario

```
@foreach ($usuarios as $usuario)
<?php
$counter++;

?>
<div class="row py-1">
  <div class="col-md-9 d-flex align-items-center"><!--Metodo para editar por clave valor-->
    <a href="{{ route('acceso-edit', ['id' => $usuario->id]) }}">{{ $usuario->nombre }}</a>
  </div>
  <div class="col-md-3 d-flex justify-content-end"><!--Metodo para borrar por solamente el valor-->
    <form action="{{ route('acceso-destroy', [$usuario->id]) }}" method="POST">
      @method('DELETE')<!--Aunque no exista en html con este metodo el formulario sera de tipo delete en vez de post-->
      @csrf
      <button class="btn btn-danger btn-sm">Eliminar</button>
    </form>
  </div>
</div>
@endforeach
<h6>El numero total de usuarios es: {{ $counter }}</h6>
</div>
```

Ahora voy a
mostrar las
rutas y
controladores
faltantes del
crud

```
$usuario = new Usuario;
$usuario->nombre = $request->nombre;
$usuario->save();
return redirect()->route('acceso')->with('success', 'Usuario insertado correctamente' );
}

public function index(){
    $usuarios = Usuario::all();
    return view('paginas.index', ['usuarios' => $usuarios]);
}

public function show($id){
    $usuario = Usuario::find($id);
    return view('paginas.show', ['usuario' => $usuario]);
}

public function update(Request $request, $id){
    $usuario = Usuario::find($id);
    $usuario->nombre = $request->nombre;
    $usuario->save();
    //dd(); importante! es para hacer un log

    // return view('paginas.index', ['success' => 'Usuario actualizado!']);
    return redirect()->route('acceso')->with('success', 'Usuario actualizado!');
}

public function destroy($id){
    $usuario = Usuario::find($id);
    $usuario->delete();

    return redirect()->route('acceso')->with('success', 'Usuario ha sido borrado!');
}
```

```
14  */
15
16  ✓ Route::get('/auxi', function () {
17      |     return view('welcome');
18  });
19
20  ✓ Route::get('/', function () {
21      |     return view('paginas.index');
22  }->name('acceso');
23
24
25  Route::get('/', [UsuariosController::class, 'index'])->name('acceso');
26
27  Route::post('/', [UsuariosController::class, 'store'])->name('acceso');|
28
29
30  Route::get('/{id}', [UsuariosController::class, 'show'])->name('acceso-edit');
31  Route::patch('/{id}', [UsuariosController::class, 'update'])->name('acceso-update');
32  Route::delete('/{id}', [UsuariosController::class, 'destroy'])->name('acceso-destroy');
33
```



```
$counter++;
```

```
?>
```

```
<div class="row py-1">
```

```
<div class="col-md-9 d-flex align-items-center"><!--Metodo para editar por clave valor-->
```

```
<a href="{{ route('acceso-edit', ['id' => $usuario->id]) }}">{{ $usuario->nombre }}</a>
```

```
</div>
```

```
<div class="col-md-3 d-flex justify-content-end"><!--Metodo para borrar por solamente el valor-->
```

```
<form action="{{ route('acceso-destroy', [$usuario->id]) }}" method="POST">
```

```
@method('DELETE')<!--Aunque no exista en html con este metodo el formulario sera de tipo delete en vez de post-->
```

```
@csrf
```

```
<button class="btn btn-danger btn-sm">Eliminar</button>
```

```
</form>
```

```
</div>
```

```
</div>
```

```
@endforeach
```

```
<h6>El numero total de usuarios es: {{ $counter }}</h6>
```

```
</div>
```

```
</div>
```

```
tion
```

Esta es otra
pagina que
es donde
actualizo los
datos

```
@extends('app')

@section('content')

<div class="container w-25 border p-4 mt-4">
  <form action="{{ route('acceso-update', ['id'=> $usuario->id]) }}" method="POST">
    @method('PATCH')
    @csrf <!--toquen para seguridad de laravel-->
    @if (session('success'))
      <h6 class="alert alert-success">{{ session('success') }}</h6>
    @endif

    @error('nombre')
      <h6 class="alert alert-danger">{{ $message }}</h6>
    @enderror

    <div class="mb-3">
      <label for="name" class="form-label">Usuario</label>
      <input type="text" name="nombre" class="form-control" value="{{ $usuario->nombre }}">
      <div id="emailHelp" class="form-text">No reveles tu nombre de usuario a nadie.</div>
    </div>

    <button type="submit" class="btn btn-primary">actualizar</button>
  </form>
</div>

@endsection
```

Y al final nuestra pagina debería quedar de este modo

Usuario

No reveles tu nombre de usuario a nadie.

Enviar

[alcachofa](#)

Eliminar

[alejandro](#)

Eliminar

[Pepe](#)

Eliminar

El numero total de usuarios es: 3