**Creación rápidamente Proyecto Laravel**

Option 1. composer create-project laravel/laravel example-app

Option 2. composer global require laravel/installer

laravel new example-app

**Herramientas FrontEnd y Backend**

**FrontEnd utiliza npm**

**Backend utiliza composer**

**Php artisan optimize:clear para realizar limpieza de un proyecto en producción**

**Php artisan tinker. Para interactuar directamente con la interfaz de nuestro sistema (depurar la aplicacion)**

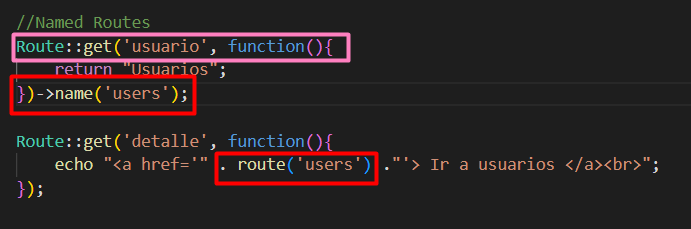
**dd(); detiene la ejecución**

**dump(); no detiene la ejecución**

**Laravel**

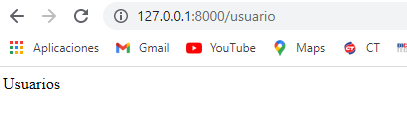
**Helpers son funciones propias de laravel que podemos usar en la documentación**

**Qué son y cómo se utilizan las RUTAS.**

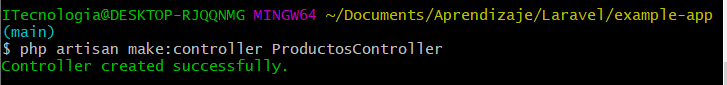


**El name de la ruta se puede acceder desde nuestro html con la función route. Pero en el navegador la url que se mostrara será usuario.**

[**https://laravel.com/docs/9.x/routing**](https://laravel.com/docs/9.x/routing)



**Rutas con controlador**





**Driver**

**Ui**

**Composer require laravel/ui para crear o facilitar el manejo de interfaces**

**Migraciones**

**Php artisan migrate:rollback para desacer el ultimo cambio en alguna tabla**

**Php artisan migrate:rollback –step=1 solamente borra el lote q este el ultimo**

**Php artisan migrate:fresh quita absolutamente todas las tablas destructivo.**

**Agregar o quitar campos sin afectar los datos**

* **Se crea la migración**
* Php artisan make:migration add\_phone**\_to\_**users\_table **automáticamente detecta laravel que se va agregar un nuevo campo**
* En español php artisan make:migration add\_phone\_users\_table **–table=users (Para editar algún campo de esa table)**

**SEEDERS para llenar los catalogos de la base de datos**

**Se crea el modelo primero, después se crea un seeder por catalogo**

php artisan make:seeder MateriaSeeder

Y se manda la información a la bd con

php artisan db:seed --class=MateriaSeeder

**CREACION DE MODELOS CONTROLADOR SEDDER**

**Php artisan make:model Materia –c –s –r**

**Php artisan make:model Materia –-controller –-seeder --resource**

**BLADE**

**Imprimir html en blade**

{!! $html !!}

Pasar data de controlador a template

$Product = Product::findOrFail($product);

Return view(‘products.show’)->with([

‘product’ => $product,

]);

With podemos enviar data en forma de array a la vista

De blade.

**Pusher web Sockets Tiempo real Proveedor**

Creando una cuenta en pusher

Crear una app Chanels

Agregamos los datos necesarios y le ponemos que queremos credenciales para ambientes producion, desarrollo etc

Y a continuacion nos genera las app keys que nos servirán para realizar la conexión a nuestro proveedor de pusher

Nos vamos a nuestro proyecto en laravel y configuramos los .env

Broadcast driver.

Y lo de pusher id,key,secret,cluster

Paqueteria para la instalación de pusher

Composer require pusher/pusher-php-server

Ahora en app en app.php

Habilitamos el provider para los canales de comunicación

BroadcastServiceProvider lo habilitamos

Con esto ya tenemos la comunicación entre laravel y pusher (backend)

Ahora por la parte del frontend

**Emplearemos Laravel ECHO para mostrar los datos en el front**

Instalamos las dependencias de laravel-echo y pusher.js

Con npm install –save-dev laravel-echo pusher-js

Ahora configuramos el archivo de bootsatrap.js dentro de resources, js, y descomentamos todo lo relacionado a laravel-echo

Y ejecutamos

Npm run dev

Siempre que agamos un cambio en el archivo js para que la aplicación lo tome en la platilla principal

Volvemos a cargar todo el proyecto desde el servicio apache

1. **Eventos**

Los eventos se crean para transmitir o realizar una operación en el frontend

Php artisan make:event PresionChanged

Se configura el evento

Y los argumentos que se van a utilizar para la transmisión de datos

Se especifica el canal que va manejar o la frecuencia en que va transmitir la informacion

**Los listeners** se encargan de escuchar cuando exista un cambio en la base de datos y dispara el evento

Php artisan make:listener BroadcastPresionesAdd

Se configura el listener

Y EN PROVIDERS EVENT SERVICE PROVIDER AGREGAMOS EL LISTENER

**Login**

**Emplearemos Composer require laravel/ui para crear o facilitar el manejo de interfaces**

php artisan ui bootstrap –auth