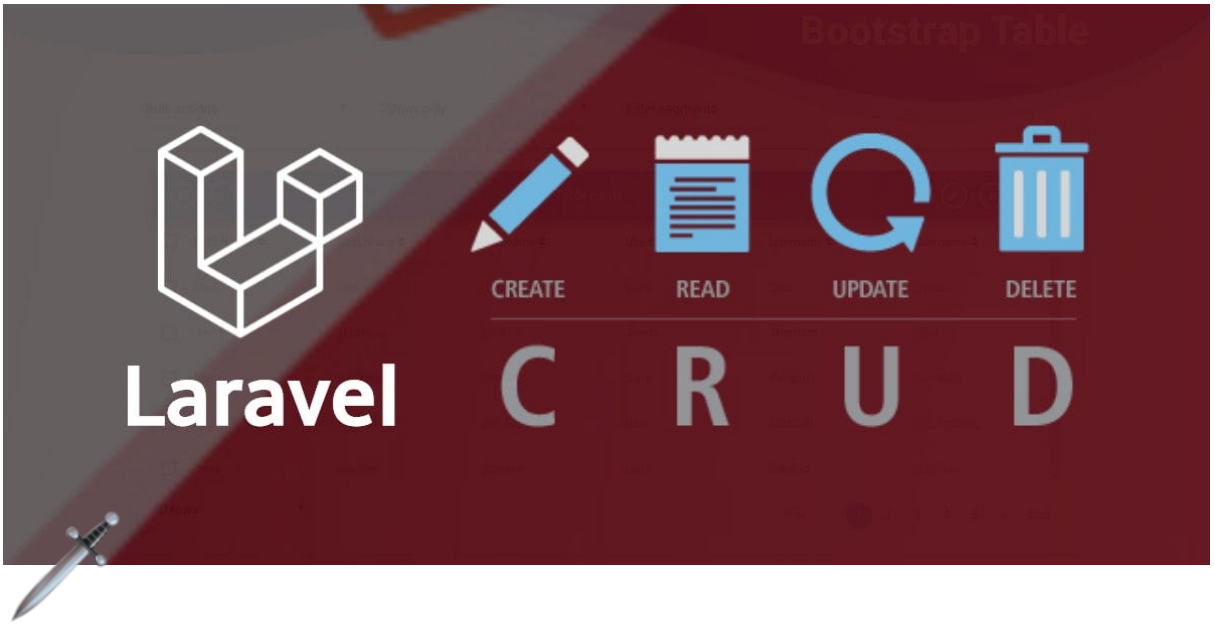


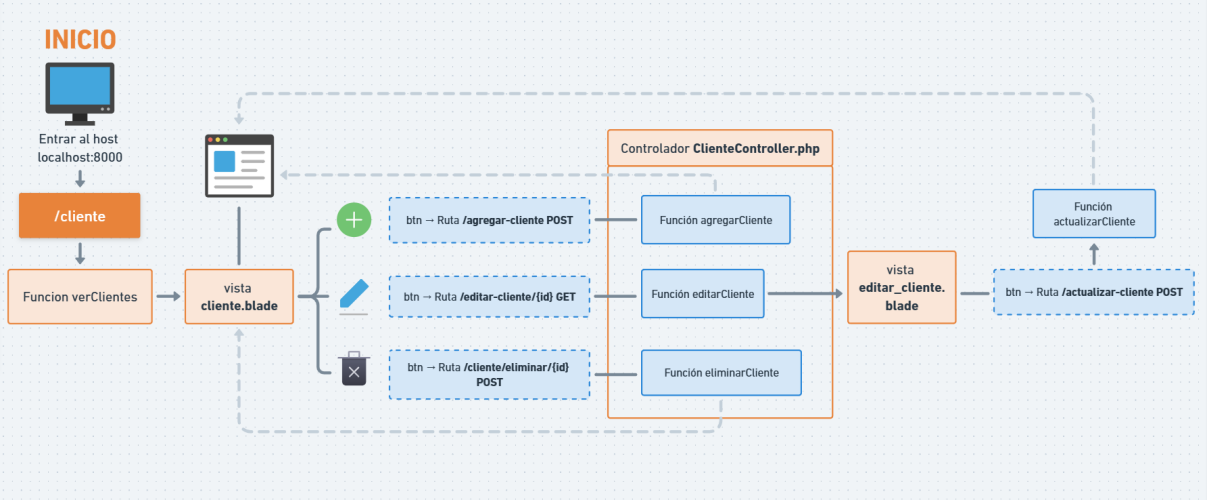
LARAVEL III - CRUD Básico

Titular	MARIA JOSE SALINAS PARRA
Etiquetas	Laravel
Autor	Henrry Gutiérrez, Maho Salinas

> **Hola!** 🦕 Esperamos que esta guía te sirva para hacer un CRUD básico con Laravel, te prometemos que lo aprenderás fácilmente. No somos profesionales, así que si cometimos algún error, por favor, háznoslo saber. 🐥 Por cierto, si quieres ayudarnos a mejorar la guía lo agradeceríamos muchísimo ✨ - *Maho y Henrry*

ÍNDICE

CRUD básico CLIENTES



1. Crear el modelo

Se usó de ejemplo la tabla de clientes. Así que se crea un modelo llamado Producto.php la cual tiene como clase padre Model. Digamos que se deben crear tantos modelos como tablas tengas, aunque no siempre se aplica.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Cliente extends Model
{
    use HasFactory;

    // DEFINIMOS LA TABLA A LA QUE HACE REFERENCIA EL MODELO
```

```
    protected $table = 'clientes';


    // DEFINIMOS LOS CAMPOS QUE SE PUEDEN LLENAR

    protected $fillable = [
        'nombre_cliente',
        'correo',
        'telefono'
    ];

    public $timestamps = false;

    protected $primaryKey = 'id';

}

?>
```

2. Crear el controlador

Crear el archivo `ClienteController.php` dentro de la carpeta de controladores, puedes utilizar el comando `php artisan make:controller ClienteController` en la consola para crear el controlador

Dentro del archivo creado encontrarás la siguiente estructura:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;



class ClienteController extends Controller
{
```

```
}
?>
```

Dentro de la clase `ClienteController` será donde definamos todos nuestras funciones para nuestro proyecto.

Antes de comenzar a realizar las funciones debemos debemos importar la clase Cliente que anteriormente creamos, pero también importaremos la clase DB que nos permitirá realizar consultas SQL.

```
use Illuminate\Support\Facades\DB; // esto importa la clase D
use App\Models\Cliente;
```

Una vez hayamos importado estas clases, podemos crear las vistas, rutas y funciones en el controlador para nuestro crud

Para este ejemplo se mostraran las 2 formas para realizar las funciones (Eloquent y SQL)

2.1. Función para ver los clientes

```
    // * ======================================================
    // * ====================== VER CLIENTES ==================
    // * ======================================================

    public function verClientes()
    {
        // --> Obtenemos todos los clientes
        $clientes = Cliente::all();

        // --> Pasamos los datos a la vista clientes.blade.ph
        return view('clientes', [
            'clientes' => $clientes
        ]);
    }

    // OBTENEMOS TODOS LOS CLIENTES CON CONSULTA SQL
```

```
    public function verClientesSQL() // esta forma es con SQL
    {
        $clientes = DB::select('SELECT * FROM clientes');
        return view('clientes', [
            'clientes' => $clientes
        ]);
    }
```

Como podemos ver la primer función es la forma de obtener todos los clientes con Eloquent mientras que en el segundo método utilizamos una consulta SQL, el uso de cualquiera de estos funcionará de la misma manera

2.2. Función para agregar un cliente

```
        // * ================================================
    // * ============== AGREGAR UN CLIENTE ================
    // * ================================================

    public function agregarCliente(Request $request)
    {

        // Crear un nuevo cliente y asignar los valores de lo
        $cliente = new Cliente();
        $cliente->nombre_cliente = $request->nombre_cliente;
        $cliente->correo = $request->correo;
        $cliente->telefono = $request->telefono;
        $cliente->save();

        // Redireccionar a la página de clientes después de g
        return redirect('/clientes');
    }


    // AGREGAR UN CLIENTE CON CONSULTA SQL
    public function agregarClienteSQL(Request $request)

    {
        DB::insert('INSERT INTO clientes (nombre_cliente, cor
```

```
            $request->nombre_cliente,
            $request->correo,
            $request->telefono
        ]);
        return redirect('/clientes');
    }
```

Al igual que el ejemplo anterior, la primera funcion es con Eloquent y el segundo con Sentencia SQL.

En la primera función se crea una instancia de la clase 'Cliente' y se obtiene el valor de los campos del formulario con la variable `$request` para asignarlos a los campos correspondientes

En la segunda función simplemente se obtienen los valores de los campos igual que en el primer método y se utiliza la sentencia `INSERT INTO` para insertar dichos valores

2.3. Método para editar un cliente

```
    // * ======================================================
    // * =============== EDITAR UN CLIENTE ===================
    // * ======================================================

    public function editarCliente($id)
    {
        $cliente = Cliente::find($id);
        return view('editar_cliente', [
            'cliente' => $cliente
        ]);
    }


    public function actualizarCliente(Request $request, $id)
    {
        $cliente = Cliente::find($id);
        $cliente->nombre_cliente = $request->nombre_cliente;
```

```
        $cliente->correo = $request->correo;
        $cliente->telefono = $request->telefono;
        $cliente->save();
        return redirect('/clientes');
    }


    public function editarClienteSQL($id)
    {
        $cliente = DB::select('SELECT * FROM clientes WHERE i
        return view('editar_cliente', [
            'cliente' => $cliente[0] // necesario porque lara
        ]);
    }

    public function actualizarClienteSQL(Request $request, $i
    {
        DB::update('UPDATE clientes SET nombre_cliente = ?, c
            $request->nombre_cliente,
            $request->correo,
            $request->telefono,
            $id
        ]);
        return redirect('/clientes');
    }
```

En el caso para editar un cliente se requiere de 2 funciones , editarCliente y actualizarCliente

- **editarCliente =** Esta función será la que reciba el id del cliente con el cual obtendrá los datos de los mismos y los enviara a la vista `editar_cliente.blade.php` que es la vista del formulario para editar el cliente
- actualizarCliente = Esta función se encarga de recibir los datos del formulario para editar el cliente
- Explicación editarClienteSQL = En esta función observas que pasa $cliente[0] a la vista, esto es necesario porque cuando ejecutas una consulta SQL directamente usando el método `DB::select()` en Laravel,

obtienes una matriz de resultados. Incluso si esperas un solo resultado, Laravel todavía lo devuelve como un array.

Así que al pasar $cliente[0] a la vista, se están pasando únicamente los datos del primer cliente del array

2.4. Función para eliminar un cliente

```
    // * ================================================
    // * ============== ELIMINAR UN CLIENTE ============
    // * ================================================

    public function eliminarCliente($id)
    {
        $cliente = Cliente::find($id);
        $cliente->delete();
        return redirect('/clientes');
    }

    public function eliminarClienteSQL($id)
    {
        DB::delete('DELETE FROM clientes WHERE id = ?', [$
        return redirect('/clientes');
    }
```

La función para eliminar un cliente es bastante sencilla, como puedes observar, ambos métodos reciben un id que será el id del cliente que se eliminará.

En el caso de Eloquent se utiliza el `delete()` para borrar el cliente con la id proporcionada mientras que en el método que utiliza SQL se utiliza la sentencia `DELETE FROM` .

Terminando el código completo de `ClienteController.php` deberia verse de la siguiente manera:

```
<?php

namespace App\Http\Controllers;
```

```
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB; // esto importa la clas
use App\Models\Cliente;

class ClienteController extends Controller
{

    // * ================================================
    // * ====================== VER CLIENTES ============
    // * ================================================

    public function verClientes()
    {
        // --> Obtenemos todos los clientes
        $clientes = Cliente::all();

        // --> Pasamos los datos a la vista clientes.blade
        return view('clientes', [
            'clientes' => $clientes
        ]);
    }

    // OBTENEMOS TODOS LOS CLIENTES CON CONSULTA SQL

    public function verClientesSQL() // esta forma es con 
    {
        $clientes = DB::select('SELECT * FROM clientes');
        return view('clientes', [
            'clientes' => $clientes
        ]);
    }

    // * ================================================
    // * ================== AGREGAR UN CLIENTE ==========
    // * ================================================

    public function agregarCliente(Request $request)
```

```
    {

        // Crear un nuevo cliente y asignar los valores de
        $cliente = new Cliente();
        $cliente->nombre_cliente = $request->nombre_client
        $cliente->correo = $request->correo;
        $cliente->telefono = $request->telefono;
        $cliente->save();

        // Redireccionar a la página de clientes después d
        return redirect('/clientes');
    }


    // AGREGAR UN CLIENTE CON CONSULTA SQL
    public function agregarClienteSQL(Request $request)

    {
        DB::insert('INSERT INTO clientes (nombre_cliente,
            $request->nombre_cliente,
            $request->correo,
            $request->telefono
        ]);
        return redirect('/clientes');
    }

    // * ================================================
    // * =============== EDITAR UN CLIENTE ==============
    // * ================================================

    public function editarCliente($id)
    {
        $cliente = Cliente::find($id);
        return view('editar_cliente', [
            'cliente' => $cliente
        ]);
    }
```

```
    public function actualizarCliente(Request $request, $i
    {
        $cliente = Cliente::find($id);
        $cliente->nombre_cliente = $request->nombre_client
        $cliente->correo = $request->correo;
        $cliente->telefono = $request->telefono;
        $cliente->save();
        return redirect('/clientes');
    }


    public function editarClienteSQL($id)
    {
        $cliente = DB::select('SELECT * FROM clientes WHER
        return view('editar_cliente', [
            'cliente' => $cliente[0] // necesario porque l
        ]);
    }

    public function actualizarClienteSQL(Request $request,
    {
        DB::update('UPDATE clientes SET nombre_cliente = ?
            $request->nombre_cliente,
            $request->correo,
            $request->telefono,
            $id
        ]);
        return redirect('/clientes');
    }

    // * ================================================
    // * =============== ELIMINAR UN CLIENTE ============
    // * ================================================

    public function eliminarCliente($id)
    {
        $cliente = Cliente::find($id);
```

```
        $cliente->delete();
        return redirect('/clientes');
    }

    public function eliminarClienteSQL($id)
    {
        DB::delete('DELETE FROM clientes WHERE id = ?', [$
        return redirect('/clientes');
    }
}
```

3. Agregar Rutas

Se deben agregar las rutas que son las que se encargarán de actuar como intermediarias entre la vista y el controlador.

Estas rutas las agregamos en el archivo `web.php` que se encuentra en el directorio `resources/routes/`

```
<?php

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\ClienteController;

Route::get('/', [ClienteController::class, 'verClientes']); /

Route::get('/clientes', [ClienteController::class, 'verClient

Route::post('/agregar-cliente', [ClienteController::class, 'a

Route::get('/editar-cliente/{id}', [ClienteController::class,

Route::put('/actualizar-cliente/{id}', [ClienteController::cl

Route::get('/eliminar-cliente/{id}', [ClienteController::clas
```

```
?>
```

4. Crear las vistas

Ahora debemos crear las vistas para el crud, estas vistas tendrán las rutas que están en el punto anterior.

Todas las vistas las crearemos en el siguiente directorio: `resources/views/`

4.1. Crear la vista clientes.blade.php

```
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initia
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="https://stackpath.bootstrapc
    @vite('resources/css/app.css')
    <title>Clientes</title>
</head>

<body>
    <!-- formulario para agregar clientes -->
    <div class="container">
        <div class="row">
            <div class="col-sm-6 m-auto">
                <h1 class="text-center mt-5">CRUD DE CLIENTES
                <form id="form" class="form-vertical" action=
                    @csrf
                    @method('POST')
                    <div class="form-group">
                        <label for="name">Nombre:</label>
                        <input type="text" class="form-contro
                    </div>
                    <div class="form-group">
                        <label for="correo">Correo:</label>
```

```
                        <input type="text" class="form-contro
                    </div>
                    <div class="form-group">
                        <label for="telefono">Telefono:</labe
                        <input type="text" class="form-contro
                    </div>
                    <div class="text-center">
                        <button type="submit" class="btn btn-
                    </div>
                </form>
            </div>
        </div>

        <!-- tabla para mostrar los clientes -->
        <div class="row">
            <div class="col-sm-8 m-auto">
                <table class="table mt-5">
                    <thead>
                        <tr>
                            <th>Id</th>
                            <th>Nombre</th>
                            <th>Correo</th>
                            <th>Telefono</th>
                            <th>Editar</th>
                            <th>Eliminar</th>
                        </tr>
                    </thead>
                    <tbody id="tbody">
                        @foreach ($clientes as $cliente)
                            <tr>
                                <td>{{ $cliente->id }}</td>
                                <td>{{ $cliente->nombre_clien
                                <td>{{ $cliente->correo }}</t
                                <td>{{ $cliente->telefono }}<
                                <td><a href="{{ route('client
                                <td><a href="{{ route('client
                            </tr>
                        @endforeach
```

```
                    </tbody>
                </table>
            </div>
        </div>
    </div>

    @vite('resources/js/app.js')
</body>
</html>
```

4.2. Crear la vista editar_cliente.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initia
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="https://stackpath.bootstrapc
    @vite('resources/css/app.css')
    <title>Editar Cliente</title>
</head>
<body>
    <!-- formulario para editar clientes -->
    <div class="container">
        <div class="row">
            <div class="col-sm-6 m-auto">
                <h1 class="text-center mt-5">EDITAR CLIENTE</
                <form id="form" class="form-vertical" action=
                    @csrf
                    @method('PUT')
                    <div class="form-group">
                        <label for="name">Nombre:</label>
                        <input type="text" class="form-contro
                    </div>
                    <div class="form-group">
                        <label for="correo">Correo:</label>
```

```
                        <input type="text" class="form-contro
                    </div>
                    <div class="form-group">
                        <label for="telefono">Telefono:</labe
                        <input type="text" class="form-contro
                    </div>
                    <div class="text-center">
                        <button type="submit" class="btn btn-
                    </div>
                </form>
            </div>
        </div>
    </div>

    @vite('resources/js/app.js')
</body>
</html>
```