Puesta en producción Segura

Proyecto web final de ciberseguridad

### 

| [**Proyecto web y base de datos 2**](#_5izycdxul2cw)  [Docker-compose.yml 2](#_a5dr3h4y2pzb)  [Base de datos 4](#_9xyqdnjlwhih)  [Conexión 4](#_9xyqdnjlwhih)  [Tabla users por defecto. 5](#_kzbd1co6a0)  [**Operaciones CRUD 5**](#_jltetzhywvmi)  [User.php 5](#_9v9bartjxnds)  [Rutas (web.php) 7](#_tynll138gydl)  [Controladores (RegisterController.php) 8](#_nwnboytl6zf)  [Vistas 10](#_ekiran7us09u)  [Index 10](#_gem2mh4rud6y)  [Register 12](#_xt62kq15fd86) |
| --- |
| Para este proyecto web he optado por los contenedores de Laravel y Mariadb de Bitnami |
| <https://hub.docker.com/r/bitnami/laravel> |
| Proyecto web y base de datos |
| Docker-compose.yml |
| # Copyright VMware, Inc.  # SPDX-License-Identifier: APACHE-2.0  version: '2'  services:  mariadb:  image: docker.io/bitnami/mariadb:11.1  environment:  # ALLOW\_EMPTY\_PASSWORD is recommended only for development.  - ALLOW\_EMPTY\_PASSWORD=yes  - MARIADB\_USER=bn\_myapp  - MARIADB\_DATABASE=bitnami\_myapp  myapp:  image: docker.io/bitnami/laravel:10  ports:  - '8000:8000'  environment:  - DB\_HOST=mariadb  - DB\_PORT=3306  - DB\_USERNAME=bn\_myapp  - DB\_DATABASE=bitnami\_myapp  volumes:  - './my-project:/app'  depends\_on:  - mariadb |
| Lanzar **docker compose up** |
| Ya con ello tenemos un proyecto web con conexión a base de datos dockerizado. |
|  |
|  |
| Base de datos |
| Conexión |
| Laravel se encarga de la conexión del proyecto a la base de datos, para ello hemos de configurar las credenciales en el archivo **.env.** |
| DB\_CONNECTION=mysql  DB\_HOST=127.0.0.1  DB\_PORT=3306  DB\_DATABASE=laravel  DB\_USERNAME=root  DB\_PASSWORD= |
| La configuración de la base de datos es un tema aparte con puntos interesantes y a veces, dependientes del framework como el sembrado, migraciones, permisos…  En este caso usaremos la tabla **users** que viene por defecto, así que solo nos ha de importar que la conexión haya sido efectuada.   * Para ampliar sobre la configuración inicial siga este [link](https://laravel.com/docs/10.x/database). * Herramienta para testeo en línea de comandos recomendada por el framework → [tinker](https://laravel.com/docs/10.x/artisan#tinker) |
|  |
| Tabla users por defecto. |
|  |
| Operaciones CRUD |
| En este punto no tenemos funcionalidad CRUD, la implementaremos con los usuario usando el modelo por defecto. |
| User.php |
| <?php  namespace App\Models;  // use Illuminate\Contracts\Auth\MustVerifyEmail;  use Illuminate\Database\Eloquent\Factories\HasFactory;  use Illuminate\Foundation\Auth\User as Authenticatable;  use Illuminate\Notifications\Notifiable;  use Laravel\Sanctum\HasApiTokens;  class User extends Authenticatable  {  use HasApiTokens, HasFactory, Notifiable;  /\*\*  \* The attributes that are mass assignable.  \*  \* @var array<int, string>  \*/  protected $fillable = [  'name',  'email',  'password',  ];  /\*\*  \* The attributes that should be hidden for serialization.  \*  \* @var array<int, string>  \*/  protected $hidden = [  'password',  'remember\_token',  ];  /\*\*  \* The attributes that should be cast.  \*  \* @var array<string, string>  \*/  protected $casts = [  'email\_verified\_at' => 'datetime',  'password' => 'hashed',  ];  } |
| Rutas (web.php) |
| Designemos las acciones que deseamos implementar como rutas.   * Creación (Create) * Muestra del formulario, * Guardado * Borrado (Delete)   Nos faltaría muestra de la vista individual (read) y edición (update) que no son necesarios para demostración. |
| Una "ruta" se refiere a una URL en tu aplicación web y cómo esa URL se asocia con una acción específica en tu código, generalmente en un controlador. Laravel utiliza un enfoque basado en rutas para manejar las solicitudes HTTP y dirigirlas a las partes apropiadas de tu aplicación. |
| <?php  use Illuminate\Support\Facades\Route;  use App\Http\Controllers\RegisterController;  /\*  |--------------------------------------------------------------------------  | Web Routes  |--------------------------------------------------------------------------  |  | Here is where you can register web routes for your application. These  | routes are loaded by the RouteServiceProvider and all of them will  | be assigned to the "web" middleware group. Make something great!  |  \*/  Route::get('/', [RegisterController::class, 'index'])->name('main');  Route::get('/register', [RegisterController::class, 'create'])->name('register');  Route::post('/register', [RegisterController::class, 'store']);  Route::delete('/users/{user}', [RegisterController::class, 'destroy'])->name('users.destroy'); |
| Controladores (RegisterController.php) |
| Para implementar la lógica usaremos un Controller, llamado RegisterController. |
| Con la ayuda de artisan lo generamos con:  **php artisan make:controller RegisterController** |
| Un controlador **es un archivo PHP que maneja las solicitudes HTTP entrantes y define cómo se deben responder**.  Interesante implementación de Laravel → [Link](https://laravel.com/docs/10.x/controllers#resource-controllers) |
| <?php  namespace App\Http\Controllers;  use App\Models\User;  use Illuminate\Http\Request;  class RegisterController extends Controller  {  public function index()  {  // Retrieve all users from the database  $users = User::all();  // Return the view with the users data  return view('main', ['users' => $users]);  }  public function create()  {  // Return the view for creating a new user  return view('register');  }  public function store(Request $request)  {  // Validation logic can be added here if needed  // Create a new user instance  $user = new User();  // Set user attributes based on the form input  $user->name = $request->input('name');  $user->email = $request->input('email');  $user->password = bcrypt($request->input('password'));  $user->save();  // Redirect to the index page with a success message  return redirect()->route('main')->with('success', 'User created successfully!');  }  public function destroy(Request $request, $id)  {  $user = User::findOrFail($id);  $user->delete();  return redirect()->route('main')->with('success', 'User deleted successfully!');  }  } |
| Vistas |
| En este punto, el back end se encuentra hecho y debería ser testeado contra test elaborados anteriormente. |
| Asumimos que se encuentran testeado y pasamos a la elaboración de vistas. Son necesarias solamente dos:   * Index (main.blade.php) * register(register.blade.php |
| Index |
|  |
| <!-- resources/views/users/index.blade.php -->  <div class="container mt-5">  <h2 class="mb-4">Users List</h2>  @if(session('success'))  <div class="alert alert-success">  {{ session('success') }}  </div>  @endif  <div class="mb-3">  <a href="{{ route('register') }}" class="btn btn-success">Add New User</a>  </div>  <table class="table table-bordered table-striped">  <thead class="thead-dark">  <tr>  <th>ID</th>  <th>Name</th>  <th>Email</th>  <th>Actions</th>  </tr>  </thead>  <tbody>  @foreach($users as $user)  <tr>  <td>{{ $user->id }}</td>  <td>{{ $user->name }}</td>  <td>{{ $user->email }}</td>  <td>  <form action="{{ route('users.destroy', ['user' => $user->id]) }}" method="POST" style="display: inline;">  @csrf  @method('delete')  <button type="submit" class="btn btn-danger btn-sm" >Delete</button>  </form>  </td>  </tr>  @endforeach  </tbody>  </table>  </div>  <!-- Add Bootstrap CSS from CDN -->  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet"> |
| Register |
|  |
| <!DOCTYPE html>  <html lang="es">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Registro de Usuario</title>  <!-- Add Bootstrap CSS from CDN -->  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">  </head>  <body class="bg-light">  <div class="container mt-5">  <h2 class="text-center">Registro de Usuario</h2>  <form method="POST" action="{{ route('register') }}" class="mt-3">  @csrf  <div class="mb-3">  <label for="name" class="form-label">Nombre:</label>  <input type="text" class="form-control" id="name" name="name" value="{{ old('name') }}" required autofocus>  @error('name')  <p class="text-danger">{{ $message }}</p>  @enderror  </div>  <div class="mb-3">  <label for="email" class="form-label">Correo Electrónico:</label>  <input type="email" class="form-control" id="email" name="email" value="{{ old('email') }}" required>  @error('email')  <p class="text-danger">{{ $message }}</p>  @enderror  </div>  <div class="mb-3">  <label for="password" class="form-label">Contraseña:</label>  <input type="password" class="form-control" id="password" name="password" required>  @error('password')  <p class="text-danger">{{ $message }}</p>  @enderror  </div>  <div class="mb-3">  <label for="password\_confirmation" class="form-label">Confirmar Contraseña:</label>  <input type="password" class="form-control" id="password\_confirmation" name="password\_confirmation" required>  @error('password\_confirmation')  <p class="text-danger">{{ $message }}</p>  @enderror  </div>  <div class="mb-3">  <button type="submit" class="btn btn-primary">Registrar</button>  </div>  </form>  </div>  <!-- Optional: Include Bootstrap JS (for certain features) -->  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>  </body>  </html> |