cd /c/xampp/htdocs/proyectos

composer create-project laravel/laravel contactos2 "7.*"

cd contactos2

```
$ php artisan --version
Laravel Framework 7.30.4
```

```
$ php artisan serve
Laravel development server started: http://127.0.0.1:8000
[Mon Sep 13 02:24:43 2021] PHP 7.4.12 Development Server (http://127.0.0.1:8000)
started
```

http://127.0.0.1:8000/
$ code .

Teclar Ctrl + C

CONFIGURAR el archivo .env
APP_NAME=Contacto
DB_DATABASE=contactos

CONFIGURAR Carpeta config/app.php
Reemplazar 'timezone' => 'UTC',
Por
'timezone' => 'America/Bogota',

```
$ php artisan make:migration directorio
Created Migration: 2021_09_12_032846_directorio
```

```
$ php artisan make:migration directorio --create=directorios
//La migración se llama directorio y la tabla a crear directorios
```
Vamos al archivo de la migración a crear la tabla

Vamos al archivo de la migración a crear la tabla

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class Directorio extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
```

```php
     */
    public function up()
    {
        Schema::create('directorios', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
        });

        Schema::create('directorios', function (Blueprint $table) {
            $table->id();
            $table->string('nombre');
            $table->string('direccion')->nulable();
            $table->integer('telefono')->unique();
            $table->string('foto')->nulable();
            $table->timestamps(); });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('directorios');
    }
}
```

php artisan migrate

```
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (0.59 seconds)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (0.78 seconds)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (0.33 seconds)
Migrating: 2021_09_12_032846_directorio
Migrated: 2021_09_12_032846_directorio (0.48 seconds)
Michelle@DESKTOP-F3H6SA3 MINGW64 /c/xampp/htdocs/proyectos/contactos2
$
```

php artisan make:seeder DirectorioSeeder

```
Seeder created successfully.
Michelle@DESKTOP-F3H6SA3 MINGW64 /c/xampp/htdocs/proyectos/contactos2
$
```

```php
<?php

use Illuminate\Database\Seeder;

class DirectorioSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        DB::table('directorios')->insert([

            [ 'nombre' =>'MARVIN DE LA PEÑA',
            'direccion'=>'Mz 1 Lt 18 Etapa 3',
            'telefono'=>'100',
            'foto'=>'null' ],

            [ 'nombre' =>'ARIANA DE LA PEÑA',
            'direccion'=>'Mz 1 Lt 18 Etapa 3',
            'telefono'=>'200',
            'foto'=>'null' ] ,

            [ 'nombre' =>'KEREN DIAZ',
            'direccion'=>'Pinillos Bolivar',
            'telefono'=>'300',
            'foto'=>'null' ]
         ]);









     }
}
```

IR A DatabaseSeeder y descomentar + Actualizar la línea
`//$this->call(UserSeeder::class);`
por
$this->call(DirectorioSeeder::class);

```php
<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        $this->call(DirectorioSeeder::class);
    }
}
```

php artisan db:seed

```
Database seeding completed successfully.
Michelle@DESKTOP-F3H6SA3 MINGW64 /c/xampp/htdocs/proyectos/contactos2
$
```

php artisan make:model Directorio

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Directorio extends Model
{
    protected $table = 'directorios';

    protected $fillable = [
        'nombre_completo',
        'direccion',
        'telefono',
        'foto'
    ];

    protected $hidden = [
        'created_at', 'updated_at'
    ];
```

```
}
```

```
$ php artisan make:controller DirectorioController --api
Controller created successfully.
Michelle@DESKTOP-F3H6SA3 MINGW64 /c/xampp/htdocs/proyectos/contactos2
```

```
Agregar esta ruta en api.php
Route::ApiResource('directorios', 'DirectorioController');
```

```
En el controlador agregar listar datos y retornar un solo registro Método: Get
Tambien el método Store
```

```
//GET listar datos

     public function index(Request $request)
    {
         if ($request->has('txtBuscar')) {
            $contactos = Directorio::where('nombre', 'like', '%' . $request-
>txtBuscar . '%')
                ->orWhere('telefono', $request->txtBuscar)
                ->get();
        } else
            $contactos = Directorio::all();

        return $contactos;
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    //Post Insertar Datos
    public function store(Request $request)
    {   $input = $request->all();
        Directorio::create($input);
        return response()->json([
        'res'=>true,
        'message' => 'Registro Guardado Correctamente'
        ]);
    }

    /**
     * Display the specified resource.
     *
```

```
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    //Retorna un solo Registro
    public function show(Directorio $directorio)
    {
        return $directorio;
    }
```

En Postman

Insertar este registro

```
{"nombre":"IVONNE ELVIRA",
"direccion":"MADRID ESPAÑA",
"telefono":1965,
"foto":"1"
}
```

Nota: Si no le coloco el campo foto, no me deja guardar

```
Michelle@DESKTOP-F3H6SA3 MINGW64 /c/xampp/htdocs/proyectos/contactos2
$ php artisan make:request CreateDirectorioRequest
Request created successfully.
Michelle@DESKTOP-F3H6SA3 MINGW64 /c/xampp/htdocs/proyectos/contactos2

Michelle@DESKTOP-F3H6SA3 MINGW64 /c/xampp/htdocs/proyectos/contactos2
$ php artisan make:request UpdateDirectorioRequest
Request created successfully.
Michelle@DESKTOP-F3H6SA3 MINGW64 /c/xampp/htdocs/proyectos/contactos2
```

Cambiar a true en CreateDirectorioRequest

```
public function authorize()
    {
        return true;
    }
```

Agregar también

```
public function rules()
    {
        return [
            'nombre' => 'required|min:5|max:100',
            'telefono' => 'required|unique:directorios,telefono'
        ];
    }
```

Quedando asi:

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class CreateDirectorioRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'nombre' => 'required|min:5|max:100',
            'telefono' => 'required|unique:directorios,telefono'
        ];
    }
}
```

En el controlador cambiar la siguiente línea

```php
//Post Insertar Datos
    public function store(Request $request)
```

por

```php
//Post Insertar Datos
    public function store(CreateDirectorioRequest $request)
```

No se le olvide importarlo

use App\Http\Requests\CreateDirectorioRequest;

Quedando así:

```php
<?php

namespace App\Http\Controllers;

use App\Directorio;
use App\Http\Requests\CreateDirectorioRequest;

use Illuminate\Http\Request;

class DirectorioController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */

     //GET listar datos
     public function index(Request $request)
    {
         if ($request->has('txtBuscar')) {
            $contactos = Directorio::where('nombre', 'like', '%' . $request-
>txtBuscar . '%')
                ->orWhere('telefono', $request->txtBuscar)
                ->get();
        } else
            $contactos = Directorio::all();

        return $contactos;
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    //Post Insertar Datos
    public function store(CreateDirectorioRequest $request)
    {   $input = $request->all();
```

```
        Directorio::create($input);
        return response()->json([
        'res'=>true,
        'message' => 'Registro Guardado Correctamente'
        ]);
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    //Retorna un solo Registro
    public function show(Directorio $directorio)
    {
        return $directorio;
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        //
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        //
    }
}
```

Ya están aplicadas las validaciones para número de teléfono que ya se encuentre en la base de datos y para cuando se intente guardar un registro sin el campo nombre.

Grabemos ahora este otro

```
{"nombre":"CARMEN ALIDA GUZMAN RAMOS",
"direccion":"TALAIGUA NUEVO",
"telefono":4000,
"foto":"2"
}
```

Vamos a crear el método Update en el controlador

```
//PUT para modificar datos
   public function update(UpdateDirectorioRequest $request, Directorio $dire
ctorio)
   {
       $input = $request->all();

       $directorio->update($input);
       return response()->json([
           'res' => true,
           'message' => 'Actualizado correctamente'
       ]);
   }
```

Luego, vamos a

```
Requests\UpdateDirectorioRequest;
```

Y hacemos los siguientes cambios;cambiamos a true

```
 public function authorize()
    {
        return false;
    }
```

Quedando así

```
public function authorize()
    {
        return true;
    }
```

Ahora, actualizamos las validaciones

```
public function rules()
    {
        return [
            //
        ];
    }
```

Quedando así:

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class UpdateDirectorioRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'nombre' => 'required|min:5|max:100',
            'telefono' => 'required|unique:directorios,telefono'
        ];
    }
}
```

Ahora, podemos utilizar la actualización o Update


```
{"nombre":"CARMEN ALIDA GUZMAN RAMOS",
"direccion":"CARTAGENA- BOLIVAR",
"telefono":4001,"foto":"2"}
```


ASI QUEDARIA EL CONTROLADOR

```php
<?php

namespace App\Http\Controllers;

use App\Directorio;
use App\Http\Requests\CreateDirectorioRequest;
use App\Http\Requests\UpdateDirectorioRequest;
```

```php
use Illuminate\Http\Request;

class DirectorioController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */

     //GET listar datos
     public function index(Request $request)
    {
         if ($request->has('txtBuscar')) {
            $contactos = Directorio::where('nombre', 'like', '%' . $request-
>txtBuscar . '%')
                ->orWhere('telefono', $request->txtBuscar)
                ->get();
        } else
            $contactos = Directorio::all();

        return $contactos;
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    //Post Insertar Datos
    public function store(CreateDirectorioRequest $request)
    {   $input = $request->all();
        Directorio::create($input);
        return response()->json([
        'res'=>true,
        'message' => 'Registro Guardado Correctamente'
        ]);
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
```

```php
     * @return \Illuminate\Http\Response
     */
    //Retorna un solo Registro
    public function show(Directorio $directorio)
    {
        return $directorio;
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
   //PUT para modificar datos
   public function update(UpdateDirectorioRequest $request, Directorio $dire
ctorio)
   {
       $input = $request->all();
       $directorio->update($input);
       return response()->json([
           'res' => true,
           'message' => 'Actualizado correctamente'
       ]);
   }


    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    //DELETE elimina datos
    public function destroy($id)
    {
        Directorio::destroy($id);
        return response()->json([
            'res' => true,
            'message' => 'Eliminado correctamente'
        ], 200);
    }
}
```

Ya se puede eliminar en Postman

Escribir el siguiente código en: [App][Exceptions][Handler.php]

```
public function render($request, Throwable $exception)
    {
        if($exception instanceof ModelNotFoundException){
            return response()-
>json(["res" => false, "error" => "Error de modelo"], 400);
        }

        if($exception instanceof QueryException){
            return response()-
>json(["res" => false, "message" => "Error de consulta BDD " , $exception-
>getMessage()], 500);
        }

        if($exception instanceof HttpException){
            return response()-
>json(["res" => false, "message" => "Error de ruta"], 404);
        }

        if($exception instanceof AuthenticationException){
            return response()-
>json(["res" => false, "message" => "Error de autenticación"], 401);
        }

        if ($exception instanceof AuthorizationException) {
            return response()-
>json(["res" => false, "message" => "Error de autorización, no tiene permiso
s"], 403);
        }

        return parent::render($request, $exception);
    }
```

Quedando así

```
<?php

namespace App\Exceptions;

use Dotenv\Exception\ValidationException;
use Illuminate\Auth\Access\AuthorizationException;
```

```php
use Illuminate\Auth\AuthenticationException;
use Illuminate\Database\Eloquent\ModelNotFoundException;
use Illuminate\Database\QueryException;
use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;
use Symfony\Component\HttpKernel\Exception\HttpException;
use Throwable;

class Handler extends ExceptionHandler
{
    /**
     * A list of the exception types that are not reported.
     *
     * @var array
     */
    protected $dontReport = [
        //
    ];

    /**
     * A list of the inputs that are never flashed for validation exceptions
.
     *
     * @var array
     */
    protected $dontFlash = [
        'password',
        'password_confirmation',
    ];

    /**
     * Report or log an exception.
     *
     * @param  \Throwable  $exception
     * @return void
     *
     * @throws \Throwable
     */
    public function report(Throwable $exception)
    {
        parent::report($exception);
    }

    /**
     * Render an exception into an HTTP response.
     *
```

```
     * @param  \Illuminate\Http\Request  $request
     * @param  \Throwable  $exception
     * @return \Symfony\Component\HttpFoundation\Response
     *
     * @throws \Throwable
     */
    public function render($request, Throwable $exception)
    {
        if($exception instanceof ModelNotFoundException){
            return response()-
>json(["res" => false, "error" => "Error de modelo"], 400);
        }

        if($exception instanceof QueryException){
            return response()-
>json(["res" => false, "message" => "Error de consulta BDD " , $exception-
>getMessage()], 500);
        }

        if($exception instanceof HttpException){
            return response()-
>json(["res" => false, "message" => "Error de ruta"], 404);
        }

        if($exception instanceof AuthenticationException){
            return response()-
>json(["res" => false, "message" => "Error de autenticación"], 401);
        }

        if ($exception instanceof AuthorizationException) {
            return response()-
>json(["res" => false, "message" => "Error de autorización, no tiene permiso
s"], 403);
        }

        return parent::render($request, $exception);
    }
}
```

Vamos a cargar la Fotografía

Creamos en el controlador DirectorioController lo siguiente:

```
private function cargarArchivo($file)
```

```
    {
        $nombreArchivo = time(). '.'. $file->getClientOriginalExtension();
        $file->move(public_path('fotografias'), $nombreArchivo);
        return $nombreArchivo;
    }
```

Y actualizamos con el siguiente código el método store

```
//Post Insertar Datos
    public function store(CreateDirectorioRequest $request)
    {   $input = $request->all();

        if($request->has('foto'))
            $input['foto'] = $this->cargarArchivo($request->foto);

        Directorio::create($input);
        return response()->json([
        'res'=>true,
        'message' => 'Registro Guardado Correctamente'
        ]);
    }
```

También actualizamos el método update

```
//PUT para modificar datos
   public function update(UpdateDirectorioRequest $request, Directorio $dire
ctorio)
   {
       $input = $request->all();
       if($request->has('foto'))
            $input['foto'] = $this->cargarArchivo($request->foto);

       $directorio->update($input);
       return response()->json([
           'res' => true,
           'message' => 'Actualizado correctamente'
       ]);
   }
```

Ahora verificar en Postman.

Verifique Y funciona perfectamente tanto la inserción de la imagen como la actualización.

Vamos a trabajar la seguridad

En el archivo routes/api.php adicionamos:

```
Route::group(['middleware'=>'auth:api'], function(){
    Route::ApiResource('directorios', 'DirectorioController');
    //Route::post('logout', 'UserController@logout');
});
```

Agregar este campo en la migración de user

```
$table->string('api_token')->nullable();
```

Ejecuto el siguiente comando

```
$ php artisan migrate:fresh --seed
Dropped all tables successfully.
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (0.99 seconds)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (0.59 seconds)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (0.31 seconds)
Migrating: 2021_09_13_023258_directorio
Migrated:  2021_09_13_023258_directorio (0.49 seconds)
Seeding: DirectorioSeeder
Seeded:  DirectorioSeeder (0.12 seconds)
Database seeding completed successfully.

Ahora creamos un nuevo controlador, el de usuario
Michelle@DESKTOP-F3H6SA3 MINGW64 /c/xampp/htdocs/proyectos/contactos2
$ php artisan make:controller UserController
Controller created successfully.

El controlador quedaría así:
```

```
<?php

namespace App\Http\Controllers;

use App\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function store(Request $request)
    {   $input = $request->all();
        $input['password'] = Hash::make($request->password);
        User::create($input);
        return response()->json([
        'res'=>true,
```

```
        'message' => 'Usuario Creado Correctamente'
        ]);
    }
}
```

Ahora vamos a registrar la Api, la ruta

```
Route::post('users', 'UserController@store');
```

Vamos a probar en postman

Probar con este registro

```
{
"name":"MARVIN BLAS",
"email":"marvin.software@gmail.com",
"password":"12345"

}
```

Guardado sin problemas

Agregamos en el UserController

```
 public function login(Request $request)
    { $user=User::whereEmail($request->email)->first();
        if(!is_null($user)&&Hash::check($request->password,$user->password))
        {
            return response()->json([
                'res'=>true,
                'message' => 'Bienvenido al sistema'
                ]);
        } else {
                return response()->json([
                'res'=>false,
                'message' => 'Cuenta o password incorrecto'
                ]);
        }

    }
```

Ahora creamos la ruta de login

```
Route::post('login', 'UserController@login');
```

Verificamos login

Funciona perfectamente

Ahora agregaremos el token, el controlador quedaría asi:

```php
<?php

namespace App\Http\Controllers;
use App\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Str;

class UserController extends Controller
{
    public function store(Request $request)
    {   $input = $request->all();
        $input['password'] = Hash::make($request->password);
        User::create($input);
        return response()->json([
        'res'=>true,
        'message' => 'Usuario Creado Correctamente'
        ]);
    }

    public function login(Request $request)
    {   $user=User::whereEmail($request->email)->first();
        if(!is_null($user)&&Hash::check($request->password,$user->password))
        {   $user->api_token = Str::random(100);
            $user->save();

            return response()->json([
                'res'=>true,
                'token' =>$user->api_token,
                'message' => 'Bienvenido al sistema'
                ]);
        } else {
                return response()->json([
                'res'=>false,
```

```
                'message' => 'Cuenta o password incorrecto'
                ]);
        }




    }

}
```

Funciona Perfectamente el Token

Agregamos la ruta logout en api.php debe estar protegida en el grupo

```
Route::post('logout', 'UserController@logout');
```

Ahora creamos la función Logout

Asi quedaría el controlador UserController

```
<?php

namespace App\Http\Controllers;
use App\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Str;

class UserController extends Controller
{
    public function store(Request $request)
    {   $input = $request->all();
        $input['password'] = Hash::make($request->password);
        User::create($input);
        return response()->json([
        'res'=>true,
        'message' => 'Usuario Creado Correctamente'
        ]);
    }

    public function login(Request $request)
```

```
    {   $user=User::whereEmail($request->email)->first();
        if(!is_null($user)&&Hash::check($request->password,$user->password))
        {   $user->api_token = Str::random(100);
            $user->save();

            return response()->json([
                'res'=>true,
                'token' =>$user->api_token,
                'message' => 'Bienvenido al sistema'
                ]);
        } else {
                return response()->json([
                'res'=>false,
                'message' => 'Cuenta o password incorrecto'
                ]);
        }


    }

    public function logout()
        {
            $user = auth()->user();
            $user->api_token = null;
            $user->save();
            return response()->json(['res' => true,
            'message' => "Adios"]);
        }

}
```