

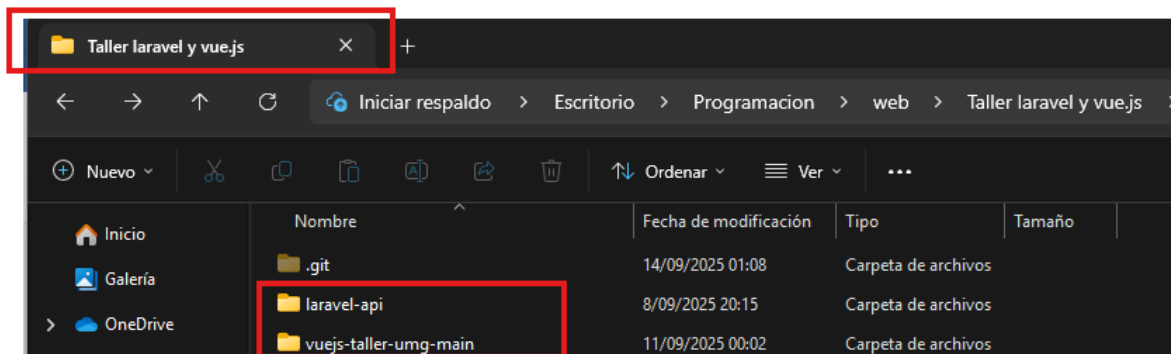
Nombre: Allan Guamuch

Carne:2990-19-22192

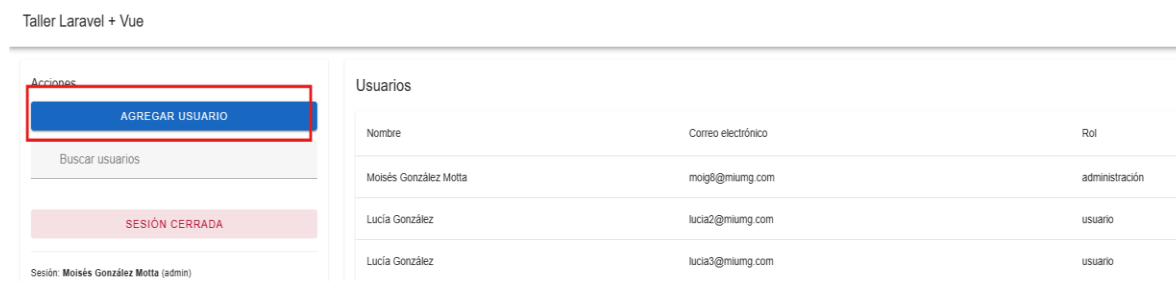
Curso: Desarrollo Web

Link: <https://github.com/ALLANROBER/Taller-laravel-Vuejs.git>

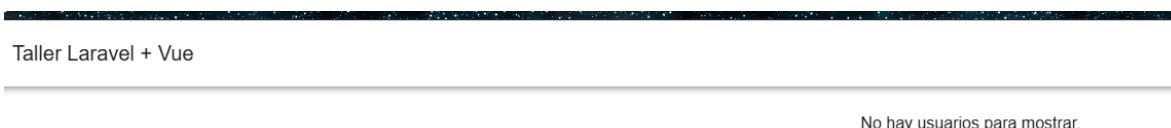
Para este taller cree una carpeta llamada Taller laravel y vue.js en la cual están las carpetas laravel-api y vuejs-taller-umg-main uno que es el proyecto **Laravel (backend)** y el otro **Vue.js (frontend)**



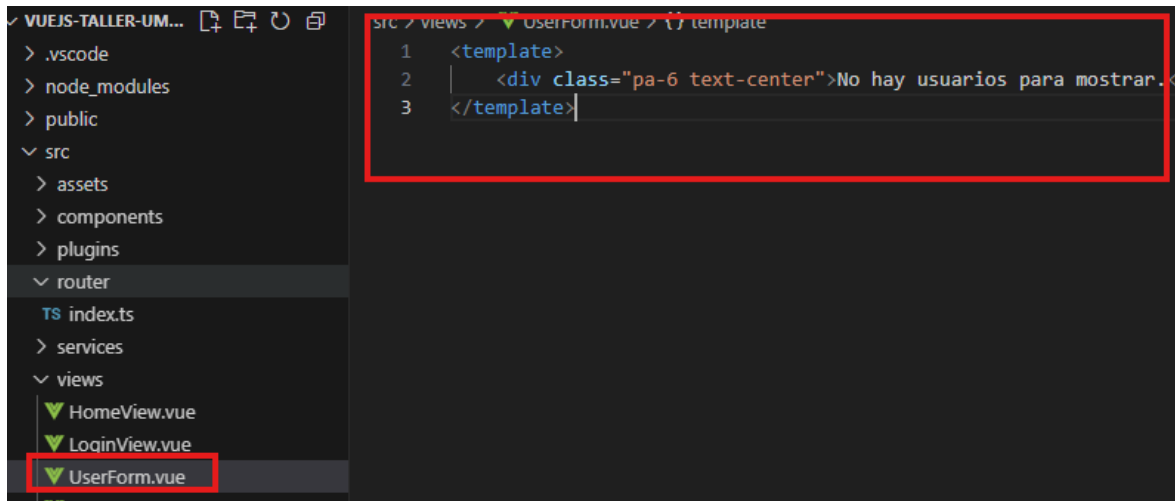
Al ejecutar el proyecto vue logramos ver los usuarios que ya tenemos en nuestra base de datos y también un botón llamado AGREGAR USUARIO



Al apretar el botón Agregar usuario nos redirecciona a otra pagina en la cual solo nos muestra un mensaje

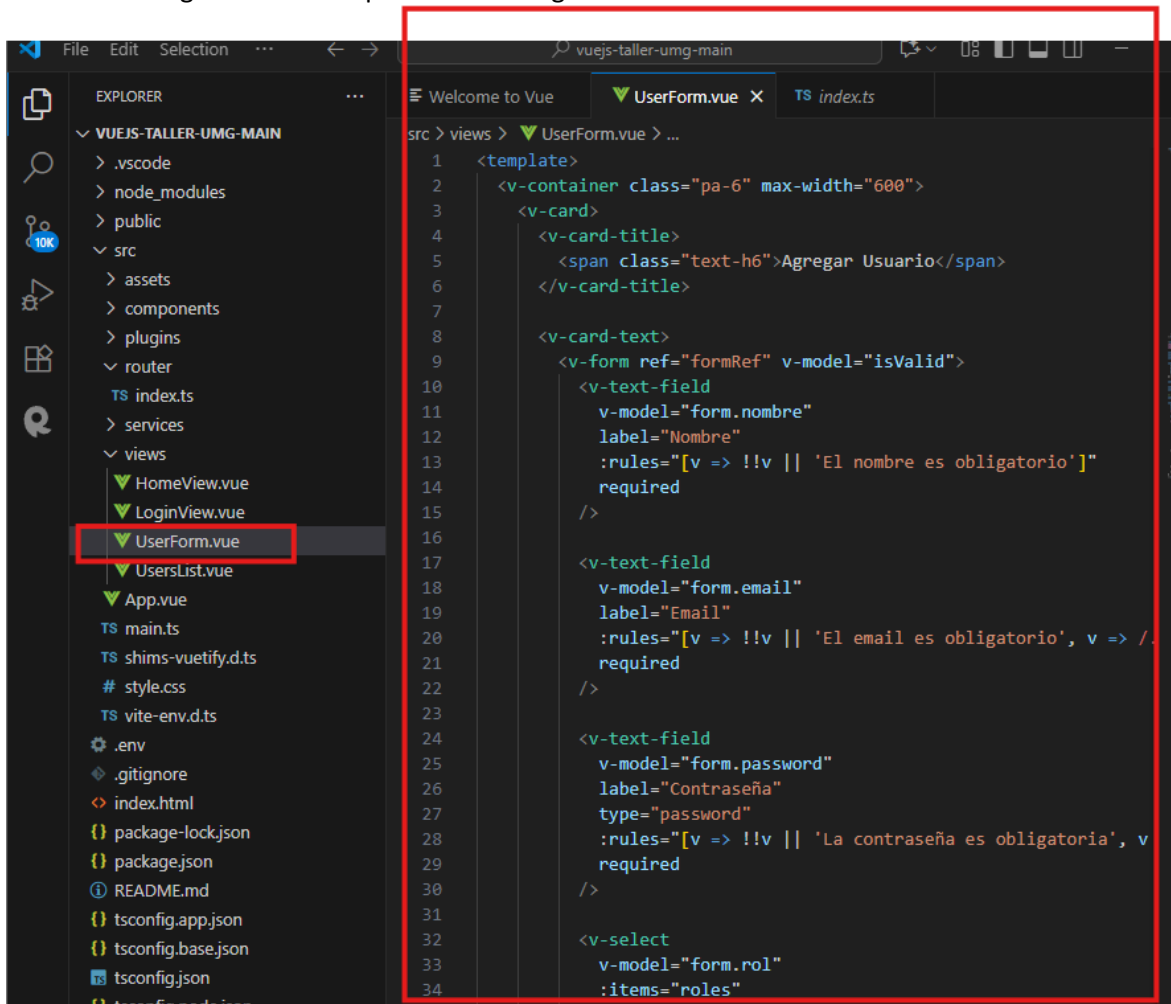


El mensaje proviene del proyecto vue de views/userform.vue, como primer tarea es modificar ese código para que poder agregar usuarios desde el navegador



```
src > views > UserForm.vue > {} template
1  <template>
2  |   <div class="pa-6 text-center">No hay usuarios para mostrar.</div>
3  </template>
```

El código modificado quedaría de la siguiente manera



```
vuejs-taller-umg-main
Welcome to Vue
UserForm.vue x TS index.ts
src > views > UserForm.vue > ...
1  <template>
2  |   <v-container class="pa-6 max-width=600">
3  |     <v-card>
4  |       <v-card-title>
5  |         <span class="text-h6">Agregar Usuario</span>
6  |       </v-card-title>
7  |       <v-card-text>
8  |         <v-form ref="formRef" v-model="isValid">
9  |           <v-text-field
10 |             v-model="form.nombre"
11 |             label="Nombre"
12 |             :rules="[v => !!v || 'El nombre es obligatorio']"
13 |             required
14 |           />
15 |           <v-text-field
16 |             v-model="form.email"
17 |             label="Email"
18 |             :rules="[v => !!v || 'El email es obligatorio', v => /.+@.+/]"
19 |             required
20 |           />
21 |           <v-text-field
22 |             v-model="form.password"
23 |             label="Contraseña"
24 |             type="password"
25 |             :rules="[v => !!v || 'La contraseña es obligatoria', v => /.+@.+/]"
26 |             required
27 |           />
28 |           <v-select
29 |             v-model="form.rol"
30 |             :items="roles"
31 |           />
32 |         </v-form>
33 |       </v-card-text>
34 |     </v-card>
35 |   </v-container>
36 </template>
```

Al guardar el código ya podríamos ver la ventana de agregar usuarios, como vemos en la siguiente imagen ya hemos agregado un usuario y ya podemos guardarlo

Taller Laravel + Vue

Agregar usuario

Nombre

Allan

Correo electrónico

Allan@miumg.com

Contraseña

Rol

administración

GUARDAR

Al dar clic al botón guardar podremos ver una pequeña notificación de que se ha agregado correctamente el usuario

Taller Laravel + Vue

Agregar usuario

Nombre

El nombre es obligatorio

Correo electrónico

El correo electrónico es obligatorio

Contraseña

La contraseña es obligatoria

Rol

Selecciona un rol

Usuario creado correctamente

Al validar la ventana de usuarios podemos ver que efectivamente se agregó el usuario añadido

Taller Laravel + Vue

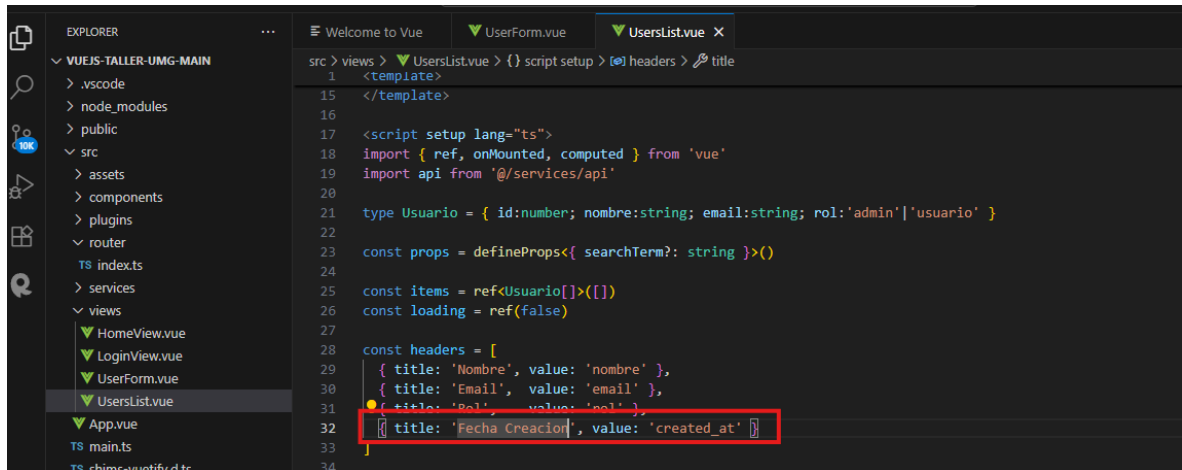
Acciones		
AGREGAR USUARIO		
Buscar usuarios		
SESIÓN CERRADA		
Sesión: Moisés González Motta (admin)		

Usuarios		
Nombre	Correo electrónico	Rol
Moisés González Motta	moig8@miung.com	administración
Lucía González	lucia2@miung.com	usuario
Lucía González	lucia3@miung.com	usuario
sss	jajaj@miung.com	administración
holabb	mijapontebonita@miung.com	administración
Alano	Alano@miung.com	administración

Como vemos en la vista solo podemos visualizar lo que es nombre, correo electrónico y rol, el siguiente reto es hacer que se visualice la fecha de creación del registro de usuario, para eso vamos a agregar en el archivo vue en views/userlist.vue la línea de código para poder visualizarlo, como vemos en la siguiente imagen en el código solo tenemos nombre, email y rol.

```
1 <template>
2   <v-data-table
3     :items="filtered"
4     :headers="headers"
5     :loading="loading"
6     class="elevation-1"
7   >
8     <template #no-data>
9       <div class="pa-6 text-center">No hay usuarios para mostrar.</div>
10    </template>
11  </v-data-table>
12 </template>
13
14 <script setup lang="ts">
15   import { ref, onMounted, computed } from 'vue'
16   import api from '@/services/api'
17
18   type Usuario = { id:number; nombre:string; email:string; rol:'admin'
19
20   const props = defineProps<{ searchTerm?: string }>()
21
22   const items = ref<Usuario[]>([])
23   const loading = ref(false)
24
25   const headers = [
26     { title: 'Nombre', value: 'nombre' },
27     { title: 'Email', value: 'email' },
28     { title: 'Rol', value: 'rol' },
29   ]
30
31   // carga desde la API
32   const fetchUsers = async () => {
33     loading.value = true
34     try {
```

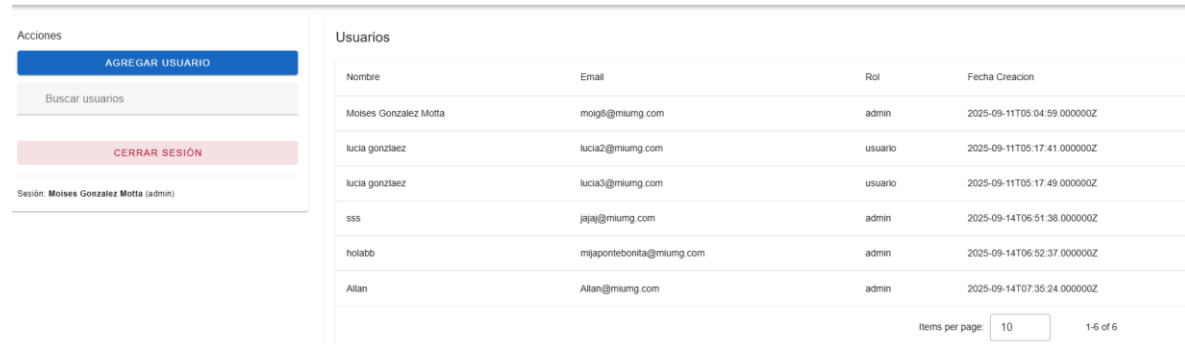
Al agregar el código para visualizar la fecha de usuario quedaría de la siguiente manera



```
src > views > UsersList.vue > {} script setup > headers > title
1 <template>
2
3 </template>
4
5 <script setup lang="ts">
6   import { ref, onMounted, computed } from 'vue'
7   import api from '@services/api'
8
9   type Usuario = { id:number; nombre:string; email:string; rol:'admin'|'usuario' }
10
11   const props = defineProps<{ searchTerm?: string }>()
12
13   const items = ref<Usuario[]>([])
14   const loading = ref(false)
15
16   const headers = [
17     { title: 'Nombre', value: 'nombre' },
18     { title: 'Email', value: 'email' },
19     { title: 'Rol', value: 'rol' },
20     { title: 'Fecha Creación', value: 'created_at' }
21   ]
```

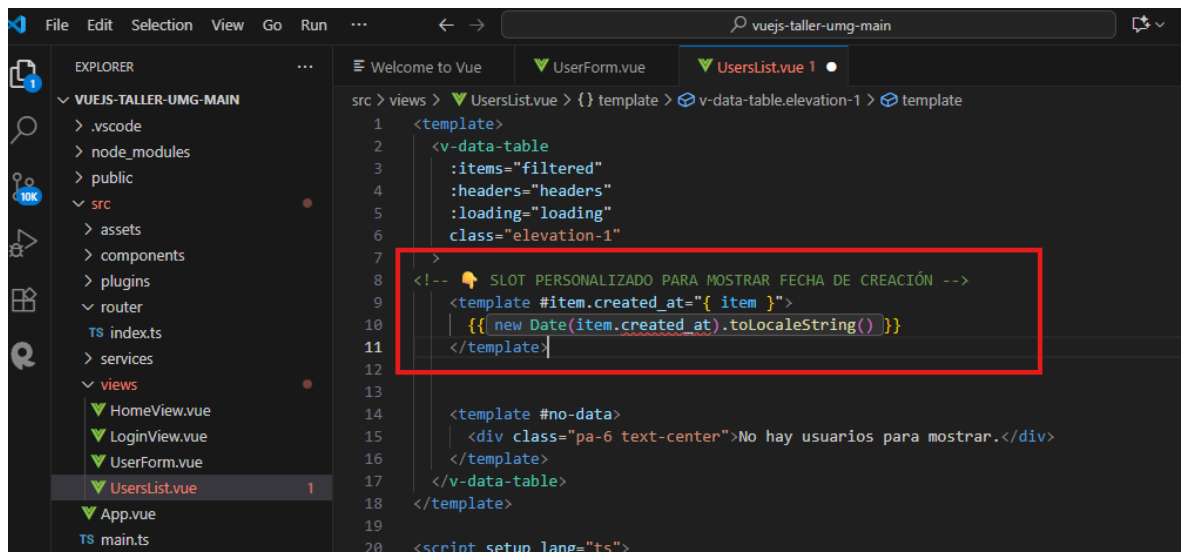
Como vemos ya tenemos la fecha de creación, solo que hay un detalle la fecha no está compuesta para poder componerla y que se mire bonita deberemos agregar otro pedazo de código

Taller Laravel + Vue



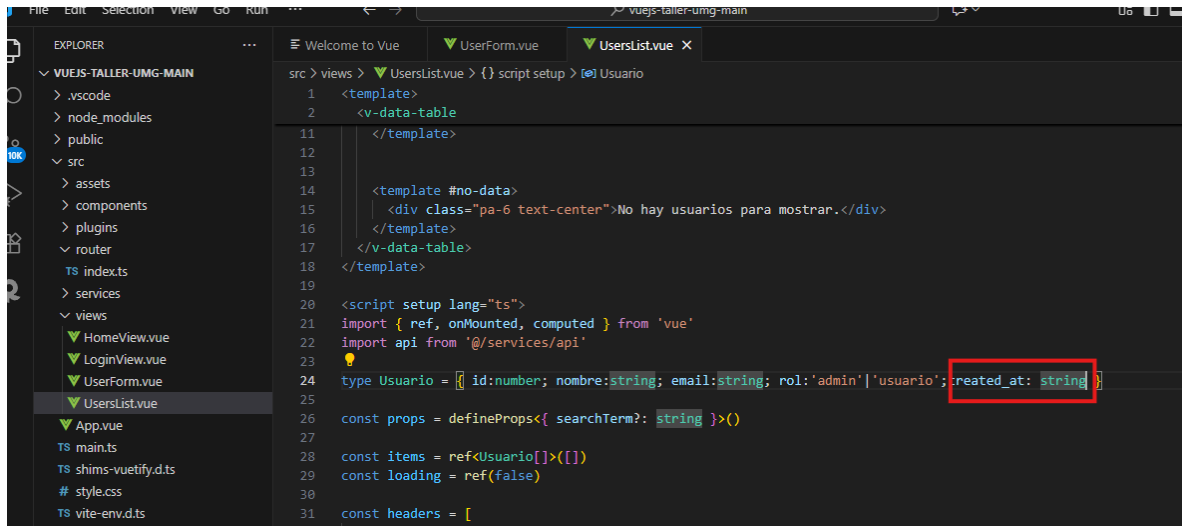
Nombre	Email	Rol	Fecha Creación
Moises Gonzalez Motta	moiga@miung.com	admin	2025-09-11T05:04:59.000000Z
lucia gonzalez	lucia2@miung.com	usuario	2025-09-11T05:17:41.000000Z
lucia gonzalez	lucia3@miung.com	usuario	2025-09-11T05:17:49.000000Z
sss	jajaj@miung.com	admin	2025-09-14T06:51:38.000000Z
holab	mijapontebonita@miung.com	admin	2025-09-14T06:52:37.000000Z
Allan	Allan@miung.com	admin	2025-09-14T07:35:24.000000Z

El código para que la fecha de mire bonita es la siguiente



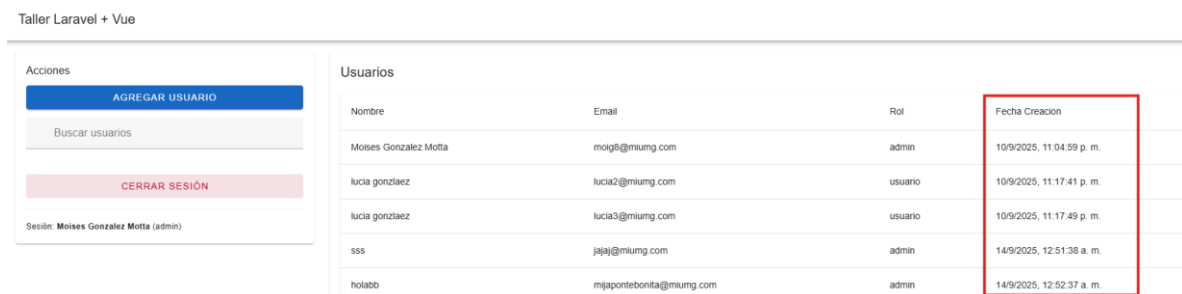
```
src > views > UsersList.vue > {} template > v-data-table.elevation-1 > template
1 <template>
2   <v-data-table
3     :items="filtered"
4     :headers="headers"
5     :loading="loading"
6     class="elevation-1"
7   >
8     <!-- SLOT PERSONALIZADO PARA MOSTRAR FECHA DE CREACIÓN -->
9     <template #item.created_at="{ item }">
10       {{ new Date(item.created_at).toLocaleString() }}
11     </template>
12
13   <template #no-data>
14     <div class="pa-6 text-center">No hay usuarios para mostrar.</div>
15   </template>
16 </v-data-table>
17 </template>
18
19 <script setup lang="ts">
```

Pero para que el código funcione y no tire error debemos agregar la propiedad `created_at` al tipo `Usuario` en tu `<script setup>` de `UsersList.vue`. y quedaría así



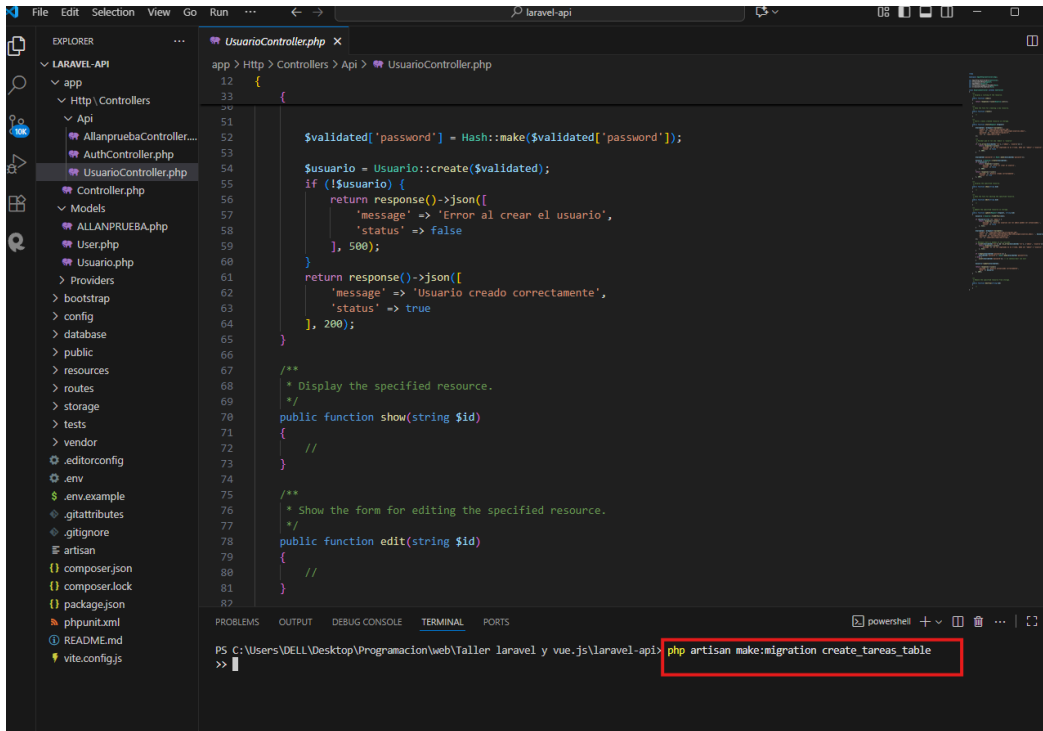
```
1 <template>
2   <v-data-table>
11 </template>
12
13
14   <template #no-data>
15     <div class="pa-6 text-center">No hay usuarios para mostrar.</div>
16   </template>
17 </v-data-table>
18 </template>
19
20 <script setup lang="ts">
21   import { ref, onMounted, computed } from 'vue'
22   import api from '@services/api'
23
24   type Usuario = { id:number; nombre:string; email:string; rol:'admin'|'usuario'; created_at: string }
25
26   const props = defineProps<{ searchTerm?: string }>()
27
28   const items = ref<Usuario[]>([])
29   const loading = ref(false)
30
31   const headers = [
```

Al ya guardar los cambios, podremos ver la fecha ya bien moldeada como se muestra en la siguiente imagen.



Nombre	Email	Rol	Fecha Creación
Moises Gonzalez Motta	moig8@miiumg.com	admin	10/9/2025, 11:04:59 p. m.
lucia gonzlaez	lucia2@miiumg.com	usuario	10/9/2025, 11:17:41 p. m.
lucia gonzlaez	lucia3@miiumg.com	usuario	10/9/2025, 11:17:49 p. m.
sss	jajaj@miiumg.com	admin	14/9/2025, 12:51:38 a. m.
holiab	mijapontebonifa@miiumg.com	admin	14/9/2025, 12:52:37 a. m.

ejecutamos código en el proyecto de laravel para la creación de la tabla de migración de usuarios

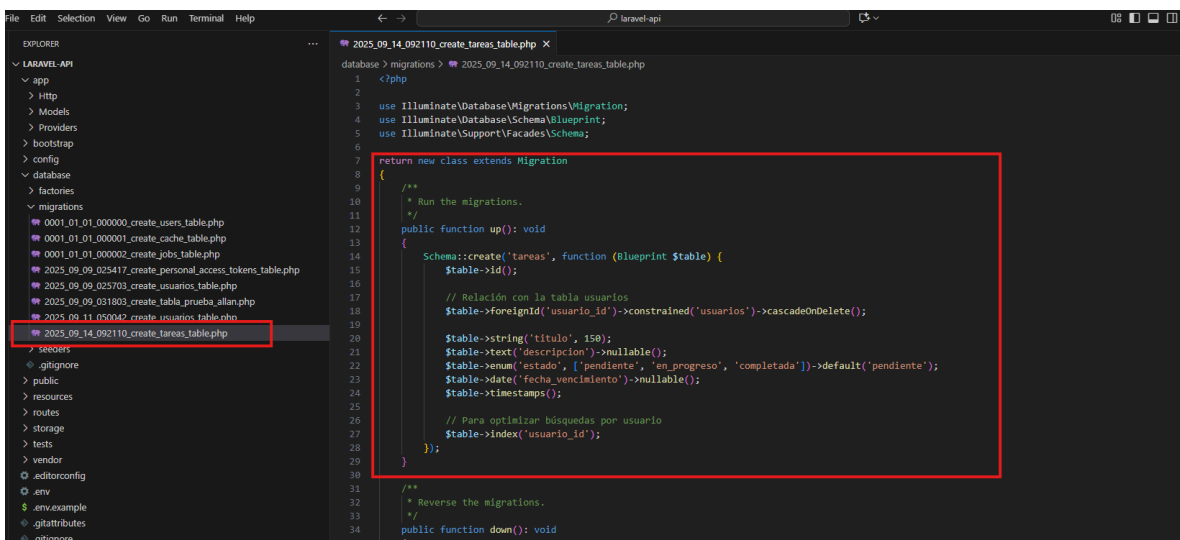


The screenshot shows the VS Code editor with the `UsuarioController.php` file open. The file contains the following code:

```
12 {
33 {
51
52 $validated['password'] = Hash::make($validated['password']);
53
54 $usuario = Usuario::create($validated);
55 if (!$usuario) {
56     return response()->json([
57         'message' => 'Error al crear el usuario',
58         'status' => false
59     ], 500);
60 }
61 return response()->json([
62     'message' => 'Usuario creado correctamente',
63     'status' => true
64 ], 200);
65 }
66
67 /**
68  * Display the specified resource.
69  */
70 public function show(string $id)
71 {
72     //
73 }
74
75 /**
76  * Show the form for editing the specified resource.
77  */
78 public function edit(string $id)
79 {
80     //
81 }
82 }
```

The terminal window at the bottom shows the command `php artisan make:migration create_tareas_table` being executed.

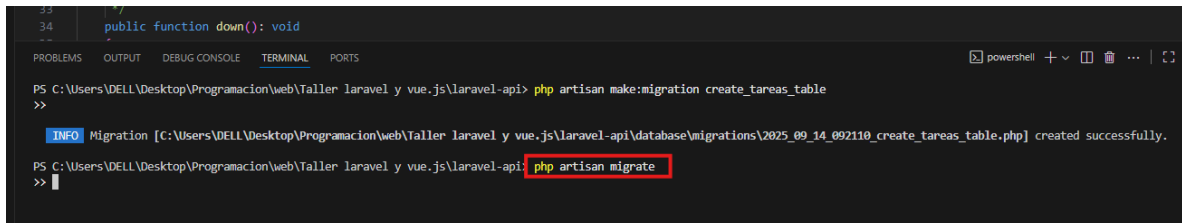
Agregamos el siguiente código



The screenshot shows the VS Code editor with the `2025_09_14_092110_create_tareas_table.php` migration file open. The file contains the following code:

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('tareas', function (Blueprint $table) {
15             $table->id();
16
17             // Relación con la tabla usuarios
18             $table->foreignId('usuario_id')->constrained('usuarios')->cascadeOnDelete();
19
20             $table->string('titulo', 150);
21             $table->text('descripcion')->nullable();
22             $table->enum('estado', ['pendiente', 'en progreso', 'completada'])->default('pendiente');
23             $table->date('fecha_vencimiento')->nullable();
24             $table->timestamps();
25
26             // Para optimizar búsquedas por usuario
27             $table->index('usuario_id');
28         });
29     }
30
31     /**
32      * Reverse the migrations.
33      */
34     public function down(): void
35     {
36     }
```

Luego de eso ejecutamos en la terminal php artisan migrate para crear la tabla en la base de datos



```
33  
34 public function down(): void  
--  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell + - [ ] [ ] [ ] [ ] [ ]  
PS C:\Users\DELL\Desktop\Programacion\web\Taller laravel y vue.js\laravel-api> php artisan make:migration create_tareas_table  
>>  
[INFO] Migration [c:\Users\DELL\Desktop\Programacion\web\Taller laravel y vue.js\laravel-api\database\migrations\2025_09_14_092110_create_tareas_table.php] created successfully.  
PS C:\Users\DELL\Desktop\Programacion\web\Taller laravel y vue.js\laravel-api> php artisan migrate  
>> |
```


- El backend deberá proteger cada **API** del CRUD:
 - Si la petición no incluye un **token válido**, deberá devolver un error **401 Unauthorized**.
 - El frontend (Vue.js) deberá manejar el flujo de login y consumo de APIs con token.

Como ya tengo el login creado a base de las instrucciones del ingeniero moises lo que tocaria hacer seria porteger las rutas de cada api, tengo que proteger las rutas en el proyecto de laravel para esto debo encerrar las rutas que quiero proteger en routes/api.php con el código `Route::middleware('auth:sanctum')`, ya con eso, si alguien intenta acceder a `/api/usuarios/listUsers` sin token, Laravel devolverá: `{"message": "Unauthenticated."}` con código **401 Unauthorized**.

```

1  <?php
2
3  use Illuminate\Http\Request;
4  use Illuminate\Support\Facades\Route;
5  use App\Http\Controllers\Api\AllanpruebaController;
6  use App\Http\Controllers\Api\UsuarioController;
7  use App\Http\Controllers\Api\AuthController;
8  use App\Http\Controllers\TareaController; // ⚡ Asegúrate de importar tu controlador
9
10 // Ruta de prueba publica
11 Route::prefix('usuarios')->group(function() {
12     Route::get('/saludar', [AllanpruebaController::class, 'index']);
13 });
14
15 // ruta agregada modificada 23/09/2025-->ruta publicas de autenticacion
16 Route::post('/login', [AuthController::class, 'login']);
17
18 // ruta agregada modificada 23/09/2025--> Rutas protegidas por Sanctum
19 Route::middleware('auth:sanctum')->group(function () {
20 // Rutas para Usuarios
21 Route::prefix('usuarios')->group(function () {
22     Route::get('/listUsers', [UsuarioController::class, 'index']);
23     Route::post('/addUser', [UsuarioController::class, 'store']);
24     Route::get('/getUser/{id}', [UsuarioController::class, 'show']);
25     Route::put('/updateUser/{id}', [UsuarioController::class, 'update']);
26     Route::delete('/deleteUser/{id}', [UsuarioController::class, 'destroy']);
27 });
28 // NUEVO: Rutas para Tareas
29 Route::prefix('tareass')->group(function () {
30     Route::get('/', [TareaController::class, 'index']); // Listar tareas
31     Route::post('/', [TareaController::class, 'store']); // Crear tarea
32     Route::get('/{id}', [TareaController::class, 'show']);
33     Route::put('/{id}', [TareaController::class, 'update']);
34     Route::delete('/{id}', [TareaController::class, 'destroy']);
35 });
36
37 // Logout
38 Route::post('/logout', [AuthController::class, 'logout']);
39 });
  
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\DELL\Desktop\Programacion\web\Taller laravel y vue.js> cd laravel-api
 PS C:\Users\DELL\Desktop\Programacion\web\Taller laravel y vue.js\laravel-api> |

Como podemos observar en la imagen dentro del archivo api.php tenemos tres partes fundamentales para la seguridad

RUTA PUBLICA: LOGIN

```
Route::post('/login', [AuthController::class, 'login']);
```

Aquí cualquiera puede entrar (no necesita token todavía).

Lo que hace es:

Recibir email y password.

Verificarlos contra la base de datos.

Si son correctos, devuelve un token (gracias a Laravel Sanctum).

Este token es como la llave de acceso para todo lo demás.

RUTAS PROTEGIDAS CON MIDDLEWARE

```
Route::middleware('auth:sanctum')->group(function () {  
    Route::post('/logout', [AuthController::class, 'logout']);  
});
```

Todo lo que está dentro de este group está protegido por Sanctum.

El middleware auth:sanctum revisa que la petición traiga el header:

Authorization: Bearer TU_TOKEN

Si no lo trae, Laravel responde: { "message": "Unauthenticated."} con código 401 **Unauthorized**.

En resumen protege las rutas de usuarios y tareas como vemos en el esquema de abajo

/api/login → 🔓 pública (sirve para obtener el token).

/api/logout → 🔒 protegida (requiere token).

/api/usuarios/* → 🔒 protegida (listUsers, addUser, updateUser, deleteUser, etc.).

/api/tareas/* → 🔒 protegida (listar, crear, actualizar, eliminar tareas).

Entonces sí: con esa configuración, los endpoints de usuarios y tareas ya no se pueden usar sin estar autenticado

Ejemplo

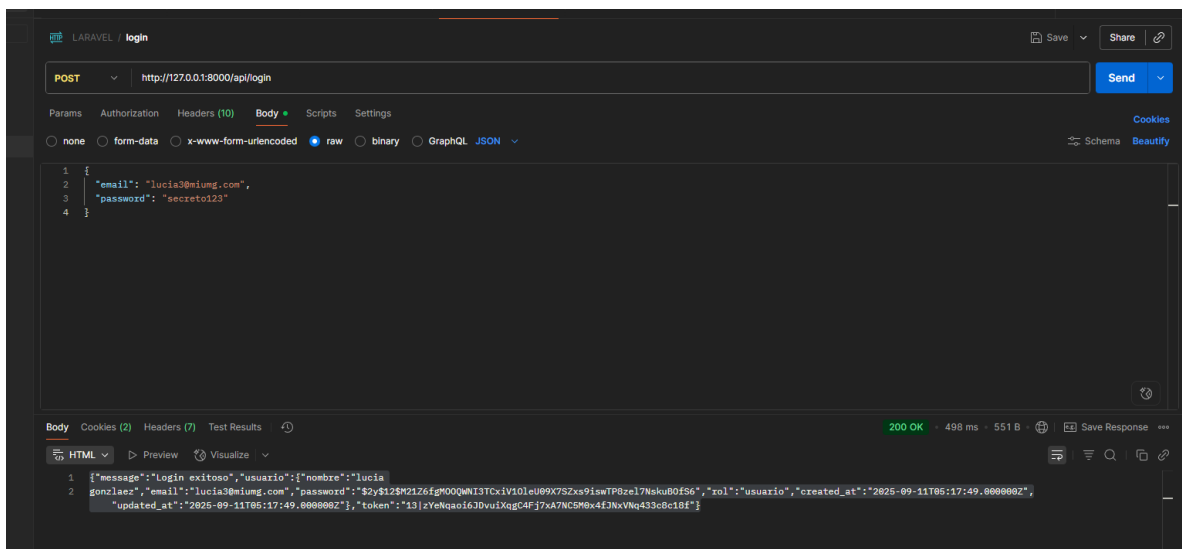
Si intentara entrar a `http://127.0.0.1:8000/api/usuarios/listUsers` sin token, me daría: { "message": "Unauthenticated."}

Y si lo hago con el Authorization: Bearer **token_extraido**, entonces sí devuelve la información.

Como podemos observar en el post nos devuelve un token

Ejemplo

1.primeramente deberemos en login



Con el get al usuario con correo lucia3@miumg.com

Con contraseña secreto123

POST /api/login

Qué hace:

Recibe un email y password.

Busca al usuario en la base de datos.

Valida que la contraseña sea correcta (usando `Hash::check`).

Si todo está bien, genera un token de acceso usando Laravel Sanctum.

Respuesta:

Devuelve un JSON con:

El mensaje de éxito.

Los datos del usuario autenticado.

El token para usar en futuras peticiones.

Ejemplo esperado

```
{
  "message": "Login exitoso",
  "usuario": {
    "nombre": "lucia gonzlaez",
    "email": "lucia3@miumg.com",
    "password":
"$2y$12$M21Z6fgMOOQWNI3TCxiV1OleU09X7SZxs9iswTP8zel7NskuBOfS6",
    "rol": "usuario",
    "created_at": "2025-09-11T05:17:49.000000Z",
    "updated_at": "2025-09-11T05:17:49.000000Z"
  },
  "token": "13|zYeNqaoi6JDvuiXqgC4Fj7xA7NC5M0x4fJNxVNq433c8c18f"
}
```

2. GET /api/usuarios/listUsers

Qué hace:

Devuelve todos los registros de la tabla usuarios.

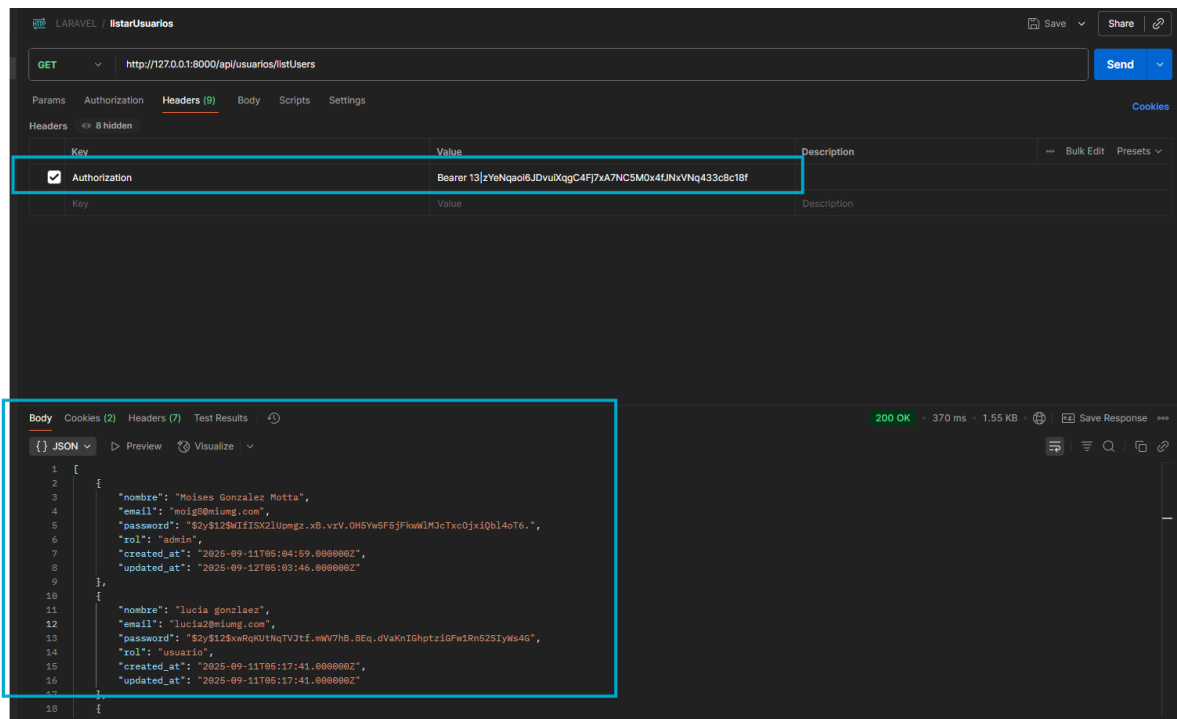
Está protegida con middleware auth:sanctum, por lo que requiere enviar el token válido.

Cómo se prueba:

En Postman (o cURL) se envía un GET con el header: Authorization: Bearer TU_TOKEN

Si el token es válido → devuelve lista de usuarios.

Si el token falta o es incorrecto → responde con: { "message": "Unauthenticated." }



Como vemos ya tenemos toda la seguridad, ahora pegare una imagen en la cual veremos la ruta <http://127.0.0.1:8000/api/usuarios/listUsers> en el navegador en el cual nos deberá tirar todo los usuarios debido a que no tendrá protección

Código api.php sin seguridad

```
... api.php M X E informacion.txt
laravel-api > routes > api.php
1 <?php
2
3 use Illuminate\Http\Request;
4 use Illuminate\Support\Facades\Route;
5 use App\Http\Controllers\Api\AllanpruebaController;
6 use App\Http\Controllers\Api\UsuarioController;
7 use App\Http\Controllers\Api\AuthController;
8 use App\Http\Controllers\TareaController;
9
10 // Ruta de prueba publica
11 Route::prefix('usuarios')->group(function() {
12     Route::get('/saludar', [AllanpruebaController::class, 'index']);
13 });
14
15 // Rutas para Usuarios (ahora todas son públicas)
16 Route::prefix('usuarios')->group(function () {
17     Route::get('/listUsers', [UsuarioController::class, 'index']);
18     Route::post('/addUser', [UsuarioController::class, 'store']);
19     Route::get('/getUser/{id}', [UsuarioController::class, 'show']);
20     Route::put('/updateUser/{id}', [UsuarioController::class, 'update']);
21     Route::delete('/deleteUser/{id}', [UsuarioController::class, 'destroy']);
22 });
23
24 // Rutas para autenticación (login seguirá estando público)
25 Route::post('/login', [AuthController::class, 'login']);
26 Route::post('/logout', [AuthController::class, 'logout']); // ya no requiere token
27
28 // NUEVO: Rutas para Tareas (ahora todas son públicas)
29 Route::prefix('tareas')->group(function () {
30     Route::get('/', [TareaController::class, 'index']); // Listar tareas
31     Route::post('/', [TareaController::class, 'store']); // Crear tarea
32     Route::get('/{id}', [TareaController::class, 'show']);
33     Route::put('/{id}', [TareaController::class, 'update']);
34     Route::delete('/{id}', [TareaController::class, 'destroy']);
35 });
36
```

Resultado en el navegador al poner <http://127.0.0.1:8000/api/usuarios/listUsers>

```
127.0.0.1:8000/api/usuarios/listUsers
Apuntes Antares Molay benchi Maps Jitsi Jitsi Agencias Random Papelaria Edicthas Mail Mail Mail Maps Mail Mail
presión con formato estilístico
[{"nombre":"Rafael Gonzalez Rotta","email":"rafael@miang.com","password":"$2y$12$BfF5X2llymg:ab:evVJ979v9f5Jf6wPi0c0jpi00-1ao76","rol":"admin","created_at":"2025-09-11T05:04:59.000000Z","updated_at":"2025-09-11T05:04:59.000000Z"}, {"nombre":"Lucia onlaza","email":"lucia@miang.com","password":"$2y$12$5u8u4U8m[V74f:am7/8a:8iaKni08pti0fwh5253y648","rol":"usuario","created_at":"2025-09-11T05:17:41.000000Z","updated_at":"2025-09-11T05:17:41.000000Z"}, {"nombre":"Lucia onlaza","email":"lucia@miang.com","password":"$2y$12$9L126fgnCOQM13fCxiV101e09X757a91swP8zi17nakub0F56","rol":"usuario","created_at":"2025-09-11T05:17:49.000000Z","updated_at":"2025-09-11T05:17:49.000000Z"}, {"nombre":"tita","email":"tita@miang.com","password":"$2y$12$118R2L3ziif48VkhvId:9p5:8p5gm0p0d0KamC13b8dV1bue","rol":"admin","created_at":"2025-09-14T06:51:18.000000Z","updated_at":"2025-09-14T06:51:18.000000Z"}, {"nombre":"holabé","email":"holabé@miang.com","password":"$2y$12$1xiq08dwyf28ydkz24j:fJ07nJk5Ch/dA2d94F5985fppU/V6","rol":"admin","created_at":"2025-09-14T06:52:37.000000Z","updated_at":"2025-09-14T06:52:37.000000Z"}, {"nombre":"Allan","email":"Allan@miang.com","password":"$2y$12$1fRaXz5982m0vrtt0Kyx0kthMpolYed/Vk9pCU8tq0064Ca","rol":"admin","created_at":"2025-09-14T07:35:24.000000Z","updated_at":"2025-09-14T07:35:24.000000Z"}]
```

Ojo para que me tirara el código {"message": "Unauthenticated."} a la hora de que no use el token crearemos un middleware global para esto deberemos ejecutar en la terminal estando en el proyecto de laravel el siguiente código **php artisan make:middleware ForceJsonResponse** esto crea la siguiente ruta app/Http/Middleware/ForceJsonResponse.php. editamos el archivo poniendo completamente el siguiente código

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;

class ForceJsonResponse
{
    public function handle(Request $request, Closure $next)
    {
        $request->headers->set('Accept', 'application/json');
        return $next($request);
    }
}
```

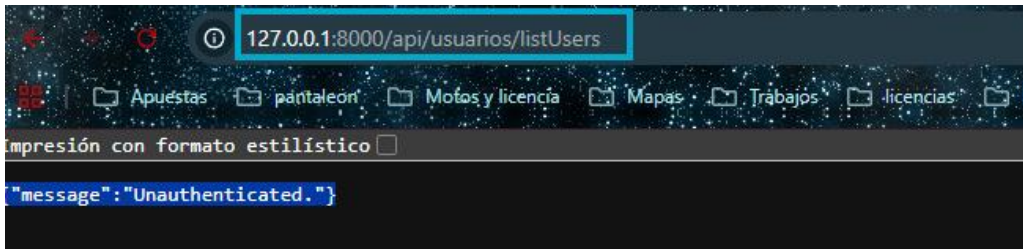
Luego tengo que irme a la siguiente ruta bootstrap/app.php y remplazar el siguiente código

```
return Application::configure(basePath: dirname(__DIR__))
    ->withRouting(
        web: __DIR__.'/../routes/web.php',
        api: __DIR__.'/../routes/api.php',
        commands: __DIR__.'/../routes/console.php',
        health: '/up',
    )
    ->withMiddleware(function (Middleware $middleware) {
        //
    })
    ->withExceptions(function (Exceptions $exceptions) {
        //
    })
    ->create();
```

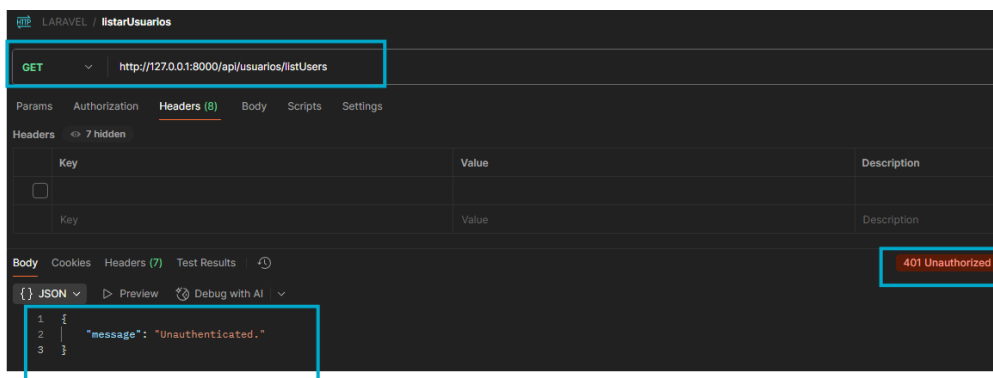
reemplazamos lo que esta en cuadrado celeste por lo siguiente Así Laravel sabrá que todas nuestras peticiones API deben devolver JSON en lugar de intentar redirigir.

```
->withMiddleware(function (Middleware $middleware) {
    $middleware->append(\App\Http\Middleware\ForceJsonResponse::class);
})
```

Con esto, cuando entremos a una ruta protegida (/api/usuarios/listUsers) sin token, Laravel ya no intentará redirigir y ahora sí debería mostrarnos el error 401:



Como vemos en la imagen de arriba nos tira el json del mensaje de error al entrar a la ruta protegida, en la imagen de abajo se aprecia de igual manera el error pero en postman, al no darle un token no nos muestra la lista de usuarios y nos manda el json de error



En resumen lo que hicimos.

Problema

1. Cuando se accedía a una ruta protegida sin token, Laravel intentaba redirigir al login (pensando en una app web con vistas Blade).
2. Como el proyecto es una API, no tiene vistas de login, entonces esa redirección causaba el error 500 Internal Server Error en lugar de devolver el 401 Unauthorized.

Solucion

1. Se implemento ForceJsonResponse Su trabajo es Forzar a que todas las peticiones respondan en JSON, aunque Laravel haya querido responder con HTML o una redirección. Esto significa: “No importa lo que pase, Laravel, recuerda que esto es una API y siempre responde en JSON”.
2. Se edito el archivo bootstrap/app.php En Laravel 11 (y 10 en algunos casos), ya no se registra middleware en Kernel.php, sino en bootstrap/app.php, que es el nuevo punto de entrada para configurar la aplicación.

Al agregar allí ForceJsonResponse, garantizamos que: Todas las rutas de tu API pasan primero por este middleware, Ya no intentan devolver HTML o redirecciones.

Cuando falta un token se responde siempre en JSON con: { "message": "Unauthenticated." } y con **código 401 Unauthorized**.