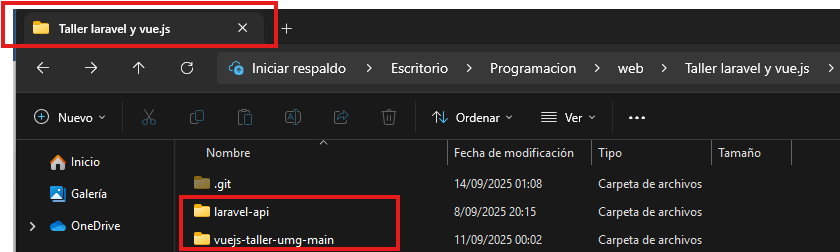
Nombre: Allan Guamuch

Carne:2990-19-22192

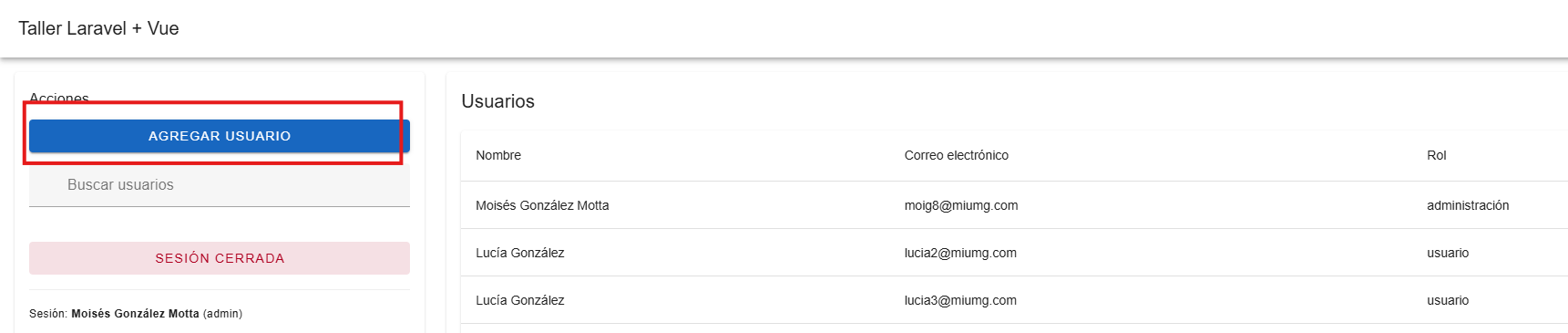
Curso: Desarrollo Web

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

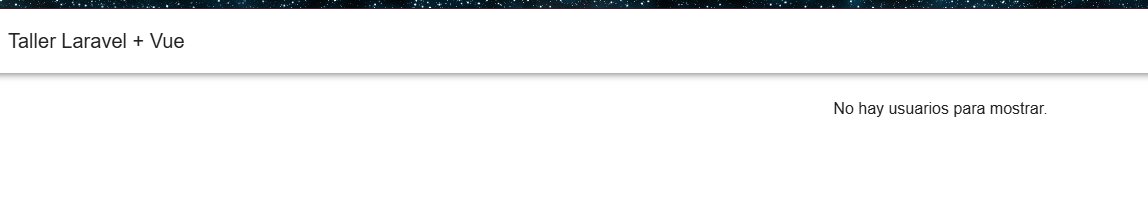
Para este taller cree una carpeta llamada Taller laravel y vue.js en la cual están las carpetas laravel-api y vuejs-taller-umg-main uno que es el proyecto **Laravel (backend)** y el otro **Vue.js (frontend)**



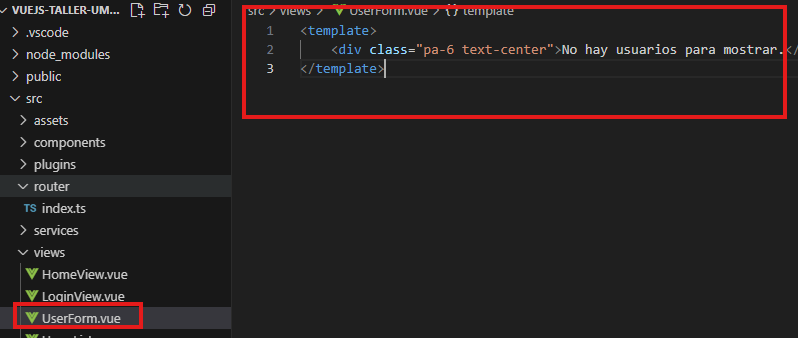
Al ejecutar el proyecto vue logramos ver los usuarios que ya tenemos en nuestra base de datos y también un botón llamado AGREGAR USUARIO

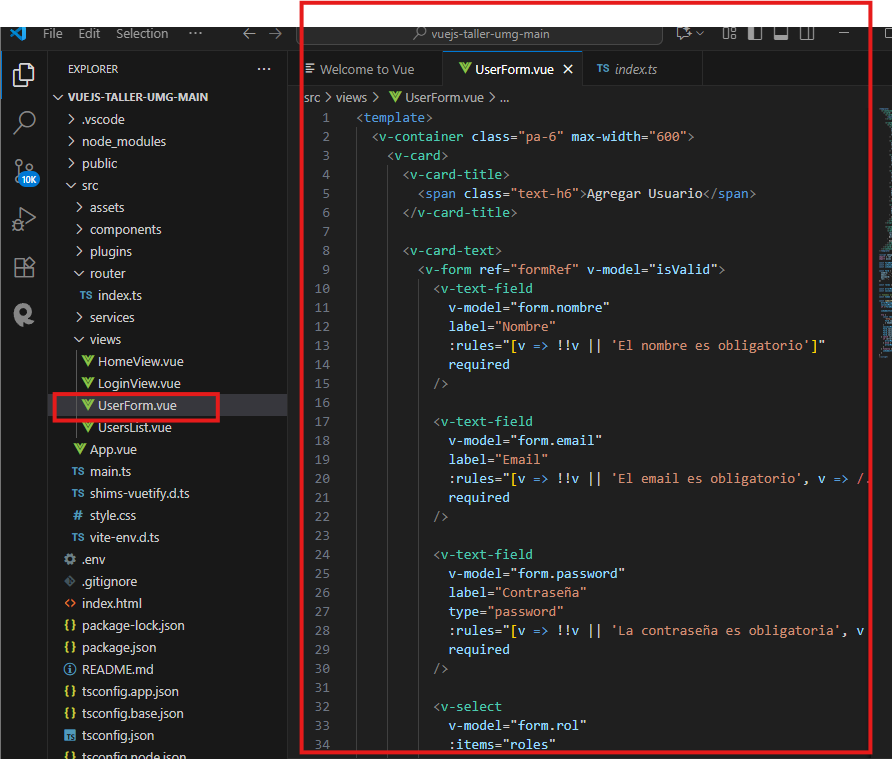


Al apretar el botón Agregar usuario nos redirecciona a otra pagina en la cual solo nos muestra un mensaje

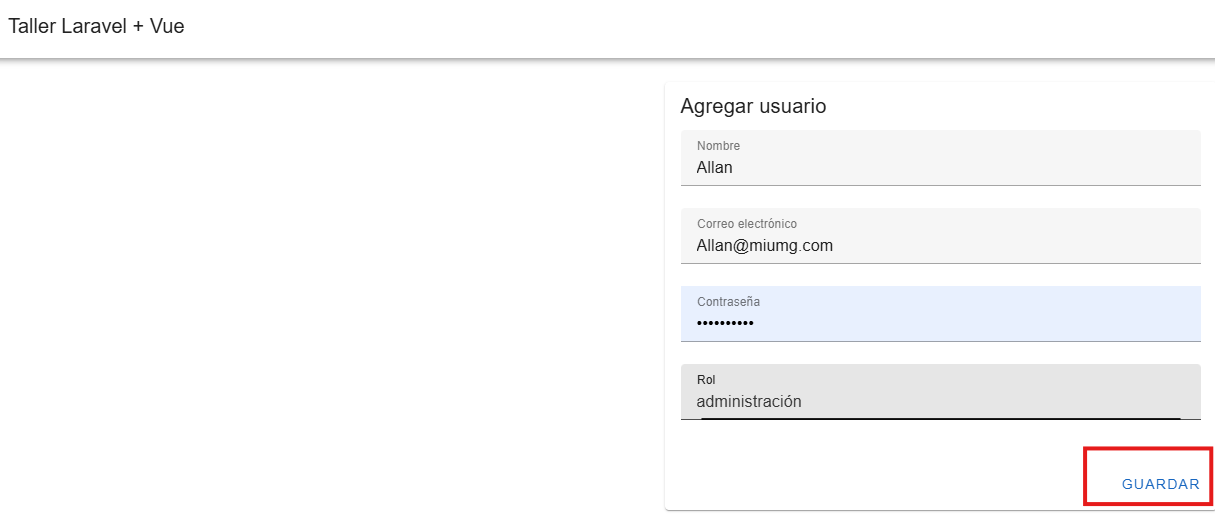


El mensaje proviene del proyecto vue de views/userform.vue, como primer tarea es modificar ese código para que podar agregar usuarios desde el navegador



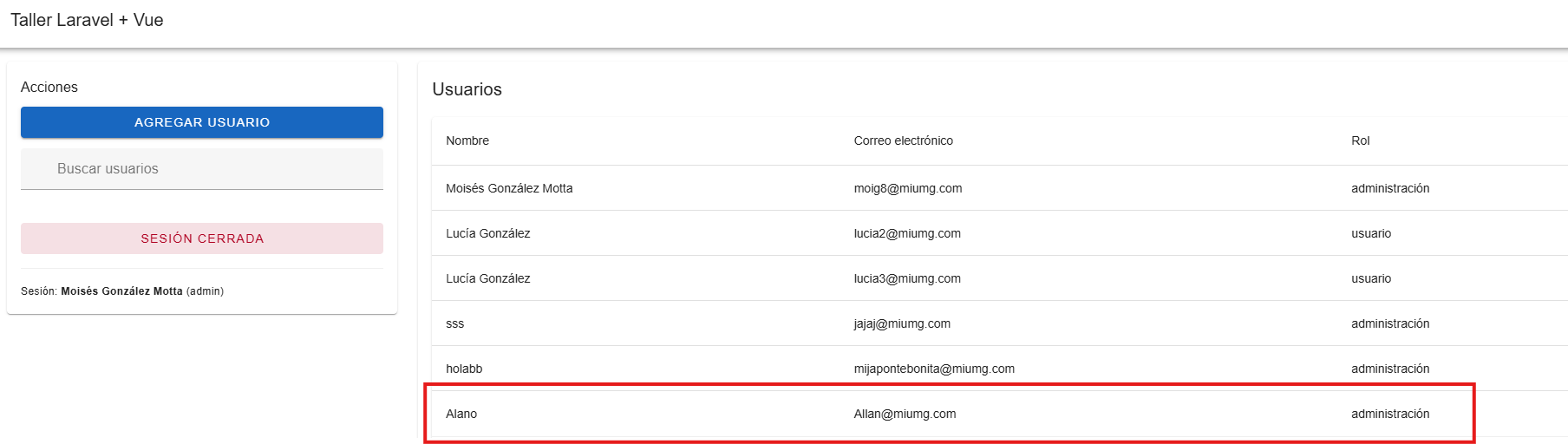
El código modificado quedaría de la siguiente manera

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

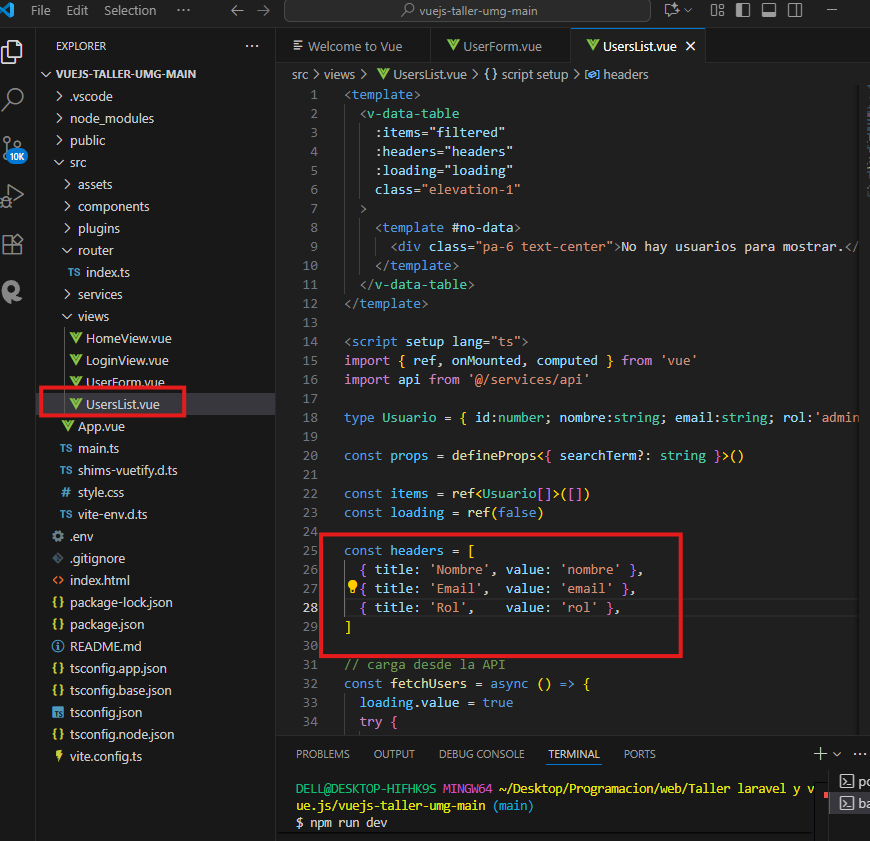


Al dar clic al botón guardar podremos ver una pequeña notificación de que se ha agregado correctamente el usuario

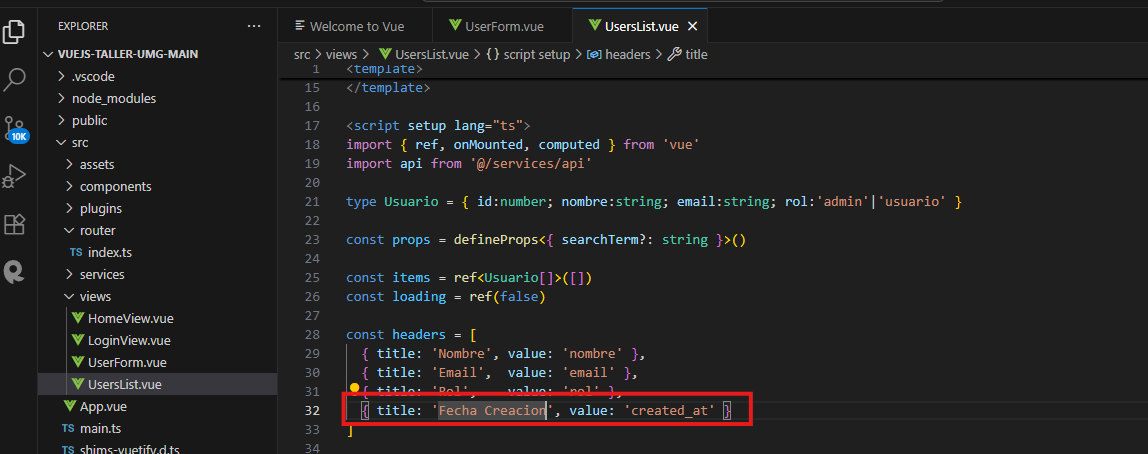


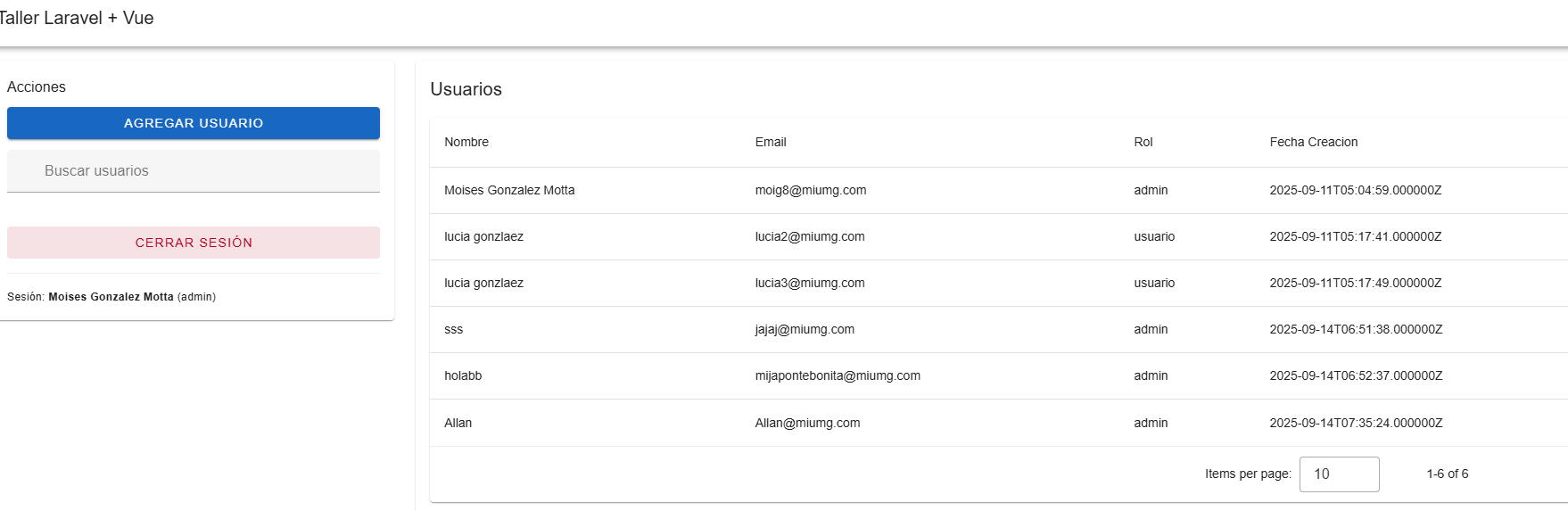
Al validar la ventana de usuarios podemos ver que efectivamente se agregó el usuario añadido

Como vemos en la vista solo podemos visaulizar 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 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.

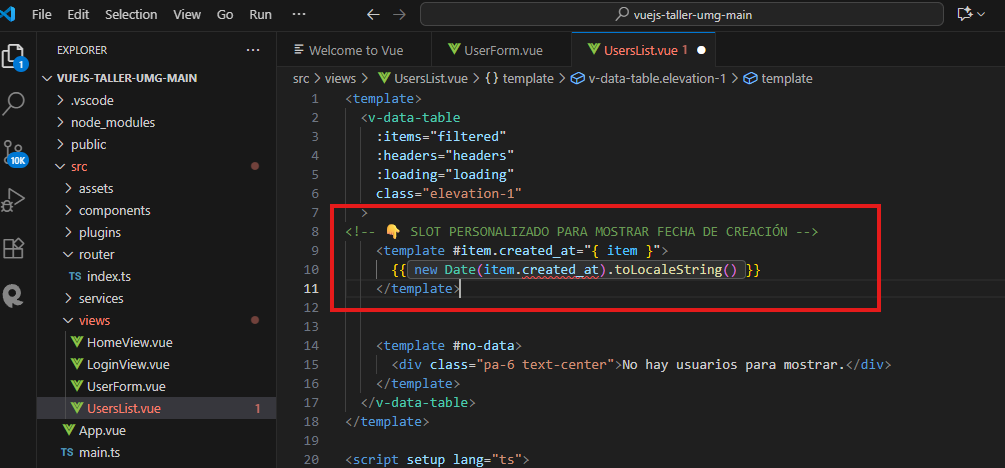


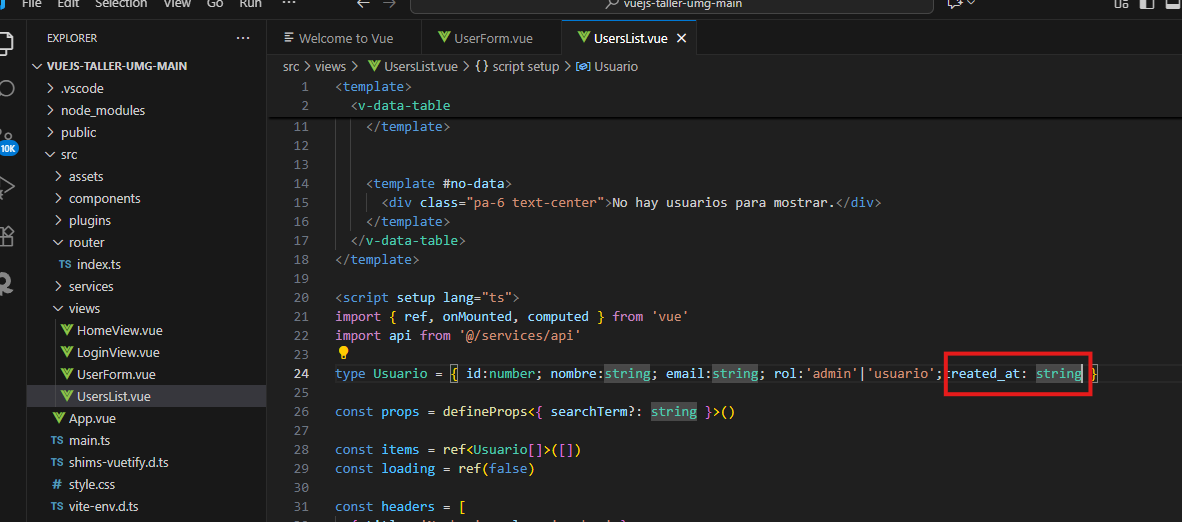
Al agregar el código para visualizar la fecha de usuario quedaría de la siguiente manera



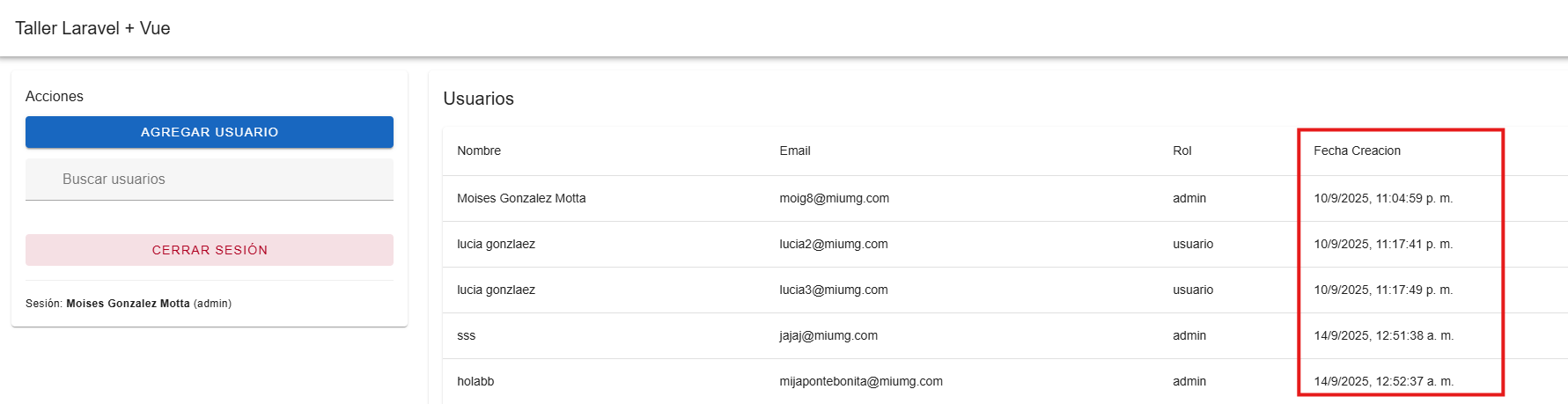


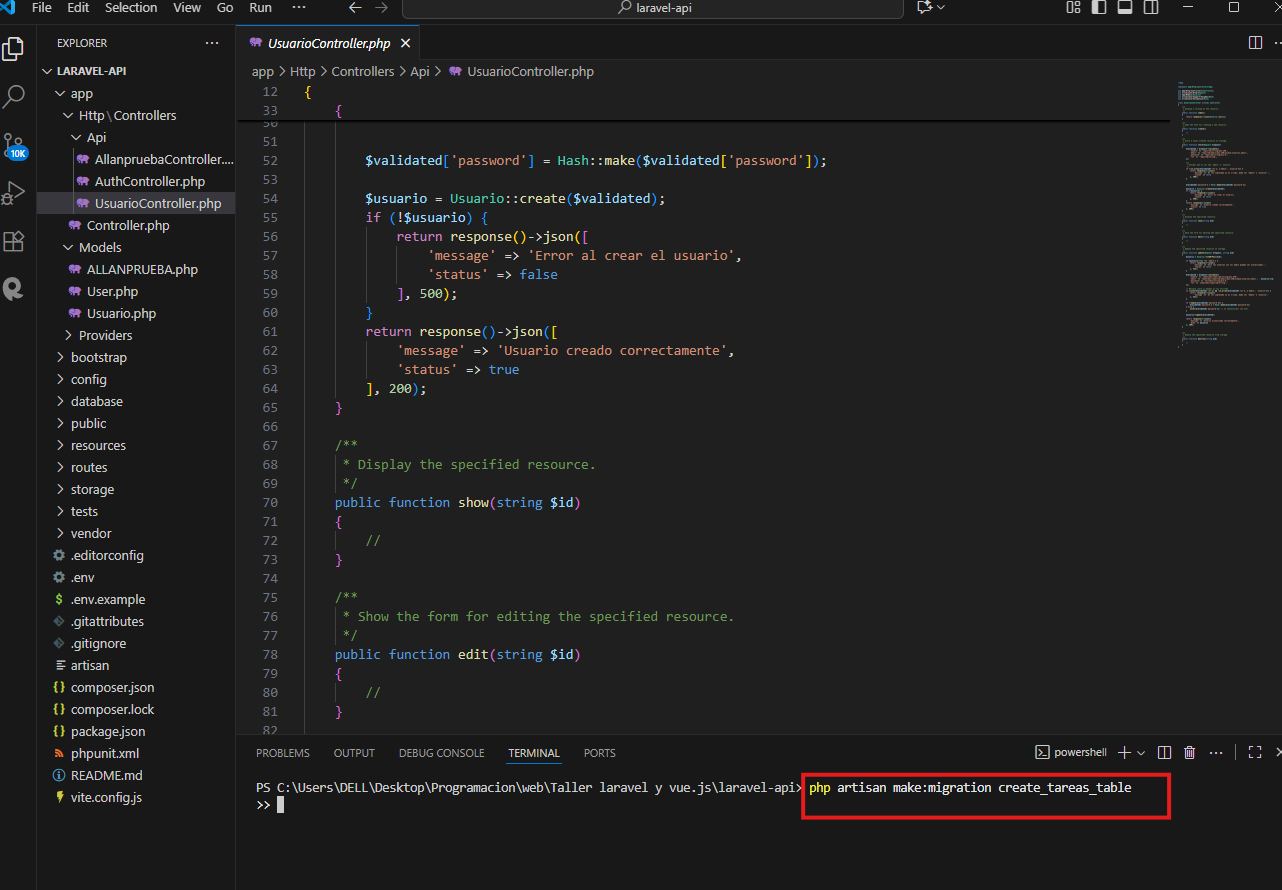
Como vemos ya tenemos la fecha de creación, solo que hay un detalle la fecha no está compuesta para poder componerla y que se mire bonita deberemos agregar otro pedazo de código

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

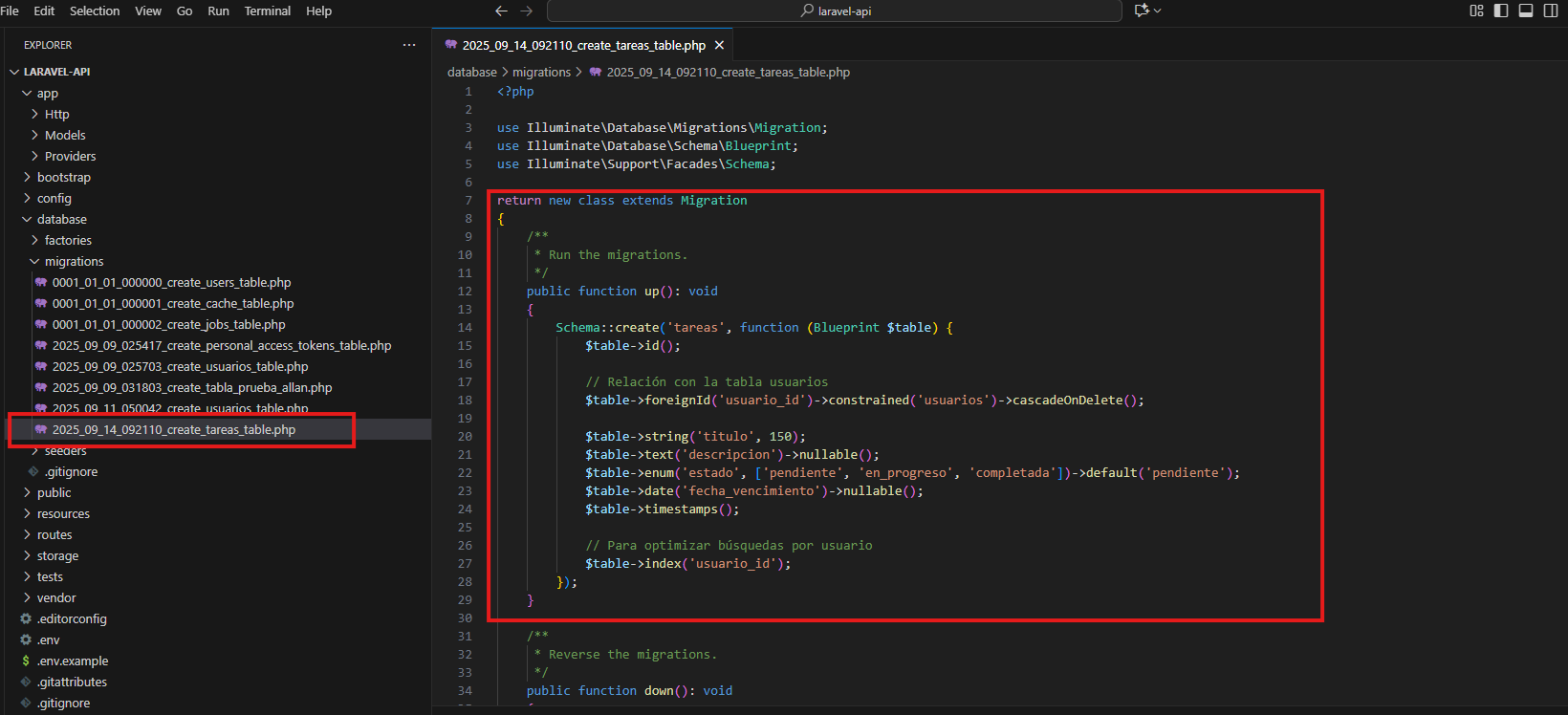
Pero para que el código funcione y no tire error debemos agregar la propiedad created\_at al tipo Usuario en tu <script setup> de UsersList.vue. y quedaría asi

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

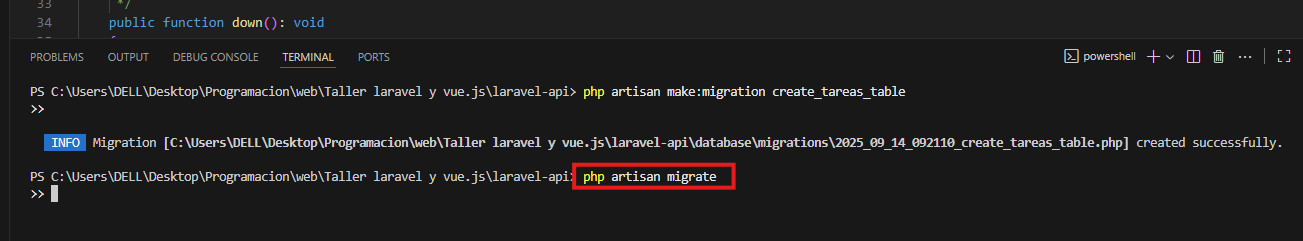


ejecutamos código en el proyecto de laravel para la creación de la tabla de migración de usuarios

Agregamos el siguiente código

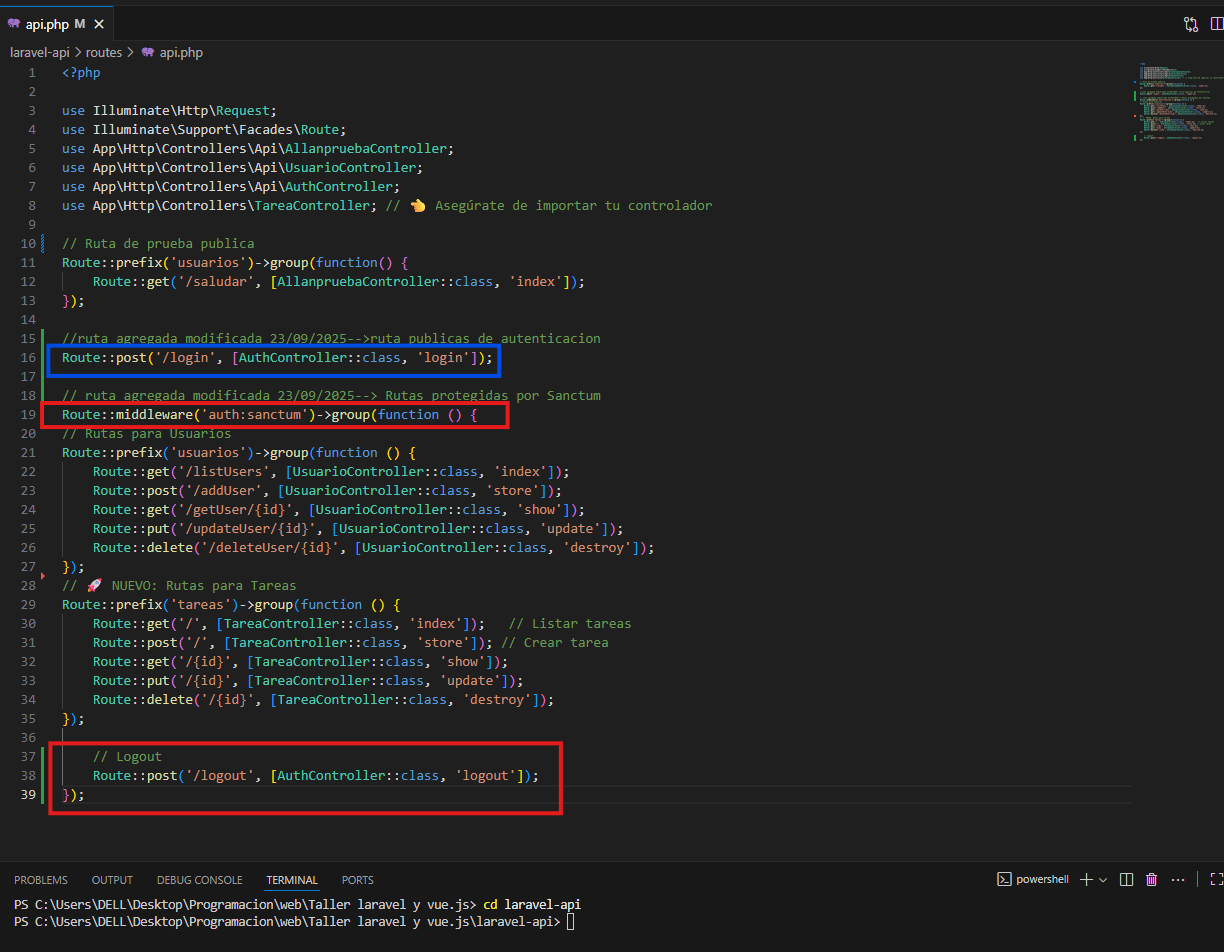


Luego de eso ejecutamos en la terminal php artisan migrate para crear la tabla en la base de datos



* 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**.

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

Aquí cualquiera puede entrar (no necesita token todavía).

**RUTA PUBLICA: LOGIN**Route::post('/login', [AuthController::class, 'login']);

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

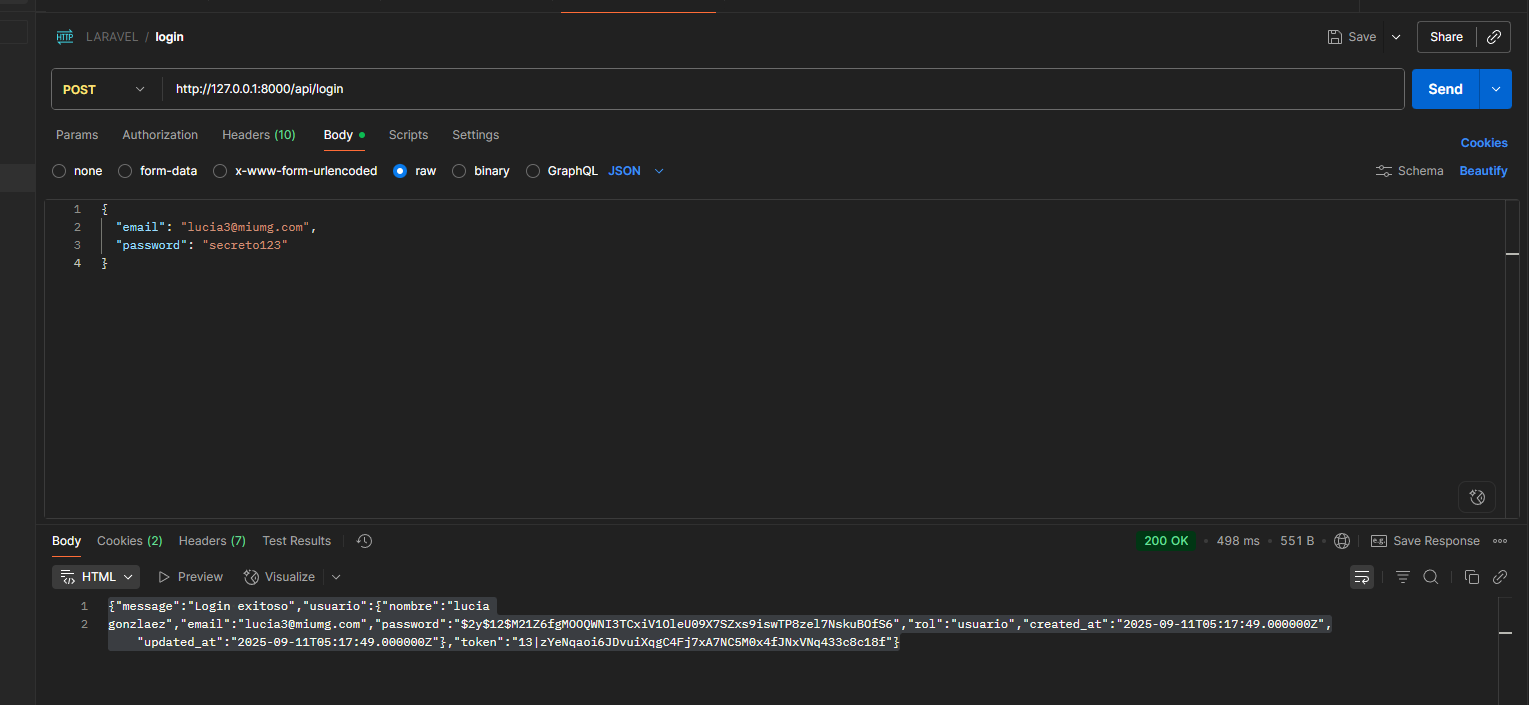
Si intentara entrar a http://127.0.0.1:8000/api/usuarios/listUsers sin token, me daria: { "message": "Unauthenticated."}

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.primero deberemos en login



Con el get al usuario con correo [lucia3@miumg.com](mailto: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.

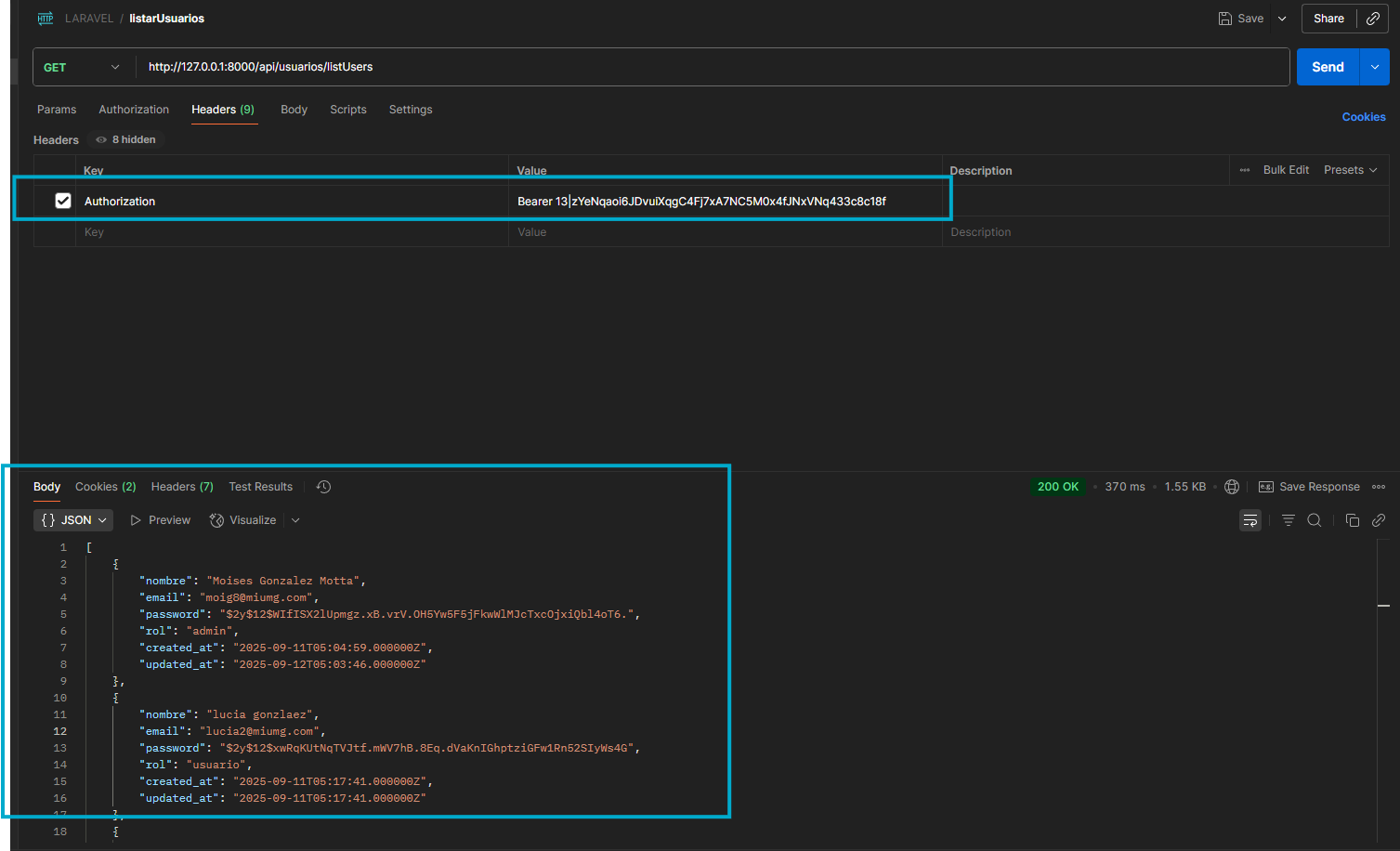
Está protegida con middleware auth:sanctum, por lo que requiere enviar el token válido.

**Cómo se prueba**:

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

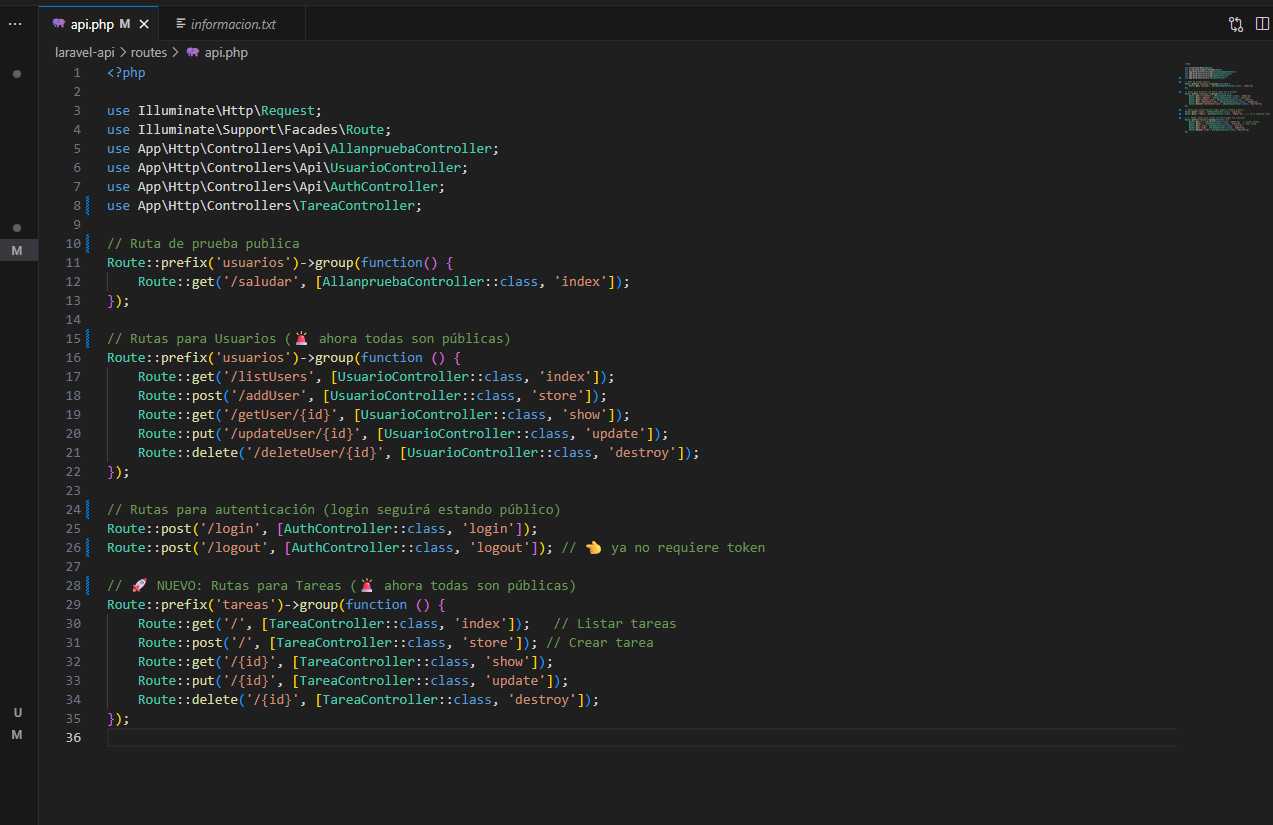
Si el token es válido → devuelve lista de usuarios.

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

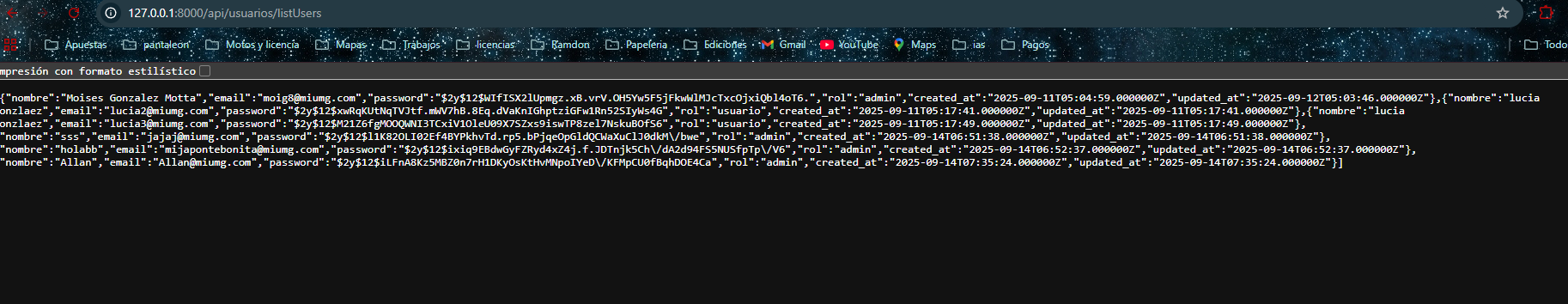


Como vemos ya tenemos toda la seguridad, ahora pegare una imagen en la cual veremos la ruta <http://127.0.0.1:8000/api/usuarios/listUsers> en el navegador en el cual nos deberá tirar todo los usuarios debido a que no tendrá protección

Código api.php sin seguridad

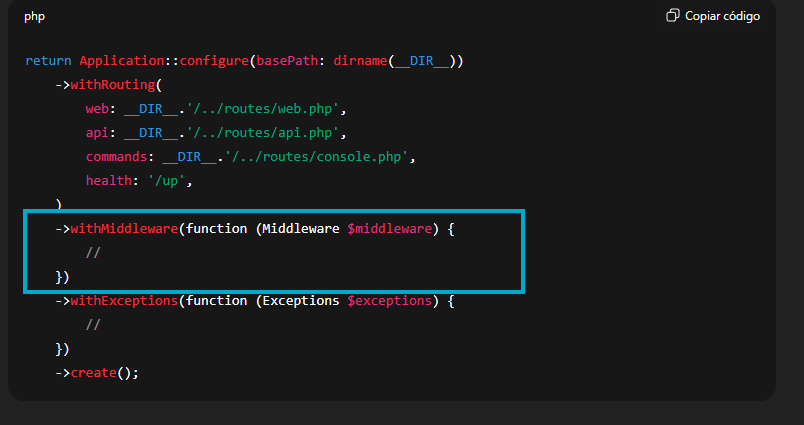


Resultado en el navegador al poner <http://127.0.0.1:8000/api/usuarios/listUsers>

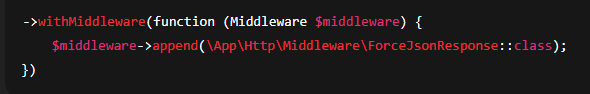


Ojo para que me tirara el código {"message": "Unauthenticated."} a la hora de que no use el token crearemos un middleware global para esto deberemos ejecutar en la terminal estando en el proyecto de laravel el siguiente código **php artisan make:middleware ForceJsonResponse** esto crea la siguiente ruta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

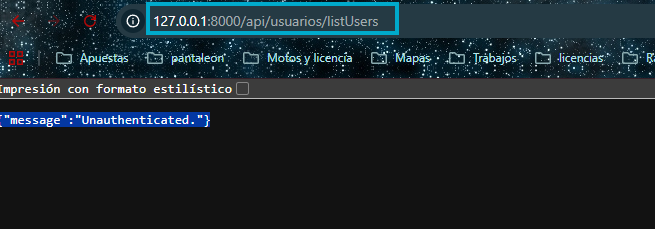


Luego tengo que irme a la siguiente ruta bootstrap/app.php y remplazar el siguiente código

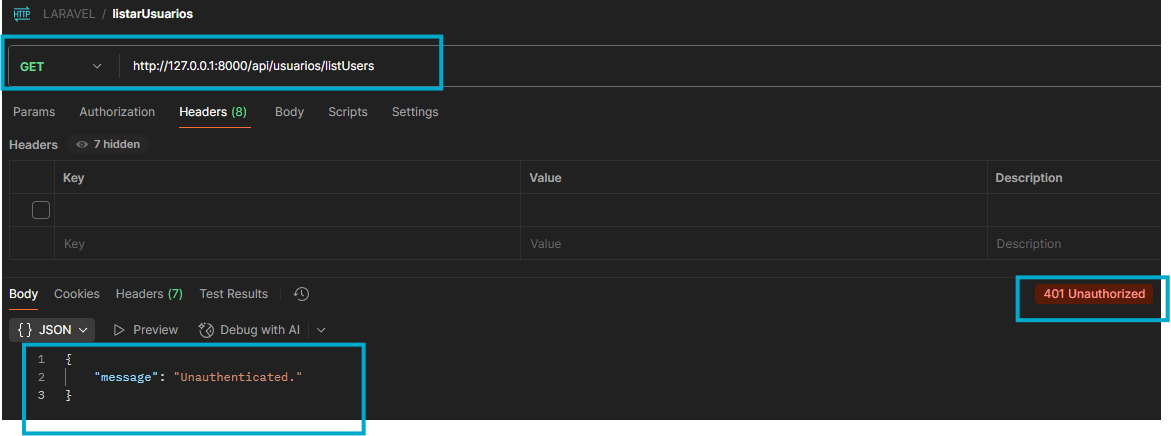
remplazamos lo que esta en cuadrado celeste por lo siguiente Así Laravel sabrá que todas nuestras peticiones API deben devolver JSON en lugar de intentar redirigir.



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 muesta la lista de usuarios y nos manda el json de error



En resumen lo que hicimos.

Problema

1.Cuando se accedia 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**.