Universidad Mariano Gálvez de Guatemala

Ingeniería en Sistemas de Información

8vo. Semestre

Ciclo 2025

Tema:

**Documentación – Segundo Parcial**

**Proyecto: Laravel + Vue.js – Multitenancy + Autenticación**

Desarrollo Web

Ing. Walter Cordova

Edgar Fernando Heredia Ramos 1690-22-2199

Fecha:

*14/08/2025*

*Dolores, Peten*

**INDICE**

[1 Introducción 3](#_Toc209868942)

[2 Marco Teorico 4](#_Toc209868943)

[2.1 Autenticación de Usuario 🔐 4](#_Toc209868944)

[2.1.1 Estrategia 4](#_Toc209868945)

[2.1.2 Implementación 4](#_Toc209868946)

[2.1.3 Diagrama de Componentes (PlantUML) 6](#_Toc209868947)

[2.1.4 Diagrama de Secuencia del Login (PlantUML) 7](#_Toc209868948)

[2.1.5 Funcionamiento del Proyecto: 8](#_Toc209868949)

[2.2 Arquitectura Multitenant 🏢 9](#_Toc209868950)

[2.2.1 Concepto 9](#_Toc209868951)

[2.2.2 Configuración en Laravel 9](#_Toc209868952)

[2.2.3 Migraciones separadas 10](#_Toc209868953)

[2.2.4 Diagrama de Componentes (PlantUML) 11](#_Toc209868954)

[2.2.5 Diagrama de Secuencia (petición protegida con token cruzado) 12](#_Toc209868955)

[2.2.6 Cómo se provisiona un nuevo tenant 12](#_Toc209868956)

[3 Conclusiones 13](#_Toc209868957)

# Introducción

Este proyecto tiene como propósito aplicar buenas prácticas de seguridad y escalabilidad en una aplicación Laravel + Vue.js.

Se realizaron dos grandes mejoras:

1. **Autenticación de usuarios** mediante Laravel Sanctum.
2. **Implementación de arquitectura Multitenant** con bases de datos independientes por cliente (tenant).

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.La entrega incluye tanto la parte técnica (código y configuración) como la documentación de problemas y soluciones encontrados en el proceso.

# Marco Teorico

## Autenticación de Usuario 🔐

### Estrategia

La autenticación se implementó usando **Laravel Sanctum**, ya que permite emitir tokens para peticiones API. Esto garantiza que el backend solo responda a usuarios autenticados.

* Protegimos el CRUD con auth:sanctum: si la petición no trae token válido → **401 Unauthorized**.
* El **frontend (Vue)**:
  + Tiene un **form de login** que llama a /api/login.
  + Guarda token y user en localStorage.
  + Un **interceptor Axios** agrega Authorization: Bearer <token> en **todas** las peticiones.

Si el backend responde **401**, limpia sesión y redirige a /login.

### Implementación

1. Se instaló Sanctum y se publicaron las migraciones.
2. En el archivo **app/Models/User.php**, se agregó el trait HasApiTokens:

#### Login

* 1. El usuario envía email y password a POST /api/login.
  2. El backend busca el usuario y verifica el hash de la contraseña.
  3. Si es correcto, **Sanctum** crea un registro en personal\_access\_tokens y la API devuelve { usuario, token }.

#### Consumo de APIs protegidas

* 1. El frontend envía Authorization: Bearer <token>.
  2. El middleware auth:sanctum valida el token contra la tabla personal\_access\_tokens.
  3. Si es válido → continúa al controlador. Si no → **401**.

#### Logout

* 1. La API revoca el **token actual** (o todos) y el frontend borra sesión.

1. Texto

   El contenido generado por IA puede ser incorrecto.Captura de pantalla de un celular

   El contenido generado por IA puede ser incorrecto.Se creó un controlador de autenticación:
2. Texto

   El contenido generado por IA puede ser incorrecto.En **routes/api.php** se protegieron rutas:

### PlantUML diagramDiagrama de Componentes (PlantUML)

### PlantUML diagramDiagrama de Secuencia del Login (PlantUML)

### Funcionamiento del Proyecto:

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación, Sitio web

El contenido generado por IA puede ser incorrecto.

## Arquitectura Multitenant 🏢

### Concepto

El **multitenancy** permite que una misma aplicación atienda múltiples clientes (tenants), manteniendo **aislados sus datos**.  
En nuestro proyecto:

* landlord\_db: controla los tenants registrados.
* empresa1\_db, empresa2\_db: contienen los datos de cada cliente.
* Agregamos una **BD central (“landlord”)** con la tabla tenants (name, subdomain, db\_\*).
* Creamos un **middleware** IdentifyTenant que corre **antes** de la autenticación:
  1. Lee el **subdominio** del host (o X-Tenant en desarrollo).
  2. Busca el tenant en landlord.tenants.
  3. **Cambia la conexión** de la app a la BD del tenant (config dinámica + DB::reconnect).

Todas las consultas (incluyendo **Sanctum**) se ejecutan en la **BD del tenant activo**: empresa1\_db, empresa2\_db, … → **aislamiento real de datos y tokens**

### Configuración en Laravel

**Texto

El contenido generado por IA puede ser incorrecto.**Se modificó **config/database.php** para definir múltiples conexiones:

### Migraciones separadas

Se creó estructura:

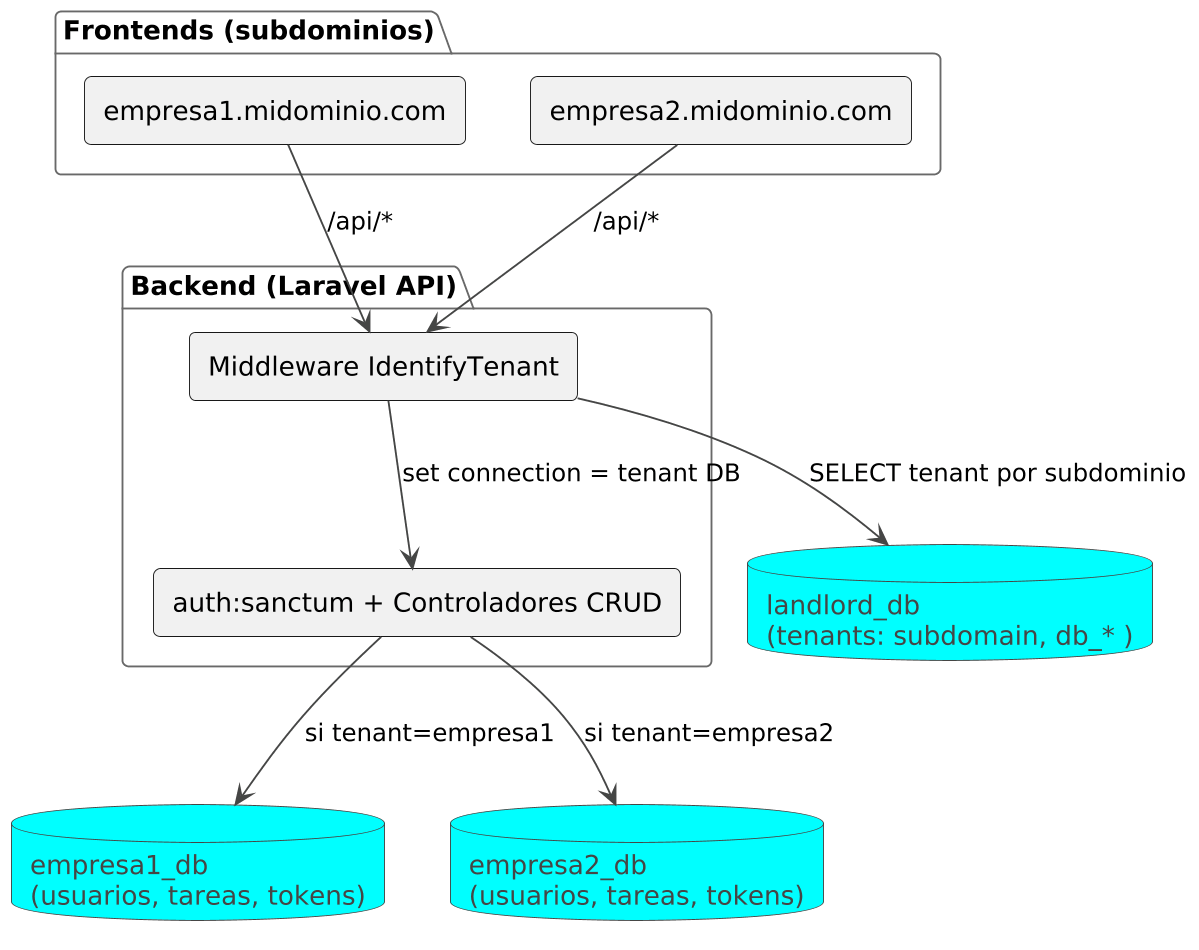
* database/migrations\_landlord/ → tablas globales.
* database/migrations/ → migraciones de cada tenant.

Texto

El contenido generado por IA puede ser incorrecto.Migración en base central:

1. Llega una petición a https://empresa1.midominio.com/api/....
2. IdentifyTenant toma **empresa1**, consulta landlord.tenants, y setea la conexión a empresa1\_db.
3. Luego corre auth:sanctum **contra esa BD del tenant** (donde están sus personal\_access\_tokens).
4. El controlador del CRUD trabaja sobre las tablas del **tenant actual**.
5. Si intentas usar un token de empresa1 en empresa2 → no existe → **401**.

### Diagrama de Componentes



### PlantUML diagramDiagrama de Secuencia

### Cómo se provisiona un nuevo tenant

1. **Crear BD** empresaN\_db.
2. **Insertar** la fila en landlord.tenants con subdomain y credenciales de esa BD.
3. php artisan tenant:migrate para crear tablas del app en empresaN\_db.
4. Crear un **admin** en esa BD (Tinker o endpoint temporal).
5. Apuntar **DNS/subdominio** empresaN.midominio.com al mismo backend (Nginx/Apache/ALB).

# Conclusiones

* Se logró integrar un **sistema de autenticación robusto** con Sanctum.
* Se implementó **multitenancy con bases de datos separadas**, asegurando el aislamiento.
* El proyecto ahora es **seguro, escalable y listo para producción**.
* La experiencia en el despliegue permitió reforzar conocimientos en **servidores Linux, PHP, Laravel y Vue.js**.

*CARNET: 1690-22-2199*

*NOMBRE: EDGAR FERNANDO HEREDIA RAMOS*

*SECCION: REMOTA "5"*

*LINK REPOSITORIOS:*

*BACKEND:*

[*https://github.com/Fernando-20040/Backend\_Taller.git*](https://github.com/Fernando-20040/Backend_Taller.git)

*FRONTEND:*

[*https://github.com/Fernando-20040/Frontend\_Taller.git*](https://github.com/Fernando-20040/Frontend_Taller.git)

VIDEO EXPLICATIVO: <https://drive.google.com/drive/folders/1kgb08bSkppUTotvhsvdNuECRpQn82Ejk?usp=sharing>