Perfecto, Chema. Ya extraje el frontend del proyecto. Ahora te detallo un **paso a paso completo** para implementar el **frontend con firma digital integrada usando SealSign** y otros escenarios de autenticación avanzada.

**✅ Objetivo**

* Configurar el frontend Vue.js del proyecto PlatSentinel.
* Integrar un sistema de **firma digital de documentos** para usuarios autenticados.
* Preparar el entorno de pruebas para **escenarios futuros** como firmas masivas o validación legal.

**🧱 Requisitos previos**

**Necesitas:**

* Frontend funcionando (Vue 3 + Router)
* Backend que valide JWT (como el que ya tenemos)
* Acceso a **API de SealSign** (usuario, endpoint, clave API, token)
* Documentos PDF en carpeta /informes/ o generados desde escaneos

**🪜 PASO A PASO – CREACIÓN Y PRUEBAS DEL FRONTEND**

**🔧 1. Instalar dependencias para frontend**

Desde la carpeta descomprimida:

cd /mnt/data/platsentinel\_ui\_firma\_digital

npm install

Instalar soporte PDF (si es necesario):

npm install pdf-lib axios

**🧩 2. Crear componente de firma digital**

📁 src/componentes/FirmaDigital.vue

<template>

<div>

<h2>🖊️ Firmar Documento</h2>

<select v-model="documento">

<option v-for="doc in documentos" :key="doc" :value="doc">{{ doc }}</option>

</select>

<button @click="firmar">Firmar con SealSign</button>

<p v-if="mensaje">{{ mensaje }}</p>

</div>

</template>

<script>

export default {

data() {

return {

documentos: [],

documento: "",

mensaje: ""

};

},

mounted() {

// Simula cargar archivos disponibles

this.documentos = ["informe\_cliente\_001.pdf", "evaluacion\_riesgos\_2025.pdf"];

},

methods: {

async firmar() {

try {

const respuesta = await fetch("http://localhost:8000/firma/solicitar", {

method: "POST",

headers: {

"Content-Type": "application/json",

"Authorization": `Bearer ${localStorage.getItem("token")}`

},

body: JSON.stringify({ archivo: this.documento })

});

const data = await respuesta.json();

this.mensaje = data.mensaje || "Solicitud de firma enviada.";

} catch (e) {

this.mensaje = "❌ Error al firmar el documento.";

}

}

}

};

</script>

**📲 3. Crear endpoint backend simulado: firma/solicitar**

(Esto va en api/app/rutas/firma.py, necesita el SDK de SealSign o integración vía REST):

from fastapi import APIRouter, Depends

from app.dependencias import obtener\_usuario\_actual

import requests

router = APIRouter(prefix="/firma", tags=["Firma Digital"])

@router.post("/solicitar")

def firmar\_documento(payload: dict, usuario=Depends(obtener\_usuario\_actual)):

archivo = payload.get("archivo")

# Lógica real de envío al servidor de SealSign (simulada aquí)

respuesta = requests.post(

"https://api.sealsign.com/firma",

headers={"Authorization": "Bearer TU\_API\_TOKEN"},

json={

"usuario": usuario["sub"],

"archivo": archivo,

"callback": f"http://tuservidor.com/firma/callback"

}

)

if respuesta.status\_code == 200:

return {"mensaje": "✅ Firma solicitada correctamente"}

else:

return {"mensaje": "❌ Error al solicitar firma"}

**🧪 Escenario de prueba preparado**

| **Escenario** | **Objetivo** | **Resultado esperado** |
| --- | --- | --- |
| Usuario firma informe PDF | Validar autenticación y envío correcto a SealSign | Firma electrónica válida aplicada |
| Usuario sin sesión intenta firmar | Validación de token JWT | Error 401: no autorizado |
| Documento inexistente | Manejo de error | Error controlado: archivo no válido |
| Firma múltiple (futuro) | Firmas encadenadas por varios perfiles | A implementar: firmas en cadena |

**¿Qué sigue?**

¿Quieres que empaquete esta versión extendida con firma en un .zip listo para subir, o generamos el backend con el endpoint /firma/solicitar funcional también?