¡Perfecto, Chema! Vamos con **ambas opciones** integradas profesionalmente:

**✅ OPCIÓN 1: Guardar resultados del escaneo web en la API (Base de Datos)**

**📁 api/app/rutas/resultados.py**

from fastapi import APIRouter, Depends

from app.dependencias import obtener\_usuario\_actual

router = APIRouter(prefix="/resultados", tags=["Resultados"])

# Simulación de almacenamiento en memoria (puedes conectarlo a DB real)

RESULTADOS\_ESCANEO = []

@router.post("/guardar")

def guardar\_resultado(resultados: dict, usuario=Depends(obtener\_usuario\_actual)):

entrada = {

"usuario": usuario["sub"],

"resultados": resultados

}

RESULTADOS\_ESCANEO.append(entrada)

return {"mensaje": "✅ Resultados almacenados correctamente."}

@router.get("/mis-resultados")

def obtener\_mis\_resultados(usuario=Depends(obtener\_usuario\_actual)):

return [r for r in RESULTADOS\_ESCANEO if r["usuario"] == usuario["sub"]]

**🔧 Integrar desde escaneo\_web.py (al final):**

# Nuevo: enviar los resultados a la API

def reportar\_a\_api(resultados, url):

try:

respuesta = requests.post(

"http://localhost:8000/resultados/guardar",

json={"url": url, "resultados": resultados},

headers={"Authorization": f"Bearer {os.getenv('TOKEN\_JWT')}"}

)

print("📤 Resultados reportados a API:", respuesta.status\_code)

except Exception as e:

print("❌ Error al reportar resultados:", e)

# Al final de lanzar\_escaneo()

reportar\_a\_api(alertas, url)

**✅ OPCIÓN 2: Generar un informe PDF/HTML del escaneo**

**📁 informes/generador\_informes.py**

from jinja2 import Environment, FileSystemLoader

from datetime import datetime

import pdfkit

import os

env = Environment(loader=FileSystemLoader('.'))

template = env.get\_template("plantilla\_reporte.html")

def generar\_informe(resultados: list, objetivo: str, usuario: str = "desconocido"):

timestamp = datetime.now().strftime("%Y-%m-%d\_%H-%M-%S")

filename = f"informe\_{usuario}\_{timestamp}.pdf"

html\_render = template.render(

usuario=usuario,

objetivo=objetivo,

fecha=datetime.now().strftime("%d/%m/%Y %H:%M"),

alertas=resultados

)

ruta\_salida = os.path.join("/informes", filename)

pdfkit.from\_string(html\_render, ruta\_salida)

return ruta\_salida

**📁 informes/plantilla\_reporte.html**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Informe de Escaneo</title>

<style>

body { font-family: Arial, sans-serif; }

h1 { color: #2c3e50; }

table { width: 100%; border-collapse: collapse; }

td, th { border: 1px solid #ccc; padding: 8px; text-align: left; }

</style>

</head>

<body>

<h1>Informe de Escaneo</h1>

<p><strong>Usuario:</strong> {{ usuario }}</p>

<p><strong>Objetivo:</strong> {{ objetivo }}</p>

<p><strong>Fecha:</strong> {{ fecha }}</p>

<h2>Alertas Detectadas</h2>

<table>

<tr>

<th>Riesgo</th>

<th>Alerta</th>

<th>URL</th>

</tr>

{% for alerta in alertas %}

<tr>

<td>{{ alerta.risk }}</td>

<td>{{ alerta.alert }}</td>

<td>{{ alerta.url }}</td>

</tr>

{% endfor %}

</table>

</body>

</html>

**🛠 Requiere instalar:**

apt-get install wkhtmltopdf

pip install pdfkit jinja2

**🔁 Flujo completo recomendado:**

1. toquen\_web/ hace escaneo.
2. Llama a /resultados/guardar.
3. Luego, generador\_informes.py genera el PDF a demanda.
4. Cliente lo descarga desde dashboard o recibe por WhatsApp.

¿Quieres que ahora integre el botón de descarga de informe en el dashboard Vue? ¿O que prepare una función para enviar directamente el PDF por WhatsApp como archivo?