🚀 Aquí tienes el archivo .zip con la **estructura completa del proyecto PlatSentinel**:

📦 [Descargar platsentinel-estructura.zip](sandbox://mnt/data/platsentinel-estructura.zip)

Incluye carpetas, archivos clave, Dockerfiles, estructura para tokens, backend, frontend, integraciones, documentación y más. Lista para empezar a trabajar en serio.

¿Quieres que ahora rellene algunos de los archivos clave (README.md, api\_whatsapp.py, Dockerfile, etc.) con contenido funcional?

Perfecto, Chema. Vamos a ir completando **los archivos clave** del proyecto con contenido funcional, útil para desarrollo real y presentación profesional.

**✅ 1. README.md**

# PlatSentinel

\*\*Plataforma modular de ciberseguridad como servicio\*\*, orientada a entornos de pequeñas y medianas empresas y equipos técnicos distribuidos. Basada en contenedores Docker, arquitectura de toquens y gestión remota vía dashboard o WhatsApp.

## Características clave

- 🔐 Tokens independientes dockerizados

- 📊 Dashboard centralizado (Vue + FastAPI)

- 📱 Interacción vía WhatsApp API (Twilio)

- 🌐 Integraciones con servicios cloud y externos

- 📂 Módulo educativo con OWASP/MITRE

## Estructura del proyecto

```bash

api/ # Backend FastAPI

tokens/ # Toquens especializados (PDF, web, cloud...)

ui/ # Frontend Vue 3

integracion/ # APIs externas (WhatsApp, etc)

bot-whatsapp/ # Bot funcional vía Twilio

documentacion/ # Manuales y arquitectura

**Cómo empezar**

cp .env.ejemplo .env

docker-compose up --build

Más detalles en documentacion/guia\_despliegue.md

---

### ✅ 2. `integracion/api\_whatsapp.py`

```python

import os

from twilio.rest import Client

from flask import Flask, request

app = Flask(\_\_name\_\_)

ACCOUNT\_SID = os.getenv("CLAVE\_TWILIO\_SID")

AUTH\_TOKEN = os.getenv("CLAVE\_TWILIO\_TOKEN")

FROM\_WA = os.getenv("NUMERO\_WHATSAPP\_REMISOR")

client = Client(ACCOUNT\_SID, AUTH\_TOKEN)

@app.route("/webhook", methods=["POST"])

def responder():

mensaje = request.form.get('Body')

numero = request.form.get('From')

if "status" in mensaje.lower():

respuesta = "Sistema activo. Todos los toquens están operativos."

elif "ayuda" in mensaje.lower():

respuesta = "Comandos disponibles: status, escanear, detener, informe"

else:

respuesta = "Comando no reconocido. Escribe 'ayuda'."

client.messages.create(

body=respuesta,

from\_=FROM\_WA,

to=numero

)

return "OK", 200

**✅ 3. tokens/toquen\_antesala/Dockerfile**

FROM python:3.10-slim

WORKDIR /app

COPY . /app

RUN pip install --no-cache-dir watchdog requests

CMD ["python", "escaneo.py"]

**✅ 4. tokens/toquen\_antesala/escaneo.py**

import os

import time

from watchdog.observers import Observer

from watchdog.events import FileSystemEventHandler

class AnalizadorArchivos(FileSystemEventHandler):

def on\_created(self, event):

if not event.is\_directory:

print(f"Nuevo archivo detectado: {event.src\_path}")

# Aquí llamarías a análisis por tipo o derivación al token adecuado

if \_\_name\_\_ == "\_\_main\_\_":

ruta = "/entrada"

os.makedirs(ruta, exist\_ok=True)

observer = Observer()

observer.schedule(AnalizadorArchivos(), path=ruta, recursive=False)

observer.start()

print(f"Monitorizando {ruta}...")

try:

while True:

time.sleep(5)

except KeyboardInterrupt:

observer.stop()

observer.join()

¿Quieres que prepare también el contenido inicial para guia\_despliegue.md, documentacion\_tecnica.md o el Dockerfile base para la API? Puedo seguir llenando la estructura con ejemplos y contenido útil.

Perfecto, Chema. Vamos a seguir completando más archivos clave de forma profesional y funcional. Aquí tienes:

**✅ 5. api/Dockerfile**

FROM python:3.10-slim

ENV PYTHONDONTWRITEBYTECODE 1

ENV PYTHONUNBUFFERED 1

WORKDIR /api

COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

COPY . .

CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000", "--reload"]

**✅ 6. api/app/configuracion.py**

from pydantic import BaseSettings

class Settings(BaseSettings):

bd\_host: str = "localhost"

bd\_puerto: int = 5432

bd\_usuario: str = "sentinel\_usr"

bd\_clave: str = "super\_secreto"

clave\_jwt: str = "clave\_ultra\_secreta"

algoritmo\_jwt: str = "HS256"

duracion\_token: int = 60

class Config:

env\_file = ".env"

settings = Settings()

**✅ 7. documentacion/guia\_despliegue.md**

# Guía de Despliegue - PlatSentinel

Esta guía proporciona los pasos necesarios para levantar todo el entorno de desarrollo o producción de PlatSentinel utilizando `docker-compose`.

## Requisitos previos

- Docker + Docker Compose

- Make (opcional)

- Python 3.10 para tareas locales

## Despliegue rápido

```bash

cp .env.ejemplo .env

docker-compose up --build

**Servicios incluidos**

* API FastAPI (http://localhost:8000)
* Frontend Vue (http://localhost:3000)
* Bot WhatsApp Twilio Webhook (/webhook)
* Tokens dockerizados (PDF, ClamAV, antesala...)

**Variables de entorno**

Consulta .env.ejemplo para saber qué configurar según tu proveedor cloud o cuenta Twilio.

---

### ✅ 8. `.env.ejemplo`

```dotenv

# Base de datos

BD\_HOST=postgres

BD\_PUERTO=5432

BD\_USUARIO=sentinel\_usr

BD\_CLAVE=super\_secreto

# Claves JWT

CLAVE\_JWT=clave\_ultra\_secreta

ALGORITMO\_JWT=HS256

EXPIRACION\_TOKEN=60

# WhatsApp / Twilio

CLAVE\_TWILIO\_SID=ACxxxxxx

CLAVE\_TWILIO\_TOKEN=xxxxxxxxxx

NUMERO\_WHATSAPP\_REMISOR=whatsapp:+1415xxxxxxx

# VirusTotal API

CLAVE\_VIRUSTOTAL=VT\_xxxxxxxx

**¿Qué sigue?**

Puedo ayudarte ahora con:

1. api\_whatsapp.py mejorado con logs y validaciones.
2. api/rutas/autenticacion.py con JWT funcional.
3. ui/src/router/index.js con rutas reales y simuladas.
4. tokens/toquen\_web/escaneo\_web.py usando ZAP.
5. generador\_informes.py con plantilla básica PDF o HTML.

¿Con cuál quieres que continúe?