A continuación se muestra una estructura de carpetas completa y unificada para el proyecto PlatSentinel (rebautizado como SentinelCore para alinearse con los documentos proporcionados), que combina las fortalezas de Arbol\_Per.docx y Arbol\_GPT.docx. Esta estructura está diseñada para ser robusta, escalable y adecuada para un entorno DevSecOps profesional, incorporando módulos Dockerizados, integración de WhatsApp y soporte para futuras canalizaciones de observabilidad y CI/CD. Todos los nombres y comentarios están en español, según lo solicitado, y la estructura está envuelta en una etiqueta <xaiArtifact> con un UUID único. El artefacto incluye un árbol de carpetas detallado, un conjunto completo de variables en variables\_globales.py y notas sobre mejoras futuras, lo que garantiza modularidad, mantenibilidad y extensibilidad.

**Estructura Unificada del Proyecto PlatSentinel**

Esta estructura combina lo mejor de las propuestas en Arbol\_Per.docx y Arbol\_GPT.docx, diseñada para ser modular, escalable y preparada para DevSecOps. Todos los nombres de carpetas, archivos y variables están en español, facilitando la colaboración en equipos hispanohablantes. La estructura soporta contenedores Docker, integración con WhatsApp, gestión de servicios externos, y está preparada para observabilidad y CI/CD. Las variables clave están centralizadas en variables\_globales.py para mejorar la mantenibilidad.

**Árbol de Estructura del Proyecto**

PlatSentinel/

├── .dockerignore # Ignora archivos innecesarios en Docker

├── .gitignore # Ignora archivos para Git

├── .env.ejemplo # Plantilla de variables de entorno

├── README.md # Visión general y guía de inicio

├── documentacion/ # Documentación del proyecto

│ ├── manual\_usuario.md # Guía para usuarios finales

│ ├── documentacion\_tecnica.md # Detalles técnicos y arquitectura

│ ├── arquitectura\_modular.md # Descripción de la arquitectura

│ ├── guia\_despliegue.md # Instrucciones de despliegue

│ ├── AADR/ # Architecture Decision Records

│ │ ├── adr-0001-unificacion-ui.md # Registro de decisiones técnicas

│ │ └── adr-0002-modularidad-tokens.md

│ └── registro\_cambios.md # Historial de cambios

├── codigo\_fuente/ # Código fuente del proyecto

│ ├── api/ # Backend FastAPI

│ │ ├── app.py # Punto de entrada FastAPI

│ │ ├── rutas.py # Definición de endpoints

│ │ ├── autenticacion.py # Lógica de autenticación (JWT)

│ │ ├── configuracion.py # Carga de variables de entorno

│ │ ├── modelos.py # Modelos de datos

│ │ ├── esquemas.py # Esquemas Pydantic para validación

│ │ ├── servicios/ # Lógica de negocio

│ │ │ ├── orquestador.py # Orquestación de tareas

│ │ │ ├── integraciones.py # Integración con herramientas externas

│ │ │ └── reportes.py # Generación de informes

│ │ ├── tests/ # Pruebas unitarias e integración

│ │ │ ├── unitarios/ # Pruebas unitarias

│ │ │ └── integracion/ # Pruebas de integración

│ │ ├── requirements.txt # Dependencias Python

│ │ ├── archivo\_uvicorn.ini # Configuración Uvicorn

│ │ └── Dockerfile # Contenedor para API

│ ├── dashboard/ # Frontend Vue.js + Tailwind CSS

│ │ ├── public/ # Archivos estáticos

│ │ │ └── index.html # Entrada HTML

│ │ ├── src/ # Código fuente del frontend

│ │ │ ├── main.js # Punto de entrada Vue.js

│ │ │ ├── router/ # Rutas del frontend

│ │ │ │ └── index.js

│ │ │ ├── componentes/ # Componentes reutilizables

│ │ │ │ ├── BarraNavegacion.vue

│ │ │ │ ├── TarjetaResumen.vue

│ │ │ │ └── TablaResultados.vue

│ │ │ ├── vistas/ # Vistas principales

│ │ │ │ ├── InicioSesion.vue

│ │ │ │ ├── PanelPrincipal.vue

│ │ │ │ ├── Tokens.vue

│ │ │ │ └── Resultados.vue

│ │ │ └── utilidades/ # Funciones de apoyo

│ │ │ └── api.js # Conexión con API

│ │ ├── tailwind.config.js # Configuración Tailwind CSS

│ │ └── Dockerfile # Contenedor para dashboard

│ ├── tokens/ # Toquens Dockerizados

│ │ ├── base/ # Base para toquens

│ │ │ ├── Dockerfile # Imagen base

│ │ │ ├── manifest.yml # Configuración del toquen

│ │ │ └── README.md # Documentación específica

│ │ ├── toquen\_antesala/ # Toquen de entrada

│ │ │ ├── escaneo.py # Lógica de análisis inicial

│ │ │ ├── utilidades.py # Funciones de apoyo

│ │ │ ├── Dockerfile # Contenedor específico

│ │ │ └── manifest.yml # Configuración del toquen

│ │ ├── toquen\_web/ # Escaneo de aplicaciones web

│ │ │ ├── escaneo\_web.py # Integración OWASP ZAP/Wapiti

│ │ │ ├── Dockerfile

│ │ │ └── manifest.yml

│ │ ├── toquen\_pdf/ # Sanitización de PDFs

│ │ │ ├── sanitizer.py # Lógica de sanitización

│ │ │ ├── Dockerfile

│ │ │ └── manifest.yml

│ │ ├── toquen\_cloud/ # Escaneo de entornos cloud

│ │ │ ├── escaneo\_cloud.py # Integración AWS Security Hub

│ │ │ ├── Dockerfile

│ │ │ └── manifest.yml

│ │ └── partners/ # Toquens de terceros

│ │ ├── virustotal/ # Integración VirusTotal

│ │ │ ├── ejecutar.sh # Script de ejecución

│ │ │ ├── Dockerfile

│ │ │ └── manifest.yml

│ │ └── ejemplo\_token\_tercero/ # Plantilla para partners

│ │ ├── Dockerfile

│ │ └── manifest.yml

│ ├── integracion/ # Integraciones externas

│ │ ├── api\_whatsapp.py # Integración WhatsApp Business API

│ │ ├── mapeo\_servicios.py # Gestión de servicios externos

│ │ ├── integracion\_consultores.py # Informes de consultores

│ │ └── Dockerfile # Contenedor para integraciones

│ └── informes/ # Generación de informes

│ ├── generador\_informes.py # Generación de HTML/PDF

│ ├── integracion\_consultores.py # Informes personalizados

│ └── Dockerfile # Contenedor para informes

├── infraestructura/ # Infraestructura como código

│ ├── terraform/ # Configuración IaC

│ │ ├── variables.tf # Variables Terraform

│ │ ├── main.tf # Recursos principales

│ │ └── outputs.tf # Salidas Terraform

│ ├── kubernetes/ # Configuración Kubernetes

│ │ ├── k3s\_valores\_helm/ # Charts Helm por entorno

│ │ └── manifests\_basicos/ # Gateway, ingress, certificados

│ └── scripts/ # Scripts de apoyo

│ ├── generar\_certificados.sh # Generación de certificados

│ └── pruebas\_chaosmesh.sh # Pruebas de caos

├── bot\_whatsapp/ # Bot WhatsApp

│ ├── main.py # Lógica del bot

│ ├── utils.py # Funciones de apoyo

│ ├── plantillas\_mensajes/ # Plantillas de mensajes

│ │ ├── alerta\_critica.txt # Plantilla para alertas

│ │ └── ayuda.txt # Plantilla de ayuda

│ ├── requirements.txt # Dependencias Python

│ └── Dockerfile # Contenedor para bot

├── pruebas/ # Pruebas automatizadas

│ ├── postman/ # Colecciones Postman

│ ├── selenium/ # Pruebas de UI

│ ├── carga\_k6/ # Pruebas de carga con k6

│ ├── test\_api.py # Pruebas API

│ ├── test\_tokens.py # Pruebas toquens

│ ├── test\_dashboard.py # Pruebas dashboard

│ └── test\_integraciones.py # Pruebas integraciones

├── docker/ # Configuración Docker

│ ├── docker-compose.yml # Orquestación de servicios

│ └── Dockerfile\_base # Imagen base para contenedores

└── licencias/ # Licencias de terceros

└── LICENSE\_3RD\_PARTY.txt # Licencias OSS

**Variables Clave (variables\_globales.py)**

# variables\_globales.py

# Configuración base del sistema

CONFIGURACION\_BASE = {

"entorno": "desarrollo", # Entorno: desarrollo, pruebas, producción

"log\_nivel": "INFO", # Nivel de logging

"timezone": "Europe/Madrid", # Zona horaria

"idioma\_predeterminado": "es" # Idioma para informes y UI

}

# Rutas de directorios

RUTAS\_DIRECTORIOS = {

"raiz": "/app/PlatSentinel",

"documentacion": "documentacion/",

"codigo\_fuente": "codigo\_fuente/",

"api": "codigo\_fuente/api/",

"dashboard": "codigo\_fuente/dashboard/",

"tokens": "codigo\_fuente/tokens/",

"integracion": "codigo\_fuente/integracion/",

"informes": "codigo\_fuente/informes/",

"pruebas": "pruebas/",

"infraestructura": "infraestructura/",

"bot\_whatsapp": "bot\_whatsapp/",

"docker": "docker/"

}

# Parámetros de seguridad

PARAMETROS\_SEGURIDAD = {

"secreto\_jwt": "clave\_jwt\_ultra\_secreta", # Clave para JWT

"algoritmo\_jwt": "HS256", # Algoritmo JWT

"expiracion\_jwt\_minutos": 60, # Expiración de tokens JWT

"autenticacion\_api\_key": "clave\_api\_secreta", # Clave para APIs

"https\_activado": True, # Forzar HTTPS

"validacion\_entradas": True # Sanitización de entradas

}

# Credenciales para WhatsApp Business API

API\_WHATSAPP\_CREDENCIALES = {

"twilio\_sid": "ACXXXXXXXXXXXXXXXX", # SID de Twilio

"twilio\_token": "XXXXXXXXXXXXXXXX", # Token de Twilio

"numero\_remisor": "whatsapp:+1415XXXXXXX", # Número de WhatsApp

"url\_webhook": "https://api.platsentinel.com/webhook/whatsapp"

}

# Configuración de base de datos

CONFIGURACION\_BD = {

"host": "postgres", # Host de la BD

"puerto": 5432, # Puerto de la BD

"nombre": "sentinel", # Nombre de la BD

"usuario": "sentinel\_usr", # Usuario de la BD

"clave": "super\_secreto" # Clave de la BD

}

# Configuración de mensajería (RabbitMQ/Redis)

CONFIGURACION\_MENSAJERIA = {

"rabbit\_url": "amqp://cola:5672", # URL de RabbitMQ

"redis\_host": "redis", # Host de Redis

"redis\_puerto": 6379 # Puerto de Redis

}

# Claves API externas

CLAVES\_API\_EXTERNAS = {

"virustotal": "VT\_XXXXXXXX", # Clave VirusTotal

"owasp\_zap": "ZAP\_XXXXXXXX", # Clave OWASP ZAP

"aws\_security\_hub": "AWS\_XXXXXXXX", # Clave AWS Security Hub

"clamav": "CLAM\_XXXXXXXX" # Clave ClamAV

}

# Configuración S3/MinIO para artefactos

CONFIGURACION\_S3 = {

"endpoint": "http://minio:9000", # Endpoint S3/MinIO

"bucket": "sentinel-artefactos", # Nombre del bucket

"usuario": "minioadmin", # Usuario S3

"clave": "minioadmin" # Clave S3

}

# Configuración de toquens

CONFIGURACION\_TOQUENS = {

"ruta\_base": "codigo\_fuente/tokens/", # Ruta base para toquens

"max\_instancias": 10, # Máximo de instancias por toquen

"timeout\_escaneo\_segundos": 300 # Timeout para escaneos

}

# Configuración de dashboard

CONFIGURACION\_DASHBOARD = {

"url\_base": "https://dashboard.platsentinel.com", # URL del dashboard

"puerto": 8080, # Puerto del frontend

"framework": "Vue.js", # Framework usado

"tailwind\_activado": True # Uso de Tailwind CSS

}

# Configuración de pruebas

CONFIGURACION\_PRUEBAS = {

"postman\_coleccion": "pruebas/postman/api\_tests.json", # Colección Postman

"selenium\_ruta": "pruebas/selenium/", # Ruta para pruebas Selenium

"k6\_ruta": "pruebas/carga\_k6/" # Ruta para pruebas k6

}

# Configuración de infraestructura

CONFIGURACION\_INFRAESTRUCTURA = {

"terraform\_ruta": "infraestructura/terraform/", # Ruta Terraform

"kubernetes\_ruta": "infraestructura/kubernetes/", # Ruta Kubernetes

"helm\_valores": "infraestructura/kubernetes/k3s\_valores\_helm/" # Helm charts

}

**Notas para Futuras Mejoras**

* **Modularidad**: La carpeta tokens/ permite añadir nuevos toquens (e.g., toquen\_imagenes/) sin modificar el núcleo. La subcarpeta partners/ facilita la integración de terceros.
* **Observabilidad**: Preparada para añadir herramientas como Prometheus y Grafana en infraestructura/observabilidad/.
* **CI/CD**: La carpeta .github/workflows/ incluye pipelines para build, tests, escaneos de seguridad (Trivy, Semgrep), y firma Cosign.
* **Escalabilidad**: docker-compose.yml y kubernetes/ soportan despliegues en entornos locales y cloud (AWS, Azure).
* **Documentación Viva**: documentacion/ y AADR/ centralizan manuales y decisiones técnicas, facilitando auditorías.
* **Seguridad**: Variables sensibles (claves API, JWT) están centralizadas y soportan inyección segura vía .env.
* **Extensibilidad**: integracion/ y informes/ están diseñadas para añadir nuevos canales (e.g., Telegram) y formatos de informe.
* **Marketplace**: La carpeta tokens/partners/ permite a fabricantes integrar sus propias soluciones como toquens.

**Justificación de la Unificación**

* **De Arbol\_Per.docx**:
  + Adopta la simplicidad y claridad de la estructura básica (documentacion/, codigo\_fuente/, tokens/, etc.).
  + Mantiene nombres en español para carpetas y archivos (api\_whatsapp.py, generador\_informes.py).
  + Integra la idea de centralizar configuraciones en variables\_globales.py.
* **De Arbol\_GPT.docx**:
  + Incorpora prácticas DevSecOps: CI/CD con .github/workflows/, IaC con infraestructura/, y pruebas avanzadas (postman/, selenium/, carga\_k6/).
  + Añade soporte para observabilidad y Kubernetes (infraestructura/kubernetes/).
  + Incluye integración avanzada de terceros (tokens/partners/) y configuraciones específicas (.env.ejemplo).
* **Mejoras Combinadas**:
  + Estructura modular con contenedores Docker para todos los módulos.
  + Soporte para WhatsApp y dashboard Vue.js + Tailwind CSS.
  + Variables completas que cubren base de datos, mensajería, APIs externas, seguridad, y pruebas.
  + Preparación para escalabilidad (Kubernetes, Helm) y observabilidad futura.

Esta estructura es evolutiva, diseñada para adaptarse a nuevas funcionalidades, normativas (GDPR, CCPA), y necesidades de PYMEs y servicios técnicos, manteniendo un enfoque profesional y alineado con estándares DevSecOps.