**Propuesta estructurada y realista**, adaptada a:

* **4 personas**
* **80 horas/persona**
* **Trabajo modular, colaborativo**
* **Toquens como innovación**
* **Enfoque MVP (producto mínimo viable)**

**🛠️ Proyecto: SentinelCore (Nombre tentativo)**

**Objetivo**  
Desarrollar una aplicación modular que centralice herramientas de análisis de vulnerabilidades en sitios web y entornos cloud, ofreciendo información clara sobre cómo mitigar riesgos, e incorporando un sistema de *toquens de antesala* que actúa antes de escribir archivos en disco para prevenir amenazas.

**🔍 Enfoque MVP (80h x 4 personas = 320h totales)**

**🔑 Funcionalidades esenciales para el prototipo**

| **Función** | **Descripción** | **Justificación** |
| --- | --- | --- |
| **Escaneo web básico** | Integrar OWASP ZAP o Wapiti para escaneo de sitios web | Fundamental para análisis inicial |
| **Sistema “toquen de antesala” básico** | Prototipo funcional que analiza un archivo antes de almacenarlo, lo clasifica (ej. PDF, script, ejecutable) y lo envía a un *toquen especializado* (simulado o real) | Núcleo innovador |
| **Consola o interfaz web simple** | Para ejecutar escaneos, ver resultados y logs de toquens | Interacción básica con el sistema |
| **Base de datos de resultados** | Guardar escaneos, acciones de toquens, y clasificación | Necesario para trazabilidad |
| **Sugerencias de mitigación** | Mostrar texto explicativo básico por tipo de vulnerabilidad | Aporta valor práctico y educativo |

**🧩 Estructura Modular del Proyecto**

SentinelCore/

│

├── core/ # Núcleo de gestión y orquestación

│ ├── main.py # Entry point

│ ├── dispatcher.py # Encaminador de archivos a toquens

│ └── config/

│

├── toquens/ # Directorio de toquens individuales

│ ├── antesala.py # Interceptor antes del disco

│ ├── static\_scan.py # Ejemplo: YARA o ClamAV

│ └── pdf\_sanitizer.py # Para PDF, DOCX, etc.

│

├── webscanner/ # Módulo para escaneo web

│ ├── zap\_wrapper.py

│ └── results\_parser.py

│

├── gui/ # Interfaz (CLI o web)

│ └── simple\_web\_gui.py

│

├── database/ # SQLite o similar

│ └── models.py

│

└── docs/ # Documentación

**👥 Reparto del trabajo por roles (equipo de 4 personas)**

| **Rol** | **Tareas principales** |
| --- | --- |
| **Dev A – WebScanner & Mitigación** | Integrar OWASP ZAP o Wapiti, parsear resultados, añadir base de datos simple y recomendaciones OWASP |
| **Dev B – Toquen de antesala & Clasificador** | Desarrollar el sistema que analiza archivos antes del disco, identifica tipo y reenvía |
| **Dev C – Toquens especializados** | Desarrollar 2 toquens (ej. ClamAV o YARA + sanitizador de PDF/script), simulando acciones si hace falta |
| **Dev D – Interfaz y orquestador** | Crear consola o interfaz web básica, menú de opciones, monitor de toquens y resultados |

⚙️ Todos colaboran en documentación, testing y revisión final.

**🧠 Valor añadido frente al mercado**

| **Competencia (Nessus, OpenVAS, etc.)** | **SentinelCore MVP** |
| --- | --- |
| Detectan vulnerabilidades | ✅ También |
| Mitigación automática real | ❌ |
| Pre-almacenamiento de archivos | ❌ |
| Modularidad / Plugins | Limitado |
| Ideal para formación/laboratorio | ⚠️ |

**📅 Planificación Tentativa (por semanas)**

| **Semana** | **Actividades** |
| --- | --- |
| 1 | Diseño técnico, definición de módulos, configuración de repositorio |
| 2 | Desarrollo de escáner web, primer toquen de antesala y mock de toquen especializado |
| 3 | Desarrollo de interfaz, conexión entre módulos, integración base de datos |
| 4 | Testing, documentación, revisión, presentación y demo final |

**✅ Resultado Esperado del MVP**

* ✅ Escaneo básico de una web y muestra de vulnerabilidades
* ✅ Sistema activo de interceptación de archivos antes de su escritura
* ✅ Clasificación básica de archivos y enrutamiento a toquens
* ✅ Dos toquens funcionales (simulados o reales)
* ✅ Panel (CLI o web) para interactuar con el sistema
* ✅ Informe básico con vulnerabilidades encontradas y sugerencias

**README técnico inicial** para arrancar el proyecto con claridad. Este archivo está listo para colocarse en la raíz del repositorio Git (/SentinelCore/README.md):

# 🛡️ SentinelCore

\*\*Plataforma modular para el análisis de vulnerabilidades web y cloud, con sistema de toquens de defensa activa.\*\*

---

## 🎯 Objetivo

SentinelCore es una aplicación de seguridad informática diseñada para:

- Centralizar herramientas de análisis de vulnerabilidades web/cloud.

- Mostrar información clara sobre riesgos y cómo mitigarlos.

- Interceptar archivos antes de su almacenamiento mediante un sistema innovador de \*\*toquens de antesala\*\*.

---

## 🚀 Funcionalidades MVP

✔️ Escaneo de páginas web con herramientas como Wapiti u OWASP ZAP

✔️ Análisis de archivos antes de su almacenamiento en disco

✔️ Clasificación y redirección de archivos a toquens especializados

✔️ Sugerencias básicas de mitigación por vulnerabilidad

✔️ Interfaz de control simple (CLI o web)

---

## 🧬 Arquitectura Modular

```plaintext

SentinelCore/

│

├── core/ → Lógica de orquestación y configuración

├── toquens/ → Toquens individuales: antesala, análisis, sanitización

├── webscanner/ → Integración con escáneres de vulnerabilidades web

├── gui/ → Interfaz de usuario básica

├── database/ → Resultados, logs y sugerencias de mitigación

└── docs/ → Documentación del proyecto

**💡 Innovación: Toquen de Antesala**

El “toquen de antesala” es un sistema que:

1. Intercepta archivos antes de que sean almacenados físicamente.
2. Analiza su tipo (binario, PDF, script, etc.).
3. Los deriva a toquens especializados para escaneo, desinfección o rechazo.
4. Se comunica con otros toquens para reforzar la seguridad desde el inicio.

Cada *toquen* puede apoyarse en otros, formando una red modular y colaborativa de protección.

**🧩 Requisitos Técnicos Iniciales**

* Python 3.10+
* SQLite (u otra DB ligera)
* Wapiti / OWASP ZAP (API REST)
* ClamAV / YARA (opcional para toquens)
* Flask (si se usa GUI web)
* Docker (para empaquetado de toquens en el futuro)

**👥 Equipo de Desarrollo**

| **Nombre** | **Rol** | **Responsabilidades** |
| --- | --- | --- |
| Dev A | WebScanner | Integración escáner y mitigación |
| Dev B | Toquen de Antesala | Interceptor y clasificador |
| Dev C | Toquens Especializados | Módulos de análisis y sanitización |
| Dev D | GUI y Orquestación | Interfaz y lógica general |

**📅 Planificación (Sprint de 4 semanas)**

| **Semana** | **Actividades** |
| --- | --- |
| 1 | Diseño técnico, división de módulos, entorno común |
| 2 | Desarrollo de scanner web y toquen antesala |
| 3 | GUI simple, BD de resultados y conexión total |
| 4 | Testing, documentación, presentación final |

**📘 Documentación**

Toda la documentación estará en /docs/, incluyendo:

* Manual del usuario
* Guía para desarrollar nuevos toquens
* Detalles de arquitectura
* Informe final

**✅ Licencia**

Proyecto académico bajo licencia MIT para su uso libre y distribución educativa.

**📢 Nota final**

Este proyecto prioriza la funcionalidad esencial, la modularidad y la innovación. El objetivo no es competir directamente con gigantes del mercado, sino aportar un enfoque pedagógico, extensible y centrado en la acción **antes** del daño.

---

¿Quieres que lo convierta también en `.docx`, o prefieres que avancemos con el \*\*diagrama de arquitectura\*\* ahora?

**Diagrama de Arquitectura: SentinelCore (MVP)**

**1. Descripción General**

SentinelCore es una aplicación modular diseñada para centralizar herramientas de análisis de vulnerabilidades y gestionar archivos potencialmente peligrosos antes de que se almacenen en el sistema. Su innovación reside en el "toquen de antesala", que intercepta archivos y los redirige a módulos especializados para su análisis e inoculación.

**2. Diagrama Lógico de Componentes (Texto)**

+-----------------------------+

| Interfaz GUI / CLI |

+-------------+--------------+

|

v

+-------------+--------------+

| Orquestador Central |

| (main.py + dispatcher.py) |

+------+------+--------------+

| | |

v v v

+------+ +--------+ +-----------------+

| Base | | Web | | Toquen de |

| de | | Scanner| | Antesala |

| Datos | | (ZAP) | | (Intercepta |

| SQLite| | | | archivos) |

+------+ +--------+ +--------+--------+

|

v

+----------+-----------+

| Clasificador de Tipo |

+----------+-----------+

|

+----------------+----------------+

| |

v v

+----------+----------+ +-------------+-------------+

| Toquen: Static Scan | | Toquen: PDF/Scripts Scan |

| (ClamAV, YARA) | | Sanitizador / heurístico |

+---------------------+ +---------------------------+

**3. Flujo de Datos**

1. **Usuario lanza un escaneo web o carga un archivo** desde la interfaz (CLI o web).
2. **El orquestador** determina si es un escaneo o un archivo a procesar.
3. Para archivos:
   * El *toquen de antesala* intercepta el archivo antes de guardarlo.
   * Se clasifica el tipo de archivo (PDF, script, binario...).
   * Se encamina a un toquen especializado que realiza análisis e inoculación.
   * Se autoriza o rechaza su almacenamiento.
4. Para webs:
   * Se lanza Wapiti o ZAP.
   * Se almacenan resultados en SQLite.
   * Se generan sugerencias de mitigación según tipo de vulnerabilidad.
5. La interfaz muestra los resultados y logs de los toquens.

**4. Consideraciones Tecnológicas**

* **Lenguaje principal**: Python 3.x
* **Frameworks sugeridos**: Flask (para GUI), SQLite (base de datos), Subprocesos para integración con herramientas externas.
* **Toquens**: scripts modulares que se comunican mediante sockets, archivos temporales o colas simples.
* **Interfaz**: Web sencilla o CLI en consola para facilitar desarrollo en entorno educativo.

**5. Ventaja Competitiva**

* Modularidad basada en toquens
* Prevención antes de almacenamiento
* Integración con herramientas reales y libres
* Adaptado a entornos educativos, SOC pequeños o entornos offline

Este diagrama lógico sirve como base para desarrollar el MVP de SentinelCore en un entorno de trabajo distribuido y colaborativo.