Marco Antonio González Pérez

Taller de integración

Proyecto Centinela: Escaneo de vulnerabilidades.

**Enunciado General del Proyecto**

Una aplicación web que permita a los usuarios ingresar un dominio o IP pública y ejecutar escaneos automatizados para detectar posibles vulnerabilidades de seguridad, clasificarlas, y generar un informe visual y descargable.

**Parte 1: Análisis de Factibilidad (15 puntos)**

Completa la tabla:

|  |  |  |
| --- | --- | --- |
| **Tipo de factibilidad** | **¿Es factible? (Si, No, Parcial)** | **Justificación técnica (2-3 líneas)** |
| **Técnica** | **Sí** | **El proyecto es técnicamente viable dado que se apoya en tecnologías maduras y ampliamente soportadas como Django, MySQL, Celery, Redis y Tailwind, las cuales garantizan modularidad, escalabilidad y un ecosistema de librerías para los distintos escaneos de seguridad.** |
| **Operativa** | **Sí** | **El sistema es operativamente factible porque los usuarios solo requieren un navegador web para interactuar con la aplicación, y la interfaz gráfica permitirá acceder fácilmente a los resultados de los análisis sin necesidad de conocimientos técnicos avanzados.** |
| **De desarrollo** | **Sí** | **El desarrollo es viable considerando que el proyecto puede construirse en fases modulares (por ejemplo, cada tipo de escaneo como un módulo independiente), lo que facilita la gestión de recursos, la implementación progresiva de funcionalidades y la integración futura de mejoras como machine learning.** |

**Parte 2: Estimación del Esfuerzo con COCOMO (20 puntos)**

**Datos:**

 Tamaño estimado del sistema: **25 KLOC**

 Tipo de proyecto: **semi-detached**

**Fórmulas:**

 **Esfuerzo (persona-mes)** = 3.0 × (KLOC)^1.12

 **Duración (meses)** = 2.5 × (Esfuerzo)^0.35

**Preguntas:**

1. Esfuerzo estimado: \_\_\_110\_\_ persona-mes

2. Duración estimada: \_\_\_13\_\_ meses

3. ¿Cuántos integrantes del equipo estimas necesarios? Justifica.

Se requieren teóricamente 8 integrantes. El proyecto contempla múltiples áreas: backend (Django, lógica de escaneo), frontend (UI con Tailwind), DevOps (Docker, despliegue), pruebas de calidad, seguridad (módulos de escaneo), y machine learning (clasificación de riesgos). Cada rol especializado aporta al desarrollo ágil y robusto. No obstante, al ser un proyecto de título, el desarrollo será asumido por una sola persona, integrando y simplificando funciones.

**Parte 3: Arquitectura del Sistema (15 puntos)**

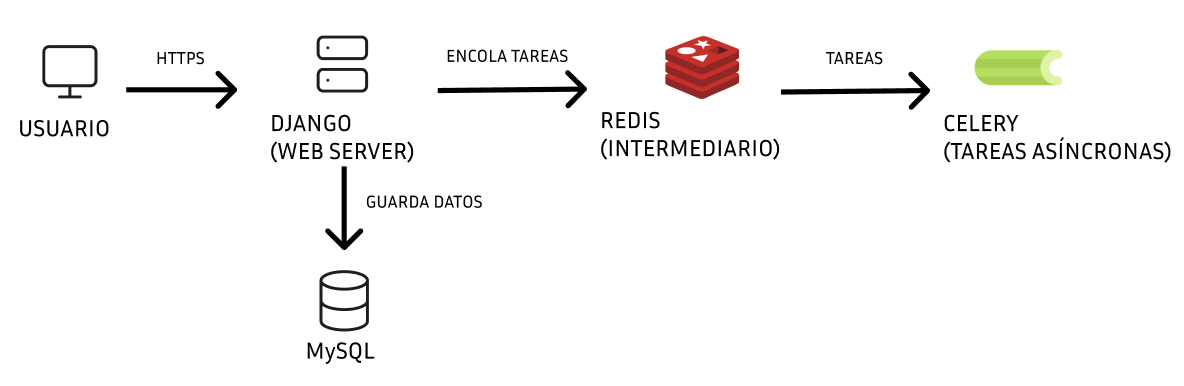
Diseña una arquitectura del sistema:

*Puedes usar dibujo, esquema o viñetas.*

El sistema desarrollado, denominado **Centinela**, se basa en una arquitectura **cliente–servidor** con separación clara entre las capas de presentación, lógica de negocio y persistencia.  
La solución implementa el patrón de diseño **MVC (Modelo–Vista–Controlador)**, siguiendo la estructura nativa del framework Django, lo que permite una adecuada separación de responsabilidades y facilita el mantenimiento y escalabilidad del proyecto.

Tecnologías seleccionadas y justificación

* **Django:** por su robustez, comunidad activa, patrón MTV(Model Template View), facilidad para crear aplicaciones modulares con autenticación integrada y ORM(Object-Relational Mapper).
* **Celery y Redis:** permiten ejecutar tareas en segundo plano, evitando bloqueos y mejorando la experiencia del usuario.
* **MySQL:** base de datos relacional confiable y ampliamente soportada.
* **Docker:** asegura portabilidad y replicabilidad del entorno.
* **Tailwind y Chart.js:** facilitan la creación de una interfaz moderna, responsiva, personalizable y visualmente atractiva.

**

**Parte 4: Metodología en Cascada (10 puntos)**

 ¿Por qué sería adecuada (o no) la metodología cascada para este proyecto?

La metodología cascada **NO sería la más adecuada** para este proyecto de scanner de vulnerabilidades, principalmente debido a la naturaleza evolutiva de la ciberseguridad y la necesidad de adaptación constante, sin embargo, al ser un proyecto en el contexto estudiantil, la metodología cascada funciona para obtener al menos el mínimo viable.

 Indica 2 ventajas y 2 desventajas en este contexto.

**Ventajas:**

1. **Documentación exhaustiva**: Permite crear documentación completa de cada módulo de escaneo y sus especificaciones técnicas, lo cual es crucial para auditorías de seguridad y cumplimiento normativo.
2. **Planificación estructurada**: Facilita la estimación precisa de recursos y tiempos para cada tipo de escaneo (puertos, SSL, vulnerabilidades web), permitiendo una mejor gestión del proyecto académico.

**Desventajas:**

1. **Rigidez ante cambios en amenazas**: Las vulnerabilidades de seguridad evolucionan constantemente, requiriendo actualizaciones frecuentes en los módulos de escaneo que no se adaptan bien al modelo secuencial y rígido de cascada.
2. **Feedback tardío del usuario**: Los usuarios no pueden probar funcionalidades hasta el final del desarrollo, impidiendo ajustes en la interfaz y usabilidad que son críticos en herramientas de seguridad.

**Parte 5: Análisis FODA (10 puntos)**

|  |  |
| --- | --- |
| **FORTALEZAS** | **OPORTUNIDADES** |
| • Tecnologías maduras y estables (Django, MySQL) con amplia documentación | • Creciente demanda de herramientas de ciberseguridad automatizadas |
| • Arquitectura modular que facilita escalabilidad y mantenimiento | • Posibilidad de integrar machine learning para mejorar detección |
| • Interfaz web accesible sin necesidad de instalación local | • Mercado en expansión para SMEs que necesitan auditorías de seguridad |
| • Ejecución asíncrona evita bloqueos y mejora UX | • Potencial para agregar compliance con estándares (ISO 27001, NIST) |

|  |  |
| --- | --- |
| **DEBILIDADES** | **AMENAZAS** |
| • Dependencia de herramientas externas (nmap, OpenVAS) para escaneos | • Competencia de soluciones comerciales consolidadas (Nessus, Qualys) |
| • Requiere conocimientos especializados para mantener módulos actualizados | • Posibles restricciones legales sobre escaneo de infraestructura externa |
| • Limitado a vulnerabilidades conocidas en bases de datos públicas | • Evolución rápida de amenazas que puede obsoletizar detecciones |
| • Proyecto desarrollado por una sola persona limita expertise diverso | • Riesgos de seguridad si la herramienta es mal utilizada |

**Parte 6: Alternativas Técnicas (20 puntos) (acá hay que poner la alternativa técnica que puse en la arquitectura de sistema con una justificación técnica breve y porque descarto otras alternativas)**

**Alternativa Seleccionada: Stack Django + Celery + Redis**

**Justificación Técnica:**

La arquitectura seleccionada se basa en **Django como framework web principal**, **Celery para procesamiento asíncrono** y **Redis como broker de mensajes**. Esta combinación ofrece:

* **Modularidad**: Django permite crear aplicaciones independientes para cada tipo de escaneo
* **Escalabilidad**: Celery distribuye tareas pesadas evitando timeouts HTTP
* **Confiabilidad**: Redis garantiza persistencia de tareas y manejo de colas
* **Ecosystem**: Amplia disponibilidad de librerías Python para ciberseguridad

**Alternativas Descartadas:**

**1. Node.js + Express + Bull/Bee-Queue**

* **Descartada porque**: Menor ecosistema de herramientas de seguridad en JavaScript comparado con Python. Las librerías para escaneo de vulnerabilidades están principalmente en Python/C.

**2. FastAPI + Celery + RabbitMQ**

* **Descartada porque**: Aunque FastAPI es más rápido, Django ofrece mayor madurez, ORM integrado y sistema de autenticación robusto. RabbitMQ añade complejidad innecesaria para este alcance.