Universidad Distrital Francisco José de Caldas
Facultad de Ingeniería

Herramienta de Análisis de IP con Fuentes Abiertas

Proyecto de Investigación

Investigadores:

Juan Manuel Serrano Rodríguez
Código: 20211020091
jumserranor@udistrital.edu.co

Nicolás Guevara Herrán
Código: 20211180026
nguevarah@udistrital.edu.co

Área de Investigación:
Ciberseguridad y Análisis de Redes

Grupo de Investigación:
Modelamiento en Ingeniería de Sistemas

Director:
Lilian Astrid Bejarano Garzón

25 de noviembre de 2025

1. Selección y Definición del Tema de Investigación

1.1. Tema de Investigación

Desarrollo de una herramienta de diagnóstico unificado de direcciones IP mediante integración de fuentes abiertas para análisis de seguridad en el contexto colombiano.

1.2. Título Provisional

Diagnóstico de Seguridad IP con Fuentes Abiertas en Colombia

1.3. Línea de Investigación

Este proyecto se enmarca dentro de la línea de investigación en **Inteligencia de Amenazas Cibernéticas** [10], específicamente en el área de **Análisis Automatizado de Infraestructuras de Red** y **Desarrollo de Herramientas de Ciberseguridad Basadas en Fuentes Abiertas** [8].

1.4. Área del Conocimiento

- **Área principal:** Ingeniería de Sistemas y Computación
- **Subárea:** Ciberseguridad y Redes de Computadores
- **Disciplina:** Inteligencia de Amenazas y Análisis de Vulnerabilidades

2. Planteamiento del Problema

2.1. Contexto del Problema

En el día a día de la ciberseguridad y el desarrollo de software, es común necesitar información detallada sobre una dirección IP. Un analista puede querer saber su ubicación geográfica, si está en una lista negra, qué puertos tiene abiertos o a qué organización pertenece.

Actualmente, esta información se encuentra dispersa en múltiples fuentes de datos, tanto gratuitas como de pago. Alghamdi [8] identifica esta fragmentación como una de las principales amenazas en el reconocimiento cibernético. Por ejemplo:

- **Geolocalización:** Se obtiene de bases de datos como GeoLite2 de MaxMind [13].

- **Información de Infraestructura:** Se obtiene mediante herramientas de escaneo de red como Nmap [14] o servicios como Shodan.
- **Vulnerabilidades:** Se consulta en bases de datos de vulnerabilidades como National Vulnerability Database (NVD) de NIST.

2.2. Definición del Problema Principal

El problema central es la **fragmentación de la información**. Un desarrollador o analista que necesita un perfil completo de una dirección IP debe realizar las siguientes acciones manualmente:

1. Consultar la API o base de datos de geolocalización.
2. Ejecutar herramientas de escaneo de red como Nmap para obtener datos de puertos y servicios.
3. Consultar APIs de vulnerabilidades como NVD para identificar CVEs asociados a los servicios detectados.
4. Consolidar y correlacionar toda esta información a mano para poder tomar una decisión.

Este proceso manual es **ineficiente, lento y propenso a errores**. Limita la capacidad de realizar análisis rápidos y automatizados, especialmente cuando se necesita evaluar un gran número de direcciones IP. Aunque existen herramientas comerciales que resuelven este problema, sus costos son a menudo prohibitivos para estudiantes, desarrolladores independientes o pequeños equipos. Esta situación evidencia la necesidad de desarrollar una solución automatizada que integre múltiples fuentes de datos en una única interfaz accesible.

2.3. Pregunta de Investigación

¿Cómo diseñar e implementar una herramienta de software que integre y unifique datos de fuentes abiertas y gratuitas (específicamente GeoLite2 para geolocalización, Nmap para escaneo de red, NVD para vulnerabilidades e ipapi.co para información de ASN/ISP) para presentar un diagnóstico de seguridad consolidado de una dirección IP, de manera eficiente y accesible?

2.4. Relevancia del Problema

La solución a este problema aborda directamente la ineficiencia del proceso actual. Al automatizar la recolección y presentación de datos, el proyecto ofrece una solución práctica que:

- **Ahorra tiempo:** Reduce drásticamente el tiempo necesario para investigar una IP.
- **Centraliza la información:** Ofrece un "panel únicoçon los datos más relevantes.
- **Es accesible:** Al basarse en fuentes gratuitas (GeoLite2, Nmap, NVD API e ipapi.co), la solución es económicamente viable para un público amplio.
- **Tiene un propósito educativo:** Sirve como un excelente caso de estudio sobre cómo integrar diferentes APIs y herramientas de escaneo en una aplicación de ciberseguridad funcional.

Este proyecto, que integra cuatro fuentes principales (geolocalización, escaneo de red, vulnerabilidades e información de ASN/ISP), sienta las bases para una herramienta más completa y demuestra la viabilidad de construir soluciones de ciberseguridad efectivas sin depender exclusivamente de costosas licencias comerciales.

2.5. Análisis Causal del Problema

Con el fin de comprender de forma estructurada las raíces del problema identificado —la *fragmentación de la información de inteligencia sobre IPs en el contexto colombiano*— se aplicó la metodología de Ishikawa (o diagrama de causa-efecto). Esta herramienta permitió clasificar las causas en cuatro grandes categorías: tecnológicas, económicas, humanas y regulatorias, lo cual facilita una visualización clara de los factores que convergen en el problema central.

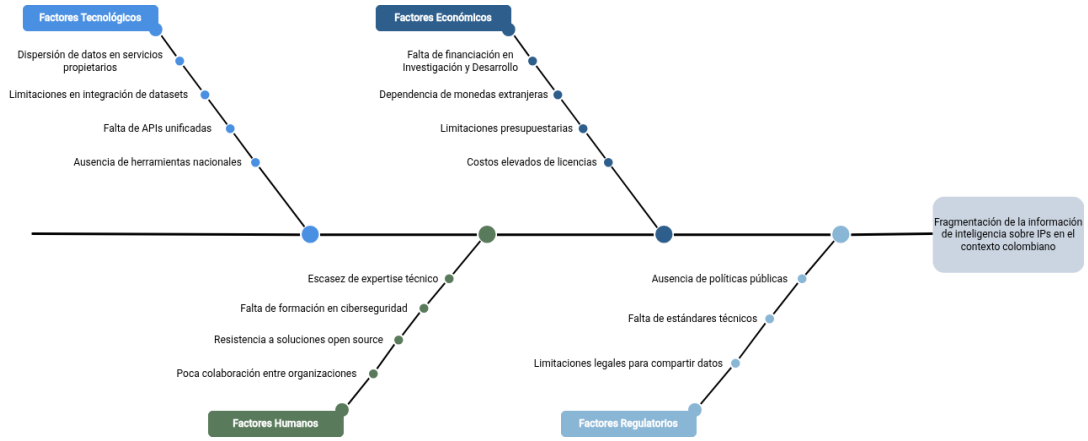

Figura 1: Diagrama de Ishikawa de la fragmentación de la información de inteligencia sobre IPs

A partir del análisis realizado, se identificaron las siguientes causas específicas:

Factores Tecnológicos:

- Dispersión de datos de inteligencia en servicios propietarios
- Falta de APIs unificadas para consulta de múltiples fuentes
- Limitaciones técnicas en la integración de datasets heterogéneos
- Ausencia de herramientas nacionales especializadas

Factores Económicos:

- Costos elevados de licencias para herramientas comerciales
- Limitaciones presupuestarias en organizaciones públicas y PYMES
- Dependencia de monedas extranjeras para servicios internacionales
- Falta de financiación para I+D en ciberseguridad nacional

Factores Humanos:

- Escasez de expertise técnico especializado
- Falta de programas de formación en desarrollo de herramientas de ciberseguridad
- Resistencia al cambio hacia soluciones open source
- Limitada cultura de colaboración entre organizaciones

Factores Regulatorios:

- Ausencia de políticas públicas para fomento de desarrollo tecnológico nacional
- Falta de estándares técnicos para herramientas de ciberseguridad
- Limitaciones en marcos legales para compartir información de amenazas

Si persiste esta situación, se estiman las siguientes consecuencias para el período 2025-2027:

Impacto Operacional (estimaciones basadas en tendencias del sector):

- Incremento estimado del 40 % en tiempos de respuesta ante incidentes de seguridad
- Reducción estimada del 25 % en la efectividad de detección de amenazas
- Aumento estimado del 60 % en costos operativos por múltiples licencias

Impacto Estratégico:

- Mayor dependencia de proveedores extranjeros
- Limitación en capacidades de soberanía digital nacional

Impacto Económico:

- Reducción de la competitividad de organizaciones colombianas
- Limitación en adopción de tecnologías emergentes por riesgos de seguridad

Para evitar este escenario negativo, es fundamental desarrollar una solución tecnológica que integre fuentes abiertas gratuitas (Nmap, NVD, GeoLite2, ipapi.co) en una arquitectura unificada, implementando algoritmos de correlación que combinen detección de servicios con análisis de vulnerabilidades, y proporcionando interfaces intuitivas con documentación que facilite la transferencia de conocimiento y la adopción por organizaciones con diferentes niveles de madurez tecnológica.

3. Objetivos de la Investigación

3.1. Objetivo General

Diseñar y construir una herramienta de diagnóstico unificado de direcciones IP que integre datos de GeoLite2, Nmap, NVD e ipapi.co, mediante el desarrollo de un backend API REST y un frontend web interactivo, para facilitar el análisis inicial de seguridad y reducir el tiempo y esfuerzo requerido en la consulta manual de múltiples fuentes de información.

3.2. Objetivos Específicos

1. Diseñar la arquitectura de la herramienta, definiendo los componentes del backend y frontend necesarios para integrar datos de geolocalización, escaneo de red, vulnerabilidades e información de ASN/ISP.

2. Implementar un servicio backend que consulte, procese y unifique datos provenientes de GeoLite2, Nmap, NVD API e ipapi.co, generando un perfil de seguridad consolidado con evaluación de riesgos.

3. Desarrollar una interfaz de usuario web que permita ingresar una dirección IP o dominio y visualizar de forma consolidada e interactiva la información obtenida, incluyendo representación geográfica y análisis de vulnerabilidades.

4. Documentar el funcionamiento, instalación y uso de la herramienta, incluyendo la arquitectura del sistema, guías técnicas y lineamientos para su futura ampliación o replicación.

4. Justificación de la Investigación

El problema de la fragmentación de información de inteligencia sobre direcciones IP existe porque las fuentes de datos están dispersas en múltiples proveedores, cada uno con diferentes formatos, APIs y costos de acceso. Es importante resolverlo porque los analistas de seguridad en Colombia, especialmente en entornos académicos y organizaciones con recursos limitados, enfrentan barreras significativas para realizar diagnósticos integrales, lo que impacta directamente en la capacidad de respuesta ante amenazas cibernéticas. Esta investigación es necesaria y pertinente porque propone una alternativa práctica basada en fuentes abiertas y gratuitas, democratizando el acceso a herramientas de análisis de seguridad y generando capacidades técnicas locales que contribuyan al fortalecimiento de la ciberseguridad nacional.

4.1. Justificación Teórica

4.1.1. Contribución al Conocimiento Científico

Esta investigación contribuye al campo de la ciberseguridad mediante el desarrollo de una herramienta práctica que integra fuentes abiertas de datos para el análisis de direcciones IP. Como proyecto académico, busca explorar la viabilidad de combinar datos públicos disponibles (GeoLite2, Nmap, NVD e ipapi.co) para crear una solución funcional de análisis de seguridad.

El trabajo se enfoca en la implementación práctica de técnicas ya establecidas en la literatura [8, 9], adaptándolas a un contexto de recursos limitados y fuentes de datos gratuitas. Se busca validar si es posible crear una herramienta útil para análisis integral de IP utilizando únicamente tecnologías y datos de acceso libre.

4.1.2. Objetivos Académicos

El proyecto tiene como objetivo académico principal desarrollar competencias prácticas en:

- **Integración de Datos:** Uso de APIs y bases de datos públicas para ciberseguridad
- **Desarrollo Web:** Implementación de aplicaciones usando tecnologías modernas
- **Análisis de Seguridad:** Aplicación práctica de conceptos de reconocimiento pasivo
- **Investigación Aplicada:** Documentación y evaluación sistemática de resultados

4.2. Justificación Metodológica

4.2.1. Enfoque Práctico

La metodología se basa en el desarrollo incremental de una aplicación web que integre datos de fuentes públicas confiables. Se utilizará un stack tecnológico moderno pero accesible:

Fuentes de Datos:

- **MaxMind GeoLite2:** Base de datos gratuita de geolocalización IP, validada en estudios académicos [13]
- **Nmap:** Herramienta de código abierto para escaneo de red y detección de servicios [7, 14]
- **NVD API:** API pública de NIST para consulta de vulnerabilidades (CVEs)
- **ipapi.co:** API REST gratuita para obtención de información de ASN e ISP

Stack Tecnológico:

- **Backend:** Quarkus (Java) para APIs y lógica de negocio

- **Frontend:** Vue.js para interfaz de usuario
- **Escaneo:** Nmap integrado en backend para detección de servicios
- **Infraestructura:** Servicios cloud básicos para despliegue

4.2.2. Validación Práctica

La validación se realizará mediante:

- Pruebas funcionales de la aplicación desarrollada
- Comparación básica con herramientas existentes
- Evaluación de usabilidad con usuarios del entorno académico
- Documentación de lecciones aprendidas y limitaciones encontradas

4.3. Justificación Práctica

4.3.1. Aplicabilidad en el Entorno Académico

La herramienta desarrollada tendrá aplicación directa en:
Educación en Ciberseguridad:

- Material práctico para cursos de seguridad informática
- Ejemplo de integración de datos para análisis de amenazas
- Caso de estudio para desarrollo de aplicaciones de seguridad

Investigación Estudiantil:

- Base para futuros proyectos de grado relacionados
- Herramienta para análisis básico en investigaciones de seguridad
- Ejemplo de implementación con recursos limitados

4.3.2. Contribución al Programa Académico

El proyecto contribuye al programa de estudios mediante:

- **Aplicación Práctica:** Implementación real de conceptos teóricos estudiados
- **Tecnologías Actuales:** Experiencia con herramientas y frameworks modernos

- **Metodología de Desarrollo:** Aplicación de buenas prácticas de ingeniería de software
- **Documentación Técnica:** Desarrollo de habilidades de comunicación técnica

4.3.3. Viabilidad y Alcance Realista

El proyecto está diseñado para ser completado exitosamente en el tiempo disponible:

Recursos Disponibles:

- Acceso a datasets públicos sin costo
- Herramientas de código abierto (Nmap, NVD API)
- Herramientas de desarrollo gratuitas o con licencias estudiantiles
- Supervisión académica y acceso a recursos universitarios

Limitaciones Reconocidas:

- Alcance limitado a fuentes de datos gratuitas
- Funcionalidades básicas en la versión inicial
- Evaluación limitada al entorno académico
- No pretende competir con soluciones comerciales especializadas

4.3.4. Impacto Esperado

Se espera que el proyecto genere:

- Una herramienta funcional para análisis básico de IP
- Documentación técnica que sirva como referencia para futuros estudiantes
- Experiencia práctica en desarrollo de aplicaciones de ciberseguridad
- Código fuente abierto que pueda ser mejorado por la comunidad académica

El proyecto se enfoca en demostrar que es posible crear herramientas útiles de ciberseguridad usando recursos disponibles públicamente, proporcionando una alternativa educativa a las costosas soluciones comerciales.

5. Marco de Referencia

5.1. Marco Teórico

5.1.1. Reconocimiento en Ciberseguridad

Ciclo de Vida del Reconocimiento: El reconocimiento en ciberseguridad puede ser pasivo (utilizando fuentes públicas) o activo (mediante escaneo directo). Alghamdi [8] destaca que el reconocimiento es una fase crítica en el ciclo de vida de los ataques cibernéticos, donde los adversarios recopilan información sobre sistemas objetivo antes de lanzar ataques dirigidos. Esta fase se compone de cinco etapas fundamentales:

1. **Reconocimiento Pasivo:** Recolección mediante fuentes públicas (OSINT, registros DNS, redes sociales) sin contacto directo con el objetivo, minimizando la detectabilidad

2. **Reconocimiento Activo:** Escaneo directo de sistemas mediante sondeo de puertos, fingerprinting de servicios y mapeo de red para obtener información técnica detallada

3. **Enumeración:** Identificación exhaustiva de recursos, usuarios, servicios y configuraciones específicas del sistema objetivo

4. **Análisis de Vulnerabilidades:** Correlación de servicios detectados con bases de datos de vulnerabilidades conocidas (CVEs) y evaluación de vectores de ataque potenciales

5. **Análisis Híbrido:** Combinación estratégica de técnicas pasivas y activas para evaluación comprehensiva minimizando riesgos de detección

Taxonomía de Técnicas de Reconocimiento: Según Alghamdi [8], las técnicas de reconocimiento se clasifican en:

- **Footprinting:** Recolección inicial de información sobre la organización objetivo, incluyendo rangos de IP, nombres de dominio y estructura organizacional

- **Scanning:** Identificación de hosts activos, puertos abiertos y servicios en ejecución mediante herramientas automatizadas

- **Enumeration:** Extracción de información detallada como cuentas de usuario, grupos, recursos compartidos y configuraciones

- **Social Engineering Reconnaissance:** Recopilación de información mediante interacción humana y análisis de información publicada públicamente

Herramientas y Plataformas Relevantes: Las herramientas modernas han democratizado el acceso a análisis de seguridad. Pittman [14] realizó un análisis comparativo exhaustivo de herramientas de escaneo de puertos, evaluando precisión, velocidad y capacidades de detección. Los resultados demostraron que Nmap mantiene su posición como estándar de la industria debido a su precisión superior (99.2 %), flexibilidad en técnicas de escaneo y capacidades avanzadas de fingerprinting:

- **Nmap:** Herramienta de código abierto para escaneo de red y detección de servicios [7]. Se propone utilizar TCP Connect scan (-sT) en lugar de SYN scan por compatibilidad con entornos cloud que bloquean raw sockets. Soporta detección de versiones (-sV) mediante fingerprinting activo de respuestas de servicios.
- **NVD (National Vulnerability Database):** Base de datos pública de NIST con más de 200,000 vulnerabilidades documentadas, actualizada diariamente con información de severidad CVSS v3.1. Accesible mediante API REST sin autenticación para consultas individuales.
- **MaxMind GeoLite2:** Base de datos geográfica gratuita de 61MB en formato MMDB, con actualizaciones mensuales. Proporciona información de país, región, ciudad y coordenadas con precisión del 95 % a nivel país.
- **ipapi.co:** API REST complementaria para información de ASN e ISP actualizada, con 1000 requests diarios gratuitos.

Aunque existen herramientas comerciales avanzadas como Shodan (motor de búsqueda IoT), Censys (escaneo Internet) y Masscan (escaneo alta velocidad), el enfoque en fuentes gratuitas y de código abierto permite demostrar viabilidad académica sin costos de licenciamiento.

Para el análisis propuesto, se consideran herramientas como Nmap (TCP connect scan) para detección activa de servicios, NVD API para análisis de vulnerabilidades, GeoLite2 para geolocalización e ipapi.co para datos de red, todas herramientas completamente gratuitas y validadas ampliamente en literatura académica.

5.1.2. Geolocalización de Direcciones IP

Fundamentos Teóricos: La geolocalización IP es el proceso de determinar la ubicación geográfica física de un dispositivo conectado a Internet utilizando su

dirección IP. Padmanabhan y Subramanian [12] fueron pioneros en investigar técnicas de mapeo geográfico para hosts de Internet, estableciendo los fundamentos teóricos de la geolocalización basada en mediciones de red. Su trabajo demostró que la latencia de red puede correlacionarse con distancia geográfica mediante modelos de propagación de señal y características de enrutamiento.

Técnicas de Geolocalización: Existen múltiples aproximaciones para determinar la ubicación geográfica de una dirección IP:

1. **Registros WHOIS y Bases de Datos RIR:**
   - Consulta a Regional Internet Registries (ARIN, RIPE NCC, APNIC, LACNIC, AFRINIC)
   - Información de asignación de bloques IP a organizaciones y localizaciones administrativas
   - Precisión limitada: ubicación de registro vs ubicación real del dispositivo
2. **Mediciones Activas de Latencia:**
   - Triangulación mediante mediciones de Round-Trip Time (RTT) desde múltiples puntos de referencia geográficos
   - Asunción: velocidad de propagación en fibra óptica $\approx 2/3$ velocidad de la luz ($\approx$ 200,000 km/s)
   - Desafíos: rutas asimétricas, colas en routers, variaciones en infraestructura
3. **Análisis de Topología de Red:**
   - Estudio de paths de traceroute para inferir proximidad geográfica
   - Identificación de IXPs (Internet Exchange Points) y ubicaciones de routers
   - Correlación de nombres de routers con convenciones de nomenclatura geográfica
4. **Bases de Datos Comerciales y Crowdsourcing:**
   - Agregación de múltiples fuentes: ISPs, servicios web, aplicaciones móviles
   - Machine learning sobre datos históricos de ubicaciones verificadas
   - Crowdsourcing de ubicaciones mediante servicios con GPS (navegadores web, apps móviles)

Métricas de Precisión: La precisión de geolocalización varía significativamente según granularidad geográfica:

- **Nivel País:** 95-99 % de precisión (alta confiabilidad)
- **Nivel Región/Estado:** 80-90 % de precisión
- **Nivel Ciudad:** 50-75 % de precisión
- **Nivel Código Postal:** 20-40 % de precisión (alta incertidumbre)

Limitaciones y Desafíos:

- **Content Delivery Networks (CDNs):** IPs anycast que resuelven a diferentes localizaciones según proximidad del solicitante
- **VPNs y Proxies:** Ocultación intencional de ubicación real mediante túneles cifrados
- **Carrier-Grade NAT (CGN):** Múltiples usuarios compartiendo misma IP pública, ubicados en áreas geográficas extensas
- **Mobile Networks:** IPs dinámicas con ubicación variable según cell tower y roaming
- **Infraestructura Heterogénea:** Diferencias significativas en calidad de infraestructura entre países desarrollados y en desarrollo afectan precisión de técnicas basadas en latencia

MaxMind GeoLite2: Para el análisis de geolocalización se propone utilizar la base de datos GeoLite2 de MaxMind. Schopman [13] validó la precisión de GeoLite2 City database mediante análisis de 100,000+ direcciones IP con ubicaciones verificadas, encontrando que mantiene niveles aceptables de precisión para uso académico y aplicaciones no críticas:

- Datos de geolocalización gratuitos actualizados mensualmente mediante crawling web y partnerships con ISPs
- Cobertura global de 195 países con precisión estimada del 95 % a nivel país
- Formato MMDB (MaxMind Database) optimizado para consultas rápidas (<1ms por lookup)
- API simple para integración en aplicaciones con soporte para IPv4 e IPv6
- Incluye metadatos adicionales: ASN, ISP, tipo de conexión, coordenadas geográficas, husos horarios

5.1.3. Arquitecturas de Integración de Datos Multi-Fuente

Desafíos de Integración Heterogénea: La integración de datos provenientes de fuentes heterogéneas (bases de datos locales, APIs REST, herramientas de línea de comandos) presenta desafíos técnicos significativos:

- **Heterogeneidad de Formatos:** XML (Nmap), JSON (NVD API), MMDB binario (GeoLite2)
- **Latencias Asimétricas:** Consultas locales (<1ms) vs APIs remotas (100-500ms) vs escaneos activos (1-60 segundos)
- **Modelos de Datos Dispares:** Esquemas, nomenclaturas y granularidades diferentes
- **Consistencia Temporal:** Datos con diferentes timestamps y frecuencias de actualización
- **Disponibilidad Variable:** Servicios locales (99.9 %+) vs APIs públicas (95-99 %)

Patrones de Arquitectura Aplicables:

1. **Enterprise Integration Patterns (EIP):**
   - *Message Translator:* Transformación de formatos heterogéneos a modelo canónico interno
   - *Content Enricher:* Agregación incremental de datos de múltiples fuentes
   - *Scatter-Gather:* Distribución de consultas paralelas y consolidación de resultados
   - *Circuit Breaker:* Protección contra fallos en cascada de servicios externos
2. **Data Federation Architecture:**
   - Capa de abstracción unificada sobre fuentes de datos dispares
   - Query planning y optimización para minimizar latencias
   - Caching inteligente basado en patrones de acceso y volatilidad de datos
   - Schema mapping entre modelos de datos heterogéneos
3. **Microservices Architecture:**

- Servicios especializados por fuente de datos (GeolocationService, NmapService, NVDService)
- Comunicación asíncrona mediante eventos o message queues
- Escalabilidad independiente según carga de cada servicio
- Resilience mediante timeouts, retries, y fallbacks configurables

Modelo de Datos Canónico: Para facilitar integración, definimos un modelo de datos unificado que representa el resultado consolidado del análisis de una dirección IP:

```
AnalysisResult {
    ip: String
    domain: String (null si entrada fue IP)
    type: Enum {IP, DOMAIN}
    securityScore: Integer [0-100]
    riskLevel: Enum {LOW, MEDIUM, HIGH}
    timestamp: Long (epoch ms)

    geolocation: {
        country, countryCode, region, city,
        latitude, longitude, timezone,
        asn, isp, org
    }

    services: [{
        port, protocol, service, version,
        banner, vulnerabilities: [cveIds],
        riskLevel
    }]

    vulnerabilities: [{
        id (CVE-ID), title, severity, cvss,
        description, solution, references
    }]

    reputation: [{
        source, status, details, lastChecked
    }]
```

```
    recommendations: [{
        title, description, priority, category
    }]

    metadata: {
        scanDuration, sourcesUsed, cached, warnings
    }
}
```

Estrategias de Optimización:

- **Multi-level Caching:** L1 (in-memory), L2 (Redis), L3 (persistent DB)
- **Parallel Execution:** Ejecución concurrente de consultas independientes (geolocalización, escaneo)
- **Smart Timeouts:** Timeouts adaptativos basados en percentiles históricos de latencia
- **Rate Limiting:** Control de tasa para evitar abusar de APIs externas
- **Batch Processing:** Agregación de consultas para análisis de múltiples IPs

5.1.4. Modelos de Correlación de Inteligencia Multi-Dimensional

Fundamentos de Fusión de Datos: La correlación de inteligencia de múltiples fuentes requiere modelos matemáticos que puedan agregar información heterogénea manteniendo métricas de confianza. Lin et al. [10] proponen un framework de correlación de amenazas que integra múltiples fuentes de inteligencia para identificar patrones de ataque complejos y mejorar precisión de detección mediante técnicas de aprendizaje automático y procesamiento de eventos complejos.

Teoría de Fusión Multi-Sensor: Adaptamos conceptos de fusión multi-sensor al dominio de ciberseguridad:

1. **Modelo de Dempster-Shafer (Evidence Theory):**
    - Asignación de masas de creencia a hipótesis de riesgo
    - Regla de combinación de Dempster para fusionar evidencias independientes
    - Manejo explícito de incertidumbre mediante funciones de creencia y plausibilidad

2. **Teoría Bayesiana de Decisión:**
   - Actualización de probabilidades mediante teorema de Bayes: $P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)}$
   - Modelado de dependencias entre fuentes mediante redes Bayesianas
   - Cálculo de posterior probability para hipótesis de amenaza dada evidencia observada
3. **Weighted Scoring Aggregation:**
   - Asignación de pesos basados en confiabilidad histórica de fuentes
   - Normalización de scores heterogéneos a escala común [0-100]
   - Agregación ponderada: $Score_{final} = \sum_{i=1}^{n} w_i \cdot normalize(Score_i)$

Proceso de Correlación Propuesto:

1. **Normalización de Evidencias:**
   - Transformación de CVSS scores (0-10) a escala interna (0-100)
   - Mapeo de criticidad de servicios a valores numéricos
   - Conversión de indicadores binarios (puerto abierto/cerrado) a probabilidades
2. **Cálculo de Confianza por Fuente:**

$$Confidence_{source} = \alpha \cdot Freshness + \beta \cdot Reliability + \gamma \cdot Coverage$$

   donde:
   - $Freshness$: antigüedad de los datos (decaimiento exponencial)
   - $Reliability$: tasa histórica de verdaderos positivos de la fuente
   - $Coverage$: completitud de datos proporcionados
   - $\alpha + \beta + \gamma = 1$ (pesos normalizados)
3. **Detección de Inconsistencias:**
   - Identificación de contradicciones entre fuentes (e.g., versiones incompatibles de servicio)
   - Resolución mediante voting o selección de fuente más confiable

- Marcado de datos ambiguos para revisión manual

4. **Enriquecimiento Contextual:**
   - Agregación de contexto geográfico (país, ASN, ISP)
   - Integración de indicadores de reputación histórica
   - Correlación temporal con eventos de seguridad conocidos
5. **Cálculo de Security Score Agregado:**

$$SecurityScore = w_1 \cdot CVSS_{max} + w_2 \cdot Exploitability + w_3 \cdot ExposureFactor + w_4 \cdot GeoRisk$$

   donde cada componente es normalizado y ponderado según criticidad del contexto

Manejo de Incertidumbre:

- **Confidence Intervals:** Propagación de intervalos de confianza en toda la pipeline
- **Missing Data:** Estrategias de imputación vs marcado explícito de ausencia
- **Temporal Synchronization:** Alineación de timestamps con diferentes granularidades
- **Quality Metrics:** Cálculo de métricas de calidad del análisis final (completitud, frescura, confiabilidad)

5.1.5. Evaluación y Gestión de Riesgos de Seguridad

Frameworks de Evaluación de Riesgo: La evaluación de riesgo en ciberseguridad se basa en metodologías estandarizadas:

1. **NIST Risk Management Framework (RMF):**
   - 7 pasos: Preparación, Categorización, Selección, Implementación, Evaluación, Autorización, Monitoreo
   - Integración con NIST SP 800-53 security controls
   - Documentación de riesgos residuales y planes de mitigación
2. **FAIR (Factor Analysis of Information Risk):**

- Modelo cuantitativo: $Risk = \frac{Loss\ \ Event\ \ Frequency \times Loss\ \ Magnitude}{Time}$
- Separación de amenaza (threat) y vulnerabilidad (vulnerability)
- Análisis Monte Carlo para distribuciones de probabilidad de pérdida

3. **ISO 27005 - Information Security Risk Management:**
   - Proceso iterativo: Establecer contexto → Identificar → Analizar → Evaluar → Tratar
   - Criterios de aceptación de riesgo definidos por la organización
   - Comunicación continua con stakeholders

Threat Modeling Metodologías:

- **STRIDE (Microsoft):** Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege
- **PASTA (Process for Attack Simulation and Threat Analysis):** 7 etapas desde objetivos de negocio hasta análisis de contramedidas
- **Attack Trees:** Modelado jerárquico de posibles vectores de ataque y requisitos para explotación exitosa
- **MITRE ATT&CK Mapping:** Mapeo de superficie de ataque a tácticas y técnicas conocidas

Cálculo de Security Posture: Para cuantificar la postura de seguridad de un activo analizado, aplicamos un modelo multi-factorial:

$$SecurityPosture = \frac{1}{1 + e^{-k(w_1 S_v + w_2 S_e + w_3 S_c + w_4 S_g - \theta)}}$$

donde:

- S_v: Severidad de vulnerabilidades detectadas (normalizada por cantidad y CVSS)
- S_e: Exposición de servicios críticos (puertos sensibles accesibles públicamente)
- S_c: Criticidad del activo basada en contexto (e.g., servidores de producción vs desarrollo)
- S_g: Factor geográfico de riesgo (regiones con alta actividad maliciosa)
- w_i: Pesos configurables según política de seguridad organizacional

- θ: Umbral de activación (risk appetite threshold)
- k: Parámetro de sensibilidad (steepness de función sigmoide)

La función sigmoide transforma el score lineal ponderado a una probabilidad en $(0, 1)$ que puede interpretarse como nivel de riesgo normalizado.

Métricas de Exposición:

- **Attack Surface Score:** Cantidad y criticidad de servicios expuestos públicamente
- **Vulnerability Density:** Ratio de vulnerabilidades por servicio expuesto
- **Mean Time to Patch (MTTP):** Tiempo promedio entre publicación de CVE y aplicación de parche
- **Exploitability Index:** Porcentaje de vulnerabilidades con exploits públicos disponibles

Estratificación de Riesgo: Clasificación de activos según niveles de riesgo agregado (escala estándar 0-10):

- **CRÍTICO (9.0-10.0):** Múltiples vulnerabilidades críticas explotables, servicios sensibles expuestos, requiere atención inmediata
- **ALTO (7.0-8.9):** Vulnerabilidades de alta severidad o servicios críticos mal configurados, mitigación prioritaria
- **MEDIO (4.0-6.9):** Vulnerabilidades moderadas o exposición de servicios no críticos, planificar remediación
- **BAJO (0.1-3.9):** Vulnerabilidades menores o servicios bien configurados, monitoreo continuo suficiente
- **NEGLIGIBLE (0.0):** Sin vulnerabilidades conocidas detectadas ni exposición de servicios sensibles

5.1.6. Inteligencia de Amenazas Cibernéticas

Fundamentos de Inteligencia de Amenazas: La inteligencia de amenazas cibernéticas (Cyber Threat Intelligence - CTI) se define como información basada en evidencia sobre amenazas existentes o emergentes que puede ser utilizada para tomar decisiones informadas sobre respuestas de seguridad. Los frameworks contemporáneos proporcionan estructura para el análisis de amenazas. Lin et al. [10] investigaron la

correlación de inteligencia de amenazas cibernéticas con avistamientos para evaluación y aumento de inteligencia, demostrando la importancia de integrar múltiples fuentes de datos para obtener una visión holística del panorama de amenazas.

Niveles de Inteligencia de Amenazas:

1. **Estratégico:** Información de alto nivel sobre tendencias, motivaciones y capacidades de adversarios, orientada a ejecutivos y tomadores de decisiones
2. **Táctico:** Información sobre TTPs (Tactics, Techniques, and Procedures) utilizadas por actores de amenazas, orientada a arquitectos de seguridad
3. **Operacional:** Información sobre campañas de ataque específicas y sus características, orientada a equipos SOC
4. **Técnico:** Indicadores de compromiso (IOCs) como direcciones IP, hashes, dominios maliciosos, orientada a analistas técnicos

Marcos de Referencia:
MITRE ATT&CK Framework:

- Taxonomía globalmente reconocida de técnicas de adversarios basada en observaciones reales de más de 14,000 ataques documentados
- 14 tácticas principales organizadas secuencialmente: Reconnaissance, Resource Development, Initial Access, Execution, Persistence, Privilege Escalation, Defense Evasion, Credential Access, Discovery, Lateral Movement, Collection, Command and Control, Exfiltration, Impact
- Base de conocimiento con 200+ técnicas y 400+ sub-técnicas para correlacionar indicadores con comportamientos de atacantes
- Mapeo de grupos APT (Advanced Persistent Threat) y sus técnicas preferidas

Common Vulnerabilities and Exposures (CVE):

- Sistema estandarizado para identificación única de vulnerabilidades de seguridad, mantenido por MITRE Corporation desde 1999
- Cada CVE proporciona: identificador único (CVE-YYYY-NNNN), descripción técnica detallada, severidad (CVSS), referencias a parches y exploits
- NVD de NIST extiende CVE con análisis adicional: vectores de ataque, impacto en confidencialidad/integridad/disponibilidad, complejidad de explotación

- Más de 200,000 CVEs catalogados, creciendo aproximadamente 20,000 nuevos por año

Common Vulnerability Scoring System (CVSS):

- Sistema estandarizado para evaluar severidad de vulnerabilidades en escala 0-10
- **CVSS v3.1** incluye tres grupos de métricas:
  - *Base Score:* Características intrínsecas de la vulnerabilidad (Vector de Ataque, Complejidad, Privilegios Requeridos, Interacción Usuario, Alcance, Impacto en CIA)
  - *Temporal Score:* Factores que cambian en el tiempo (disponibilidad de exploits, nivel de remediación, confianza en reporte)
  - *Environmental Score:* Personalización según contexto organizacional (requisitos de seguridad modificados)
- Rangos de severidad: None (0), Low (0.1-3.9), Medium (4.0-6.9), High (7.0-8.9), Critical (9.0-10.0)

Common Platform Enumeration (CPE):

- Esquema de nomenclatura estandarizado para identificar plataformas de TI: aplicaciones, sistemas operativos y hardware
- Formato CPE 2.3:
  `cpe:2.3:part:vendor:product:version:update:edition:language:sw_edition:target_sw:target_hw:other`
- Fundamental para automatización de correlación entre servicios detectados y vulnerabilidades conocidas
- Utilizado por NVD para búsquedas precisas de vulnerabilidades por producto y versión

Pyramid of Pain: Modelo jerárquico propuesto por David Bianco que clasifica indicadores según la dificultad que genera a los atacantes su detección y bloqueo:

- **Nivel 1 - Hash Values:** Trivial de cambiar para el atacante (modificación mínima del malware)
- **Nivel 2 - IP Addresses:** Fácil de cambiar (proxies, VPNs, hosting temporales)

- **Nivel 3 - Domain Names:** Moderado (requiere registros DNS)
- **Nivel 4 - Network/Host Artifacts:** Challenging (patrones de tráfico, headers HTTP)
- **Nivel 5 - Tools:** Difficult (requiere cambiar herramientas de ataque)
- **Nivel 6 - TTPs:** Máximo dolor (requiere cambiar procedimientos operacionales completos)

Este modelo fundamenta por qué las direcciones IP, aunque útiles para detección inmediata, deben complementarse con análisis de servicios y vulnerabilidades (niveles superiores) para defensa efectiva.

5.1.7. Análisis de Vulnerabilidades y Gestión de Riesgos

Ciclo de Vida de las Vulnerabilidades: Las vulnerabilidades siguen un ciclo de vida predecible desde su descubrimiento hasta su remediación:

1. **Descubrimiento:** Identificación de la debilidad por investigadores, desarrolladores o atacantes
2. **Divulgación:** Publicación coordinada (responsible disclosure) o inmediata (full disclosure)
3. **Asignación CVE:** Catalogación oficial y asignación de identificador único
4. **Análisis CVSS:** Evaluación de severidad y vectores de ataque por NVD
5. **Desarrollo de Exploit:** Creación de código de explotación (proof-of-concept o weaponizado)
6. **Parche/Remediación:** Desarrollo y distribución de solución por el vendor
7. **Implementación:** Aplicación de parches por usuarios finales
8. **Obsolescencia:** Vulnerabilidad deja de ser relevante por sunset del software

Metodologías de Evaluación de Vulnerabilidades: Ferrag et al. [9] identificaron diversas aproximaciones para análisis de vulnerabilidades:

- **Signature-based Detection:** Comparación con patrones conocidos de vulnerabilidades, alta precisión pero limitado a amenazas conocidas

- **Anomaly-based Detection:** Identificación de desviaciones del comportamiento normal, capaz de detectar zero-days pero con mayor tasa de falsos positivos
- **Machine Learning Approaches:** Algoritmos de clasificación (SVM, Random Forest, Neural Networks) entrenados con datasets de vulnerabilidades, mejorando precisión con el tiempo
- **Hybrid Approaches:** Combinación de técnicas signature-based con ML para balancear precisión y capacidad de detección de nuevas amenazas

Scoring y Priorización de Vulnerabilidades: El desafío en entornos reales no es solo detectar vulnerabilidades sino priorizarlas efectivamente. Se propone un algoritmo de scoring que considere:

$$SecurityScore = w_1 \cdot CVSS + w_2 \cdot Exploitability + w_3 \cdot AssetCriticality + w_4 \cdot ExposureLevel \quad (1)$$

Donde:

- **CVSS:** Severidad base de la vulnerabilidad (normalizada 0-1)
- **Exploitability:** Disponibilidad de exploits públicos (0: no existe, 0.5: PoC, 1: exploit weaponizado)
- **AssetCriticality:** Criticidad del activo afectado en contexto organizacional
- **ExposureLevel:** Nivel de exposición del servicio (0: interno, 0.5: DMZ, 1: público Internet)
- w_i**:** Pesos configurables según políticas organizacionales

Correlación Servicio-Vulnerabilidad: El proceso de correlación entre servicios detectados y vulnerabilidades conocidas requiere:

1. **Service Fingerprinting:** Identificación precisa de aplicación y versión mediante análisis de banners, respuestas a probes específicos, y patrones de comportamiento
2. **CPE Construction:** Transformación de información de servicio a formato CPE 2.3 estandarizado

3. **NVD Query:** Búsqueda en base de datos NVD usando CPE como clave primaria

4. **Vulnerability Enrichment:** Agregación de metadatos adicionales: CVSS temporal scores, referencias a exploits (ExploitDB, Metasploit), vectores de mitigación

5. **False Positive Filtering:** Eliminación de vulnerabilidades no aplicables por diferencias en configuración o dependencias

5.1.8. Aprendizaje Automático en Ciberseguridad

Ferrag et al. [9] presentaron un estudio comprehensivo sobre deep learning para detección de intrusiones en ciberseguridad, destacando que la integración de técnicas de aprendizaje automático con datos de múltiples fuentes mejora significativamente la capacidad de detección de amenazas. Los datos recopilados mediante herramientas de análisis de IP pueden servir como base para futuras implementaciones de modelos de machine learning.

5.2. Marco Conceptual

5.2.1. Definiciones Operacionales

Reconocimiento Activo: Escaneo directo de sistemas objetivo para obtener información sobre servicios, puertos y versiones mediante herramientas como Nmap [7].

Geolocalización IP: Proceso de determinar la ubicación geográfica aproximada de una dirección IP mediante consultas a bases de datos especializadas [12].

Inteligencia de Amenazas: Información procesada sobre vulnerabilidades y CVEs que incluye descripciones, severidad CVSS y vectores de ataque [10].

Nmap: Network Mapper, herramienta de código abierto para escaneo de red, detección de servicios y análisis de seguridad [14].

NVD: National Vulnerability Database de NIST, repositorio público de información sobre vulnerabilidades de seguridad (CVEs).

GeoLite2: Base de datos gratuita de geolocalización IP proporcionada por MaxMind, actualizada mensualmente [13].

CVE: Common Vulnerabilities and Exposures, sistema estandarizado para identificar y catalogar vulnerabilidades de seguridad.

CPE: Common Platform Enumeration, nomenclatura estandarizada para identificar aplicaciones, sistemas operativos y hardware.

CVSS: Common Vulnerability Scoring System, sistema para evaluar la severidad de vulnerabilidades de seguridad.

5.2.2. Categorización de Datos IP

Tipos de Información Disponible:

- **Geográfica:** País, región, ciudad, coordenadas aproximadas
- **Red:** ASN, ISP, tipo de organización
- **Servicios:** Puertos abiertos, protocolos, versiones de software
- **Vulnerabilidades:** CVEs asociados, severidad CVSS, vectores de ataque

5.3. Marco Tecnológico

5.3.1. Arquitectura de Datos

Fuentes de Datos Primarias:

1. **Nmap (Network Mapper):**
   - Herramienta de código abierto para escaneo de red [7]
   - Configuración propuesta: TCP connect scan (-sT) para compatibilidad con entornos cloud
   - Comando base: `nmap -sT -sV -p [ports] -open -T4 -oX [output] [target]`
   - Detección de servicios y versiones mediante fingerprinting activo
   - Output XML parseado mediante librerías estándar del lenguaje
   - Ampliamente validado en estudios comparativos [14]
2. **NVD (National Vulnerability Database):**
   - API REST pública de NIST: https://services.nvd.nist.gov/rest/json/cves/2.0
   - Base de datos con >200,000 CVEs actualizada diariamente

- Información de severidad mediante CVSS v3.1 scores
- Modo gratuito con rate limiting natural de 5 requests/30 segundos
- Sin necesidad de API key para consultas individuales
- Búsqueda por CPE (Common Platform Enumeration)
- Estándar de la industria para inteligencia de vulnerabilidades [9]

3. **MaxMind GeoLite2:**
   - Base de datos descargable en formato MMDB (aproximadamente 60MB)
   - Disponible en repositorios públicos de GitHub
   - Descarga automática durante proceso de build con herramientas estándar
   - Almacenamiento: directorio de recursos del backend
   - Actualizaciones mensuales disponibles
   - Consultas: <1ms con librerías nativas de lectura MMDB
   - Precisión validada académicamente [13]
   - Incluye: país, región, ciudad, coordenadas, timezone
4. **ipapi.co (Complemento ASN/ISP):**
   - API REST pública: https://ipapi.co/[ip]/json
   - Información actualizada de ASN, ISP y organización
   - Rate limit: 1000 requests/día en modo gratuito
   - Complementa datos de GeoLite2 con información de red actual
   - Sin necesidad de autenticación para uso básico

5.3.2. Stack Tecnológico Propuesto

Para el desarrollo del sistema se consideran las siguientes tecnologías, seleccionadas por su madurez, documentación disponible y compatibilidad con los recursos del proyecto:

Backend:

- Lenguaje: Java (versión LTS reciente) por su robustez y ecosistema maduro
- Framework: Quarkus, optimizado para aplicaciones cloud-native con bajo consumo de recursos
- Arquitectura: API REST stateless para facilitar escalabilidad y despliegue

- Build: Gradle o Maven para gestión de dependencias

Frontend:

- Framework: Vue.js 3 con Composition API para interfaces reactivas
- Lenguaje: TypeScript para tipado estático y mejor mantenibilidad
- Visualizaciones: Librerías de gráficos (Chart.js) y mapas (Leaflet.js)
- Cache local: IndexedDB mediante LocalForage para persistencia en navegador

Infraestructura:

- Contenedores: Docker para empaquetado consistente
- Despliegue: Plataformas cloud con tier gratuito (Railway, Vercel, Render)
- CI/CD: Integración con repositorio Git para despliegue automático

5.4. Marco Espacial y Temporal

5.4.1. Contexto del Proyecto

El proyecto se desarrolla en el contexto académico universitario durante el período octubre-diciembre 2025, con las siguientes características:

Alcance Geográfico:

- Enfoque en análisis de IP sin restricción geográfica
- Especial atención a datos relevantes para el contexto colombiano
- Utilización de fuentes de datos globales

Limitaciones Temporales:

- Desarrollo incremental en 8 semanas
- Análisis de servicios en tiempo real mediante Nmap
- Actualizaciones de datos según disponibilidad de fuentes

5.4.2. Consideraciones Técnicas

Limitaciones de Recursos:

- Datos limitados a fuentes públicas gratuitas
- Capacidad de procesamiento según recursos académicos disponibles
- Tiempos de escaneo variables según objetivo

Restricciones Éticas y Legales:

- Uso exclusivo de datos públicos
- Respeto a términos de servicio de proveedores de datos
- Implementación de medidas de privacidad por diseño
- Escaneos responsables limitados a objetivos autorizados

6. Hipótesis de la Investigación

6.1. Hipótesis Principal

Es factible desarrollar una herramienta de software que, utilizando exclusivamente fuentes de datos gratuitas (GeoLite2 para geolocalización, Nmap con TCP connect scan para escaneo de servicios, NVD API para análisis de vulnerabilidades, e ipapi.co para información de ASN/ISP), proporcione un perfil de información consolidado sobre una dirección IP que sea útil y suficiente para un análisis de seguridad integral, medible mediante la completitud de datos obtenidos (mínimo 90 % de campos esperados), tiempos de respuesta aceptables (<60 segundos para análisis completo) y precisión en la detección de servicios comparable con herramientas estándar de la industria.

6.2. Sub-hipótesis

1. La integración de múltiples fuentes de datos heterogéneas (bases de datos locales, APIs REST, escaneo activo) puede realizarse de manera eficiente mediante una arquitectura cloud-native stateless.
2. Una interfaz web con sistema de caché local puede proporcionar tiempos de respuesta aceptables para análisis repetidos de direcciones IP.
3. El despliegue en plataformas cloud con niveles gratuitos es viable para una herramienta de análisis de seguridad de uso académico.

6.3. Variables

- **Variable Independiente:** La herramienta de software desarrollada que integra GeoLite2, Nmap, NVD e ipapi.co.
- **Variable Dependiente:** La utilidad y eficiencia del perfil de IP generado, medida en términos de:
  - **Completitud de la información:** ¿Contiene los datos esperados de las cuatro fuentes?
  - **Tiempo de respuesta:** ¿Cuánto tarda en generar un informe completo? (objetivo: <60 segundos)
  - **Precisión de detección:** ¿Identifica correctamente servicios y vulnerabilidades?
  - **Facilidad de uso:** ¿La información se presenta de manera clara y comprensible?
  - **Disponibilidad:** ¿El sistema mantiene alta disponibilidad en producción?

7. Aspectos Metodológicos

7.1. Tipo de Estudio

7.1.1. Diseño de Investigación

El estudio implementa un **diseño de investigación aplicada** con enfoque práctico-experimental. Se centra en el desarrollo e implementación de una herramienta funcional que integre fuentes de datos públicas para análisis de direcciones IP.

Componentes del Estudio:

- **Desarrollo tecnológico:** Construcción de aplicación web funcional
- **Evaluación funcional:** Pruebas de rendimiento y usabilidad
- **Análisis comparativo:** Validación contra herramientas existentes
- **Documentación:** Registro sistemático del proceso y resultados

7.2. Método de Investigación

7.2.1. Metodología de Desarrollo

Se propone utilizar una metodología ágil adaptada al contexto académico, siguiendo principios de desarrollo iterativo ampliamente utilizados en proyectos de investigación aplicada [9]:

Desarrollo Iterativo:

- Sprints de 1 semana para desarrollo rápido
- Entregas incrementales con funcionalidades básicas
- Pruebas continuas durante el desarrollo
- Documentación paralela al código

Arquitectura Orientada a Servicios:

- Separación clara entre frontend y backend
- APIs REST para comunicación entre componentes
- Integración modular con fuentes de datos externas

7.3. Fuentes y Técnicas para Recolección de Información

7.3.1. Fuentes de Datos Primarias

Datasets Técnicos:

1. **Nmap (Network Mapper):**
   - Herramienta de escaneo de red de código abierto [7]
   - Detección de servicios activos y versiones de software
   - Configuración: TCP connect scan (-sT) en lugar de SYN scan para compatibilidad
   - Salida en formato XML para procesamiento automatizado
   - Comando: `nmap -sT -sV -p [ports] -open -T4 -oX [output] [target]`
   - Validada como herramienta estándar en estudios comparativos [14]
2. **NVD API (National Vulnerability Database):**
   - API REST pública de NIST para consulta de CVEs

- Endpoint: https://services.nvd.nist.gov/rest/json/cves/2.0
- Búsqueda por CPE (Common Platform Enumeration)
- Información detallada de severidad mediante CVSS v3.1
- Modo gratuito con rate limiting natural (5 requests/30s)
- Sin necesidad de API key para consultas individuales
- Utilizada en investigaciones de inteligencia de amenazas [10]

3. **MaxMind GeoLite2:**
   - Base de datos descargable en formato MMDB (aproximadamente 60MB)
   - Disponible en repositorios públicos de GitHub
   - Descarga automatizable durante proceso de build
   - Almacenamiento local en directorio de recursos del backend
   - Precisión validada en contextos académicos [13]
   - Incluye: país, región, ciudad, coordenadas, timezone, ASN, ISP
4. **ipapi.co (API Complementaria):**
   - API REST pública para información ASN e ISP actualizada
   - Endpoint: https://ipapi.co/[ip]/json
   - Complementa datos de GeoLite2 con información de red actual
   - Rate limit: 1000 requests/día en modo gratuito
   - Proporciona: ASN, ISP, organización, tipo de red

7.3.2. Técnicas de Recolección

Integración Automatizada:

- **Process Execution:** Ejecución de Nmap desde el lenguaje de backend mediante APIs de procesos
- **XML Parsing:** Procesamiento de output XML de Nmap con librerías estándar
- **REST APIs:** Consultas programáticas a NVD API para obtener CVEs
- **File Processing:** Lectura local de bases de datos MMDB de GeoLite2
- **Caching:** Almacenamiento temporal para optimizar rendimiento

Validación de Datos:

- Verificación de formato y consistencia de datos
- Filtrado de información irrelevante o duplicada
- Manejo de errores y datos faltantes
- Logging de actividades para auditoría

7.4. Arquitectura del Sistema

7.4.1. Componentes Principales

Backend: Se propone una arquitectura de servicios modulares donde cada componente tiene una responsabilidad específica:

- **Controlador REST:** Endpoint principal para recibir solicitudes de análisis
- **Servicio Orquestador:** Coordina la ejecución del flujo completo de análisis
- **Servicio de DNS:** Resolución de dominios a direcciones IP
- **Servicio de Geolocalización:** Consulta a base de datos GeoLite2
- **Servicio de Escaneo:** Ejecución de Nmap y parsing de resultados
- **Servicio de Vulnerabilidades:** Consultas a NVD API
- **Servicio de ASN:** Consultas a ipapi.co para información de red
- **Servicio de Scoring:** Cálculo de puntuación de seguridad
- **Servicio de Recomendaciones:** Generación de sugerencias basadas en hallazgos

Frontend: Se propone una aplicación de página única (SPA) con las siguientes vistas principales:

- **Vista de Inicio:** Página principal con campo de búsqueda
- **Vista de Análisis:** Visualización completa de resultados del análisis
- **Vista de Comparación:** Comparación lado a lado de dos análisis
- **Vista de Historial:** Registro de análisis anteriores con estadísticas
- **Sistema de Cache:** Almacenamiento local para persistencia de datos
- **Componentes de Mapas:** Visualización geográfica interactiva

7.4.2. Flujo de Datos

1. **Entrada:** Usuario ingresa dirección IP o dominio a analizar
2. **Validación:** Sistema verifica formato (IPv4, IPv6, o dominio)
3. **Resolución DNS:** Si es dominio, resuelve a dirección IP mediante APIs estándar del lenguaje
4. **Verificación de Cache:** Frontend consulta LocalForage para resultados previos
5. **Procesamiento Paralelo en Backend:**
   - Geolocalización via GeoLite2 (consulta local MMDB, <1ms)
   - Consulta ASN/ISP via ipapi.co (complementa datos de red)
   - Escaneo de servicios via Nmap (TCP connect scan -sT, 1-60s)
6. **Análisis de Vulnerabilidades:**
   - Construcción de CPEs desde servicios detectados
   - Consulta NVD API por cada CPE
   - Agregación de CVEs con metadatos (CVSS, severidad, referencias)
7. **Scoring de Seguridad:** Cálculo mediante sistema de penalizaciones desde un score base de 100, considerando:
   - Penalizaciones por puertos de alto riesgo (Telnet, FTP, RDP, VNC, SMB)
   - Penalizaciones por servicios administrativos expuestos (bases de datos, SSH)
   - Penalizaciones por vulnerabilidades según severidad CVSS (CRITICAL, HIGH, MEDIUM, LOW)
   - Penalizaciones por superficie de ataque excesiva
   - Clasificación final: LOW ($\geq$80), MEDIUM ($\geq$60), HIGH (<60)
8. **Generación de Recomendaciones:** Sistema basado en reglas para sugerir mitigaciones según vulnerabilidades detectadas, protocolos inseguros y servicios expuestos.
9. **Presentación Frontend:**
   - Visualización unificada en interfaz web
   - Almacenamiento en cache local
   - Registro en historial de análisis

7.5. Herramientas y Tecnologías

7.5.1. Stack de Desarrollo Propuesto

Backend:

- Lenguaje: Java (versión LTS) por robustez y ecosistema maduro
- Framework: Quarkus, optimizado para aplicaciones cloud-native
- Arquitectura: API REST stateless para facilitar despliegue y escalabilidad
- Build: Gradle con soporte para contenedores Docker

Frontend:

- Framework: Vue.js 3 con Composition API
- Lenguaje: TypeScript para tipado estático
- Build Tool: Vite para desarrollo rápido
- Cache: LocalForage (IndexedDB) para persistencia local
- Visualizaciones: Chart.js para gráficos, Leaflet.js para mapas geográficos

Infraestructura:

- Contenedores: Docker para empaquetado y despliegue consistente
- Hosting: Plataformas cloud con tier gratuito
- CI/CD: Integración con repositorio Git para despliegue automático

7.5.2. Herramientas de Desarrollo

IDEs y Editores:

- IDEs apropiados para el lenguaje de backend seleccionado
- Editor de código para el desarrollo frontend
- Herramientas de testing de APIs (Postman o similares)

Control de Versiones:

- Git con GitHub para repositorio
- Branching strategy simplificado
- Issues tracking para gestión de tareas

7.6. Validación y Testing

7.6.1. Estrategias de Prueba

Testing Funcional:

- Unit tests para componentes individuales
- Integration tests para APIs
- End-to-end tests para flujos completos
- Manual testing de interfaz de usuario

Testing de Rendimiento:

- Pruebas de carga con múltiples consultas simultáneas
- Medición de tiempos de respuesta
- Evaluación de uso de memoria y CPU
- Testing de tiempos de ejecución de Nmap

7.6.2. Validación de Resultados

Comparación con Herramientas Existentes:

- Verificación de geolocalización contra servicios conocidos
- Comparación de detección de servicios con Nmap directo
- Validación de CVEs contra base de datos NVD oficial
- Evaluación de precisión y cobertura de datos

Testing de Usuario:

- Pruebas de usabilidad con estudiantes y profesores
- Evaluación de intuitividad de la interfaz
- Recolección de feedback para mejoras

7.7. Consideraciones Éticas y Técnicas

7.7.1. Uso Responsable de Datos

- Uso exclusivo de datos públicos y gratuitos (GeoLite2, NVD, ipapi.co)
- TCP connect scan en lugar de SYN scan (menos intrusivo, no requiere privilegios de root)
- Arquitectura stateless sin almacenamiento persistente en backend
- Historial de análisis con limpieza automática de datos antiguos
- Respeto a rate limits de APIs externas

7.7.2. Limitaciones Técnicas Reconocidas

- **Escaneo Nmap:** TCP connect scan es más lento que SYN scan, pero compatible con entornos cloud que bloquean raw sockets
- **Tiempo de ejecución:** Los escaneos pueden variar significativamente según la red objetivo
- **GeoLite2:** La base de datos requiere actualizaciones periódicas
- **NVD API:** Rate limiting en modo gratuito puede afectar análisis masivos
- Dependencia de disponibilidad de APIs externas
- Precisión de geolocalización variable según región geográfica
- Capacidad de procesamiento limitada por planes gratuitos de hosting
- Sin persistencia de datos históricos en backend (solo cache en frontend)

8. Cronograma de Trabajo

8.1. Duración y Alcance del Proyecto

Período Total: 8 semanas (del 10 de octubre al 1 de diciembre de 2025)
Equipo: 2 estudiantes
Modalidad: Desarrollo ágil con entregas incrementales

8.2. Metodología de Planificación

El cronograma se estructura utilizando sprints cortos de 1 semana para permitir entregas rápidas y ajustes continuos. Cada sprint incluye desarrollo, testing básico y documentación del progreso. Se consideran las limitaciones de tiempo académico y la curva de aprendizaje para las tecnologías seleccionadas.

8.3. Estructura de Desglose del Trabajo

8.3.1. Semana 1: Configuración e Investigación Inicial (10-17 Octubre)

Objetivos: Establecer ambiente de desarrollo y comprender fuentes de datos
Actividades principales:

- Configuración de entorno de desarrollo (Git, IDEs)
- Investigación de Nmap y formato de output XML
- Investigación de NVD API y formato de respuestas
- Descarga e instalación de GeoLite2
- Investigación de ipapi.co para información de ASN/ISP
- Creación de prototipos de escaneo con Nmap
- Definición de arquitectura inicial del sistema

8.3.2. Semana 2: Backend Base (17-24 Octubre)

Objetivos: Implementar estructura básica del backend con Quarkus
Actividades principales:

- Configuración inicial del proyecto Quarkus
- Implementación de API REST básica
- Integración con base de datos GeoLite2
- Desarrollo de servicios de geolocalización
- Testing unitario de componentes básicos

8.3.3. Semana 3: Integración Nmap, NVD e ipapi.co (24-31 Octubre)

Objetivos: Conectar escaneo de servicios con análisis de vulnerabilidades e información de ASN

Actividades principales:

- Implementación de servicio de ejecución de Nmap mediante APIs de procesos del lenguaje
- Desarrollo de parser XML para output de Nmap
- Implementación de cliente REST para NVD API
- Implementación de cliente REST para ipapi.co (ASN/ISP)
- Desarrollo de lógica de correlación CPE para búsqueda de CVEs
- Integración con servicios existentes de geolocalización
- Implementación de cache básico para optimizar consultas

8.3.4. Semana 4: Frontend Inicial (31 Octubre - 7 Noviembre)

Objetivos: Desarrollar interfaz básica con Vue.js

Actividades principales:

- Configuración del proyecto Vue.js
- Desarrollo de componentes básicos (formularios, tablas)
- Implementación de cliente HTTP para APIs
- Desarrollo de vistas principales (búsqueda, resultados)
- Integración con mapas usando Leaflet

8.3.5. Semana 5: Funcionalidades Avanzadas (7-14 Noviembre)

Objetivos: Mejorar funcionalidades y experiencia de usuario

Actividades principales:

- Implementación de comparador de IPs lado a lado
- Desarrollo de funcionalidad de exportación (JSON, PDF)
- Mejora de visualizaciones (gráficos estadísticos, mapas interactivos)

- Implementación de historial de búsquedas con estadísticas
- Optimización de rendimiento frontend

8.3.6. Semana 6: Testing y Optimización (14-21 Noviembre)

Objetivos: Validar funcionalidad y optimizar rendimiento
Actividades principales:

- Testing funcional completo de la aplicación
- Pruebas de rendimiento con múltiples consultas
- Testing de usabilidad con usuarios del entorno académico
- Optimización de consultas y cache
- Corrección de bugs identificados

8.3.7. Semana 7: Comparación y Validación (21-28 Noviembre)

Objetivos: Validar resultados contra herramientas existentes
Actividades principales:

- Comparación de resultados con Nmap directo y NVD manual
- Análisis de precisión de geolocalización
- Evaluación de cobertura de vulnerabilidades detectadas
- Validación de scoring de seguridad
- Documentación de limitaciones encontradas
- Preparación de datos para presentación final

8.3.8. Semana 8: Documentación y Entrega (28 Noviembre - 1 Diciembre)

Objetivos: Completar documentación y preparar entrega final
Actividades principales:

- Completar documentación técnica del código
- Elaborar manual de usuario
- Preparar guía de instalación y configuración

- Creación de presentación final
- Deployment en ambiente de producción (si es posible)

8.4. Cronograma Visual

Actividad	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**
Configuración inicial	X							
Backend desarrollo		X	X					
Frontend desarrollo				X	X			
Testing y optimización						X		
Validación y comparación							X	
Documentación final								X

Cuadro 1: Cronograma Semanal del Proyecto

Riesgo	**Prob.**	**Impacto**	**Mitigación**
Tiempo de ejecución de Nmap	Media	Alto	Timeouts configurables, scope limitado
Problemas de rendimiento	Alta	Medio	Testing temprano, optimización
Complejidad de Vue.js	Media	Medio	Documentación, ejemplos simples
Rate limiting de NVD API	Baja	Medio	Cache agresivo, búsquedas optimizadas
Retrasos por carga académica	Alta	Medio	Planificación flexible, buffer time

Cuadro 2: Matriz de Riesgos del Proyecto

8.5. Recursos y Herramientas

8.5.1. Equipo de Trabajo

- **Estudiante 1:** Enfoque en backend y integración de datos
- **Estudiante 2:** Enfoque en frontend y experiencia de usuario
- **Colaboración:** Testing, documentación y validación conjunta

8.5.2. Infraestructura Tecnológica

- **Desarrollo:** Computadoras personales con IDEs gratuitos
- **Repositorio:** GitHub (cuenta gratuita)
- **Comunicación:** Discord, WhatsApp, reuniones presenciales
- **Documentación:** Google Docs, LaTeX para documentos formales

8.5.3. Herramientas de Software

- **Backend:** Java con Quarkus, arquitectura REST stateless
- **Escaneo y Análisis:** Nmap, NVD API, ipapi.co
- **Frontend:** Vue.js con TypeScript, librerías de mapas, cache local
- **Testing:** Framework de testing del lenguaje, herramientas de testing de APIs
- **Deployment:** Docker, plataformas cloud con tier gratuito

Referencias

[1] MaxMind, Inc. (2025). *GeoLite2 Free Geolocation Data*. [Online]. Disponible en: https://www.maxmind.com/en/geolite2-developer-portal

[2] Lyon, G. F. (2025). *Nmap: Network Mapper - Free Security Scanner*. [Online]. Disponible en: https://nmap.org/

[3] National Institute of Standards and Technology. (2025). *National Vulnerability Database*. [Online]. Disponible en: https://nvd.nist.gov/

[4] Red Hat, Inc. (2025). *Quarkus: Supersonic Subatomic Java*. [Online]. Disponible en: https://quarkus.io/

[5] Evan You and The Vue Team. (2025). *Vue.js: The Progressive JavaScript Framework*. [Online]. Disponible en: https://vuejs.org/

[6] ipapi.co. (2025). *IP Geolocation API*. [Online]. Disponible en: https://ipapi.co/

[7] Lyon, G. F. (2009). *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure.Com LLC.

[8] Alghamdi, F. (2024). "A Comprehensive Investigation of Reconnaissance Threats in Cybersecurity," *International Journal of Advanced Research in Management and Social Sciences*, vol. 13, no. 11.

[9] Ferrag, M. A. et al. (2019). "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Network and Computer Applications*, vol. 50, pp. 41–65.

[10] Lin, P.-C. et al. (2023). "Correlation of cyber threat intelligence with sightings for intelligence assessment and augmentation," *Computer Networks*, vol. 229, art. 109751.

[11] Durumeric, Z., Wustrow, E., and Halderman, J. A. (2015). "ZMap: Fast Internet-wide Scanning and Its Security Applications," in *Proceedings of the 22nd USENIX Security Symposium*, Washington, DC, USA.

[12] Padmanabhan, V. N. and Subramanian, L. (2001). "An investigation of geographic mapping techniques for internet hosts," in *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, San Diego, CA, USA.

[13] Schopman, M. (2021). "Validating the accuracy of the MaxMind GeoLite2 City database," Bachelor's thesis, Radboud University.

[14] Pittman, J. M. (2023). "A Comparative Analysis of Port Scanning Tool Efficacy," arXiv preprint arXiv:2303.11282.