Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

ÁREA DE LENGUAJES Y SISTEMAS INFORMÁTICOS

Integración de una solución software ITSM con bases de datos de vulnerabilidades para la mejora de la ciberseguridad de las infraestructuras y sistemas TI

D. César del Río Fernández
TUTOR ACADÉMICO: D. José García Fanjul
TUTOR DE LA EMPRESA: D. Alejandro Castro Valdés

FECHA: Junio 2022

ÍNDICE

TABLA DE ILUSTRACIONES

TABLAS

1. Introducción.

En este documento se describirá la memoria desarrollada para el trabajo fin de grado : *Integración de una solución software ITSM con bases de datos de vulnerabilidades para la mejora de la ciberseguridad de las infraestructuras y sistemas TI*.

Este proyecto se ha llevado a cabo en colaboración con el grupo empresarial Espiral MS, en la evolución de su producto ProactivaNET, cuyo departamento de desarrollo se encuentra en Gijón, con el que se ha trabajado conjuntamente.

1.1.- ADAPTACIÓN DE LAS NORMATIVAS EN LA DOCUMENTACIÓN

Para la elaboración de este documento se han seguido las pautas y directrices establecidas en la siguiente normativa:

- Reglamento de Trabajos Fin de Grado (EPI Gijón) - TFG de Productos Software
  Este documento define como se tramita la propuesta, asignación, defensa y calificación del trabajo y que documentación se debe aportar, así como el formato y los métodos para generarla.

A continuación se describirá que apartados y secciones de esta memoria corresponden con los documentos y contenidos establecidos en la normativa establecida para el desarrollo de trabajos fin de grado.

Reglamento EPI	Apartados del documento
1. Memoria	
1.1. Objetivo y alcance del trabajo	5. Objetivo y alcance del trabajo. 22
1.2. Antecedentes	6. Antecedentes. 24
1.3. Análisis de alternativas y justificación de las decisiones adoptadas.	7. Análisis de alternativas 25
1.4. Programación temporal de los trabajos a realizar	11. Planificación..... 70
1.5. Conclusiones	16. Conclusión 82
1.6. Bibliografía	17. Bibliografía. 83
2. Presupuesto del desarrollo efectuado	12. Presupuesto 76
3. Documentos técnicos.	
3.1. Requisitos de usuario o pliego de condiciones.	2. Descripción general..... 10 3. Lista de usuarios participantes ..... 18 4. Catálogo de Requisitos. ..... 20 7. Análisis de alternativas 25
3.2. Análisis de requisitos del sistema	8. Análisis de requisitos del sistema..... 28
3.3. Diseño (de arquitectura y de detalle).	9. Diseño del sistema. 37
3.4. Pruebas (diseño y resultados de ejecución).	10. Plan de pruebas. 53
4. Aplicación instalable	14. Aplicación instalable. 80
5. Manual de usuario	15. Manual de configuración..... 81

Tabla 1-1.- Correspondencia Normativa

2. Descripción general.

2.1.- GESTIÓN DE TI

En cualquier empresa hoy en día, la gestión de las tecnologías de información es indispensable. La gestión de TI comprende la administración tanto de los servicios TI como de las infraestructuras que lo componen, con el fin de proporcionar valor al cliente a partir de una mejora de la calidad del servicio.

Cada organización llevará a cabo la gestión de TI siguiendo un marco de referencia y normas distinto, en función del ámbito propio de la empresa, del ámbito específico de TI o de las funciones y tecnologías concretas que se desarrollan.

2.1.1.- ITIL

Dentro de todos los marcos posibles, ITIL se enfoca en la especificación de buenas prácticas que permiten conseguir una mayor calidad en los servicios TI. Para ello presenta una serie de directrices y procesos detallados que permiten a una organización conseguir aportar valor a los servicios que ofrece.

ITIL se ha ido adaptando a las necesidades que surgen a lo largo del tiempo, debido a los avances en las tecnologías que permiten la gestión de TI, cambiando el enfoque de sus buenas prácticas. En su última versión, ITIL4, amplia su perspectiva, centrándose más en como aportar valor a través de los procesos y métricas que define, sin dejar de lado todas las prácticas aportadas en la versión anterior ITILv3, enfocada al ciclo de vida de los servicios y en qué actividades hay que hacer para aportar valor. Este cambio de rumbo viene dado por la incorporación de nuevas prácticas emergentes en el desarrollo y gestión de TI, como son las técnicas Agile o DevOps, que aportan mayor flexibilidad y adaptación.

Con este nuevo punto de vista, toma importancia la forma en la que se realizan las prácticas y procesos, acentuando la relevancia del cliente, que al final, es aquel que percibe y determina el valor de un servicio.

ITIL4 propone un modelo de 4 dimensiones que dividen la gestión de TI en:

- Organizaciones y personas: los valores y cultura de una organización para alcanzar sus objetivos, junto a la formación, aptitud y competencia de las personas que la forman.
- Información y tecnología: información, conocimiento y saber de una organización, como las tecnologías que permiten la gestión de TI .
- Socios y Proveedores: relaciones con otras empresas involucradas en el diseño, implementación, entrega, soporte y mejora continua de los servicios.
- Flujos de valor y procesos: estructura y organización de todos los componentes para la creación de valor a través de productos y servicio

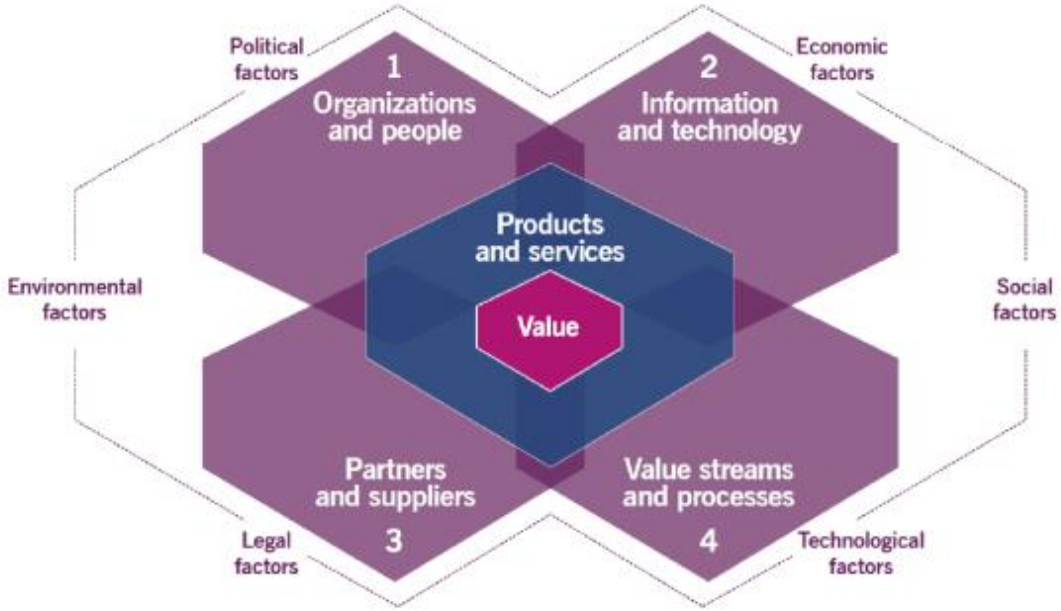


Figura 2.1.- Modelo de 4 dimensiones de la gestión de TI

En base a este modelo, ITIL4 organiza las buenas prácticas para gestionar todas las dimensiones de una organización en tres grupos:

- Prácticas de gestión generales: orientadas a la administración de las dimensiones “Organizaciones y personas” y “Socios y Proveedores”.
- Prácticas de gestión de servicios: orientadas a la administración de las dimensiones “Información y tecnologías” y de los productos y servicios.
- Prácticas de gestión técnica: orientadas a la administración de la dimensión “Flujos de valor y procesos”

2.1.2.- ProactivaNET

ProactivaNET, dentro de las líneas de negocio desarrollados por Espiral MS, es un producto que permite a los encargados de TI de una empresa gestionar activos y servicios de manera sencilla, siempre dentro del marco de las prácticas ITIL.

A partir de las buenas prácticas que propone ITIL4, ProactivaNET se enfoca en el ámbito de la información y de las tecnologías, dentro de las prácticas de gestión de servicios. Concretamente, este producto se centra en la gestión de activos IT, también denominada ITAM (Information Technology Asset Management), con el objetivo de aportar valor a sus servicios a partir de la co-creación de valor, uno de los términos que incluye ITIL4.

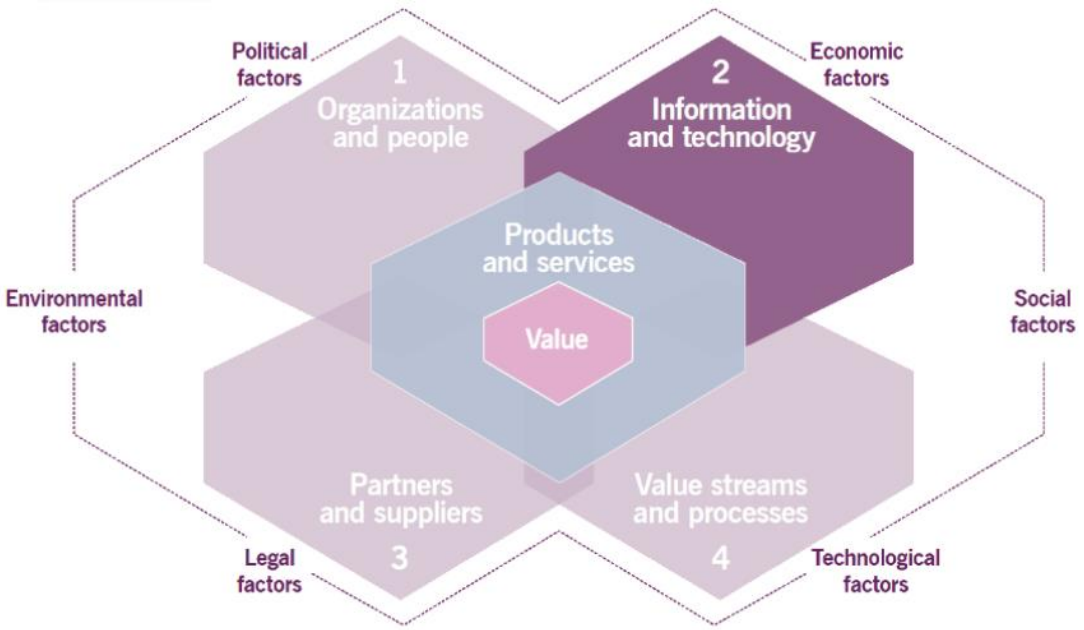


Figura 2.2.- Modelo de 4 dimensiones de gestión de TI de ProactivaNET

Este término define un nuevo enfoque de la relación entre el proveedor de servicios y el cliente. El proveedor no crea valor únicamente para sus clientes, sino que colabora con ellos para aportar valor a ambas partes, al cliente en forma de calidad en los servicios que obtiene y al proveedor en forma de valor material obtenido por suministrar el servicio o en forma de nuevas capacidades o aptitudes que permitan ofrecer servicios adicionales.

ProactivaNET gracias a sus servicios de gestión de activos permite:

- Reducción de costes: en organizaciones de gran tamaño se estima que existe un 30% de componentes informáticos que no están controlados. ITAM permite monitorizar al completo todos los componentes de una organización, reduciendo costes de mantenimiento de componentes obsoletos o en desuso que no aportan valor. De la misma forma, se crean métricas e informes sobre cada activo, determinando quien, cuando, donde, como o cuanto tiempo se utiliza. La reducción de costes gracias a ITAM aumenta el rendimiento y el ciclo de vida de los servicios que ofrece una organización.
- Reducción de riesgos: tener un control detallado de todos los activos que tiene una organización permite conocer el estado de seguridad en el que se encuentra cada uno de ellos, minimizando al máximo el riesgo de ataques que, podrían suceder, en caso de tener dispositivos sin monitorizar con versiones de software desactualizadas vulnerables.
- Mejorar resultados: gracias el incremento del rendimiento de los servicios y a la reducción de costes y de riesgos, se aumenta tanto la calidad de los servicios como la satisfacción del usuario o el ROI, retorno de inversión.

2.2.- SEGURIDAD

Una parte de las ventajas de realizar una correcta gestión de activos siguiendo las métricas y buenas prácticas que proporciona ITIL4, a través del uso de programas como ProactivaNET, es la reducción de riesgos, aspecto de vital importancia en el ámbito de la seguridad. Además, frecuentemente las organizaciones complementan sus estudios sobre seguridad, desarrollando un análisis DAFO en el momento de evaluar la situación estratégica de la organización, determinado aspectos internos como sus fortalezas y debilidades y aspectos externos como oportunidades y amenazas.

Estos estudios permiten encontrar posibles amenazas y debilidades que, en caso de pasar inadvertidas, pueden implicar graves consecuencias que afecten a la calidad de los servicios, en cuanto a la disponibilidad o a la integridad de los datos que se manejan. En este escenario, el conocimiento respecto a la seguridad, en el marco de los sistemas informáticos, aporta gran valor a los servicios de una empresa.

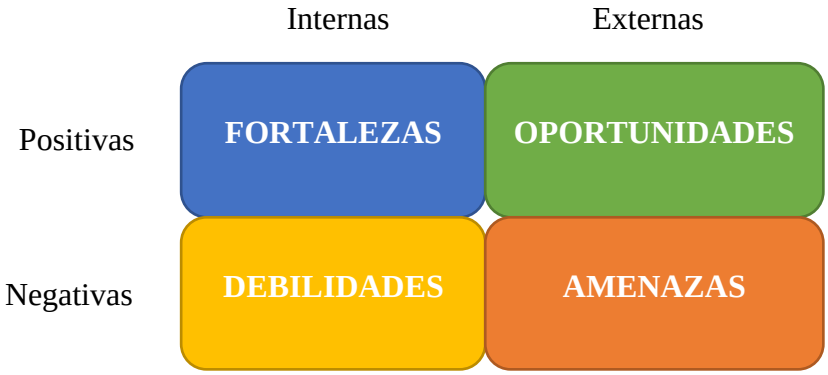


Figura 2.3.- Análisis DAFO

Dentro de este contexto, ProactivaNET por medio de este proyecto, pretende mejorar y aumentar en gran medida la información que proporciona a sus clientes respecto sus activos en el campo de la seguridad. Para ello se propone una integración de NVD, una base de datos que proporciona el NIST, en la base de datos de ProactivaNET.

2.2.1.- NIST y NVD

El NIST (National Institute of Standars and Tecnology) es una institución que forma parte del Departamento de Comercio de los Estados Unidos, cuya labor es fomentar la innovación y la competitividad industrial del país, por medio del avance de la ciencia y de la tecnología de medición. Esta institución proporciona un conjunto de estándares y mediciones, en ámbitos como la alimentación o la seguridad, con el objetivo de incentivar el desarrollo de nuevas tecnologías que se fundamenten en dichas reglas.

Entre muchas de sus actividades, el NIST desarrolla NVD (National Vulnerability Database), una base de datos que almacena y registra las vulnerabilidades que se detectan en cualquier elemento informático, ya sea hardware o software. Estos datos permiten la gestión de dichas vulnerabilidades y la medición y cumplimiento de la seguridad.

NVD determina tres elementos de seguridad, a partir de los cuales crea toda la estructura de información de seguridad, CVE, CPE y CWE. CVE, el cual representa una vulnerabilidad, CPE, el cual representa a un determinado activo y CWE, el cual representa una debilidad.

En el siguiente ejemplo se pueden observar las relaciones entre estos tres elementos. El CPE que representa al sistema operativo Windows 10, 'cpe:2.3:o:microsoft:windows_10:-:*:*:*:*:*:*:*', tiene asociados varios CVE, entre los que se encuentra el 'CVE-2021-43217', vulnerabilidad de ejecución remota de código del sistema de cifrado de archivos (EFS). Esta vulnerabilidad conlleva una serie de debilidades en los activos en los que se detecta, como la vulnerabilidad 'CWE-94', la cual implica un control inadecuado de la generación de código, posibilitando inyección de código.

2.2.1.1.- CVE (Common Vulnerabilities and Exposures)

CVE es una lista de información sobre vulnerabilidades de seguridad. Cada entrada de esta lista representa una vulnerabilidad que ha sido detectada, estudiada y posteriormente registrada.

Cada vulnerabilidad de la lista está asociada a un identificador único que toma el mismo nombre, CVE. Dicho identificador tiene el siguiente formato, CVE-YYYY-NNNN, donde YYYY representa el año en el que se detectó y NNNN el número de vulnerabilidad. Este formato es universal, es decir, todas las vulnerabilidades están identificadas según dicho patrón.

2.2.1.2.- CPE (Common Platform Enumeration)

CPE es un esquema de nombrado estructurado, con el cual se construye un identificador que representa, inequívocamente, de la misma forma, a sistemas, software y paquetes. Dicho esquema se basa en la sintaxis genérica utilizada para los identificadores uniformes de recursos (URI).

Un CPE puede ser de varios tipos, hardware, aplicación o software. El tipo que tendrá un CPE cambia la forma en la que se construye dicho identificador, siguiendo el siguiente formato.

"cpe23Uri": "cpe:2.3:v1:v2:v3:v4:v5:v6:v7:v8:v9:v10:v11"

Valor	Significado	Definición
V1	Parte	Puede ser cualquiera de los siguientes valores: • "a" → para aplicaciones • "o" → para sistemas operativos • "h" → para hardware
V2	Vendedor	Especifica la identidad de la persona u organización que creó o fabricó el producto.
V3	Producto	Describe o identifica el nombre más común para dicho producto.
V4	Versión	Cadena alfanumérica que especifica la versión del producto.
V5	Actualización	Cadena alfanumérica que especifica la identificación de la actualización del producto, service pack o big update.
V6	~~Edición~~	
V7	Lenguaje	Lenguaje soportado por la interfaz de usuario del producto, siguiendo el RFC5646
V8	Edición de Software	Descripción de como un producto está adaptado a un mercado específico o al usuario final
V9	Software objetivo	Describe el software en el que se ejecuta el producto.
V10	Hardware objetivo	Describe el conjunto de instrucciones en el que el producto opera (por ejemplo, x86).
V11	Otros	Otros datos que pueden ser relevantes, o identificar información que no encaja con ninguno de los apartados anteriores.

Figura 2.4.- Formato CPE

La construcción de un CPE, en función del tipo del activo, conlleva un proceso complejo, ya que cada campo que lo forma requiere una información concreta que puede resultar difícil de obtener.

Debido a esta complejidad en cuanto a la formación del CPE, en este proyecto solo se abarcará la integración de la información relativa a CPEs de tipo sistema operativo, dado que las dos marcas de sistemas operativos más utilizadas, Linux y Windows, permiten obtener de manera sencilla toda la información necesaria para crear el CPE. Los sistemas operativos Linux, proporcionan directamente dicho CPE ya formado, mientras que Windows facilita el acceso a la información necesaria para la creación del CPE.

En el archivo que se adjuntará a esta memoria, denominado "cpe_windows.xlsx" ,se han volcado todos los distintos CPEs para sistemas operativos Windows, donde se puede observar que, en función de la versión y del sistema que soporte, la creación del identificador varía.

2.2.1.3.- CWE (Common Weakness Enumeration)

Lista de tipos de debilidades software y hardware, que sirve de referencia para herramientas de seguridad y como punto de partida para la definición, prevención y atenuación de las debilidades.

Cada debilidad de la lista está asociada a un identificador único que toma el mismo nombre, CWE. Dicho identificador tiene el siguiente formato, CWE-Id, siendo id un identificador que representa el número de la debilidad. Este formato es universal, es decir, todas las vulnerabilidades están identificadas según dicho patrón.

La diferencia entre CVE y CWE, vulnerabilidad y debilidad, es que mientras que el CVE determina las consecuencias ante un determinado ataque de seguridad, el CWE determina las causas de ese ataque.

2.3.- SEGURIDAD Y GESTIÓN DE ACTIVOS

Incluyendo la información proporcionada por NVD para cada elemento monitorizado a través de la gestión de activos e inventariado de ProactivaNET, se proporciona al cliente una gran cantidad de información de seguridad para cada activo de su organización.

Esta información de las vulnerabilidades y debilidades de un determinado activo permite tener una conocimiento exhaustivo y profundo del estado de seguridad en el que se encuentra, siendo un factor diferencial en la gestión de servicios y aportando un gran valor al cliente, el cual es conocedor del estado de seguridad al completo de su organización.

3. Lista de usuarios participantes.

En este apartado se determinará y definirá la lista de Stakeholders que forman parte en el desarrollo del proyecto. Se denomina Stakeholder o partes interesadas a un individuo u organización que muestra interés por un producto, proyecto o servicio debido a que sus actividades, objetivos, recursos o entregables pueden satisfacer sus necesidades o expectativas.

Los Stakeholder pueden ser internos, como personal, grupos o equipos de la empresa que desarrolla el proyecto, o externos, como clientes, usuarios o proveedores.

- Internos
    - Equipo de desarrollo web de ProactivaNET : grupo de trabajo que tiene como función el diseño, ejecución y mantenimiento del apartado web del producto ProactivaNET. A partir de la información recogida de NVD, este equipo será el encargado de desarrollar un módulo en el cual se muestre al usuario final dicha información.

    - Equipo de Sistemas Anexos de ProactivaNET : grupo de trabajo que tiene como función el desarrollo de procesos a bajo nivel que permiten la realización de funciones centrales de ProactivaNET como el inventariado de activos. Encargados de supervisar la adhesión del proceso de integración de datos de NVD con la base de datos de ProactivaNET en su propia arquitectura.

    - Equipo de Soporte de ProactivaNET : grupo de trabajo de primera línea, que da asistencia y gestiona las dudas, incidencias o problemas que comunican los usuarios de ProactivaNET. La información que aporta NVD puede ser útil para resolver peticiones de los clientes, por lo que deberá tener una estructura fácil de comprender sin amplios conocimientos de seguridad.

  - Equipo de Administración y Mantenimiento de Sistemas : grupo de trabajo que gestiona la infraestructura informática de ProactivaNET. De la misma manera que el equipo de soporte, este equipo puede beneficiarse de la información que aporta NVD sobre sus propios dispositivos, pudiendo tomar decisiones con el objetivo de mejorar la seguridad de los sistemas que gestionan.

- Externos
  - Clientes de ProactivaNET : el personal encargado de gestionar la seguridad en las organizaciones que obtengan los servicios que ofrece ProactivaNET accederán a la información que se integrará desde NVD para comprender mejor el estado actual de sus dispositivos.

4. Catálogo de Requisitos.

Tras varias reuniones con los Stakeholders del proyecto se han definido los siguientes requisitos de usuario, en función de sus condiciones y necesidades.

Se determinarán dos tipos de requisitos en función de su naturaleza, funcionales y no funcionales. Los requisitos funcionales son aquellos que definen como se tiene que comportar el programa, qué tiene que hacer o realizar. En cambio, los no funcionales son aquellos que determinan características como el rendimiento, la seguridad o la adaptabilidad.

- Funcionales

R.F.1.- Respecto a las funciones del software

R.F.1.1.- Selección de la fuente de información: permitir al usuario obtener información de varias fuentes, determinando al inicio del programa a cual se realizarán las peticiones.

R.F.1.2.- Proceso de descarga de datos: proceso encargado de la obtención, procesamiento y almacenamiento de los datos.

R.F.1.3.- Purgado periódico de la información obsoleta almacenada: eliminación de registros obsoletos que estén en la base de datos sin modificar durante un periodo de tiempo.

R.F.1.4.- Configuración de los periodos de tiempo para el purgado de registros obsoletos en base de datos.

R.F.1.5.- Selección de una determinada url para obtener información de uno o varios elementos concretos.

R.F.2.- Respecto a los datos manejados

R.F.2.1.- Procesamiento de los datos obtenidos en la respuesta de una petición, para filtrar y almacenar la información de los elementos de seguridad CVE (vulnerabilidades), CPE (activos) y CWE (debilidades).

- No funcionales

R.NF.1.- Rendimiento

R.NF.1.1.- El proceso de obtención, procesamiento y almacenamiento de la información debe de afectar lo menos posible al rendimiento de los demás procesos con los que se integrará.

R.NF.2.- Continuidad de negocio

R.NF.2.1.- El proceso debe poder adaptarse a interrupciones debido a actualizaciones, lanzamientos de versiones o pausas de mantenimiento, pudiendo retomar su ejecución.

R.NF.3.- Estándares aplicables

R.NF.3.1.- Se deberán respetar los estándares de programación, de diseño de la arquitectura y la normativa en cuanto a la nomenclatura, definidos para el desarrollo del producto ProactivaNET.

5. Objetivo y alcance del trabajo.

5.1.- OBJETIVO

Teniendo en cuenta los requisitos establecidos en la fase de estudio de la viabilidad del sistema, el objetivo de este proyecto es el desarrollo de una solución software, que permita la integración de datos de seguridad de una fuente de información certificada, con el fin de aportar valor a los servicios proporcionados por el producto ProactivaNET.

De esta forma, ProactivaNET proveerá a sus usuarios de la información necesaria para conocer el estado detallado de la infraestructura informática de su organización, en el ámbito de la seguridad, por medio de sus procesos de inventariado y gestión de activos.

5.2.- ALCANCE

En este proyecto únicamente se abarcará la obtención y almacenamiento de la información que proporciona NVD, base de datos de vulnerabilidades desarrollada por el NIST, institución del gobierno de estados unidos que define estándares y mediciones en el ámbito de la seguridad, puesto que actualmente, proporciona una API a través de la cual se puede establecer de manera sencilla una conexión con su base de datos.

En consecuencia, se diseñará un programa que permita la introducción de una fuente de información como parámetro, teniendo en cuenta futuras ampliaciones con más fuentes de información certificadas, aunque en este proyecto solo se abarque la conexión vía API con NVD.

Para ello se llevará a cabo una ampliación de la base de datos principal, para poder almacenar los datos obtenidos a partir de NVD, teniendo en cuenta la nomenclatura y estándares establecidos por ProactivaNET.

Tras la integración de este modelo de dominio en ProactivaNET, se definirán dos algoritmos que abarquen todo el proceso de comunicación con la NVD, el cual comprende la obtención, procesamiento y almacenamiento de la información en la base de datos de ProactivaNET y el purgado de vulnerabilidades obsoletas o que no estén relacionadas con los activos o servicios del cliente. Estos procesos se diseñarán de manera que se tenga en cuenta el

rendimiento y eficiencia, tanto por separado como en su integración con los procesos principales de ProactivaNET, con el objetivo de minimizar el impacto en el producto global.

Dado que la construcción de un CPE en función del tipo del activo conlleva un proceso complejo, ya que cada campo que lo forma requiere una información concreta que puede resultar difícil de obtener, se integrará en la base de datos de ProactivaNET la información de las vulnerabilidades y datos de seguridad asociados exclusivamente a sistemas operativos, puesto que las dos marcas de sistemas operativos más utilizadas, Linux y Windows, permiten obtener de manera sencilla toda la información necesaria para crear dicho identificador. Por lo tanto, la integración de vulnerabilidades relacionadas con aplicaciones y hardware no se abarcará en este proyecto.

Por último, no se contemplará el diseño e implementación de una interfaz gráfica que permita la configuración de dicho programa, con el objetivo de enfocar todo el tiempo y los recursos disponibles al perfeccionamiento y desarrollo de los procesos principales de descarga de datos y purga de la información obsoleta. Por el mismo motivo, tampoco se abarcará la implementación de un módulo en la interfaz de ProactivaNET para visualizar la información obtenida.

6. Antecedentes.

Este proyecto tomará como referencia la investigación realizada previamente por José Carlos Díaz Ruénez, recogida en el documento: *Detección de vulnerabilidades en los activos inventariados en la herramienta Inventario de ProactivaNET - Integración entre ProactivaNET, NVD e INCIBE*, en el cual se realiza un estudio teórico de:

- El tipo y formato de los datos que proporciona la API de NVD.
- El modo en el que se realiza las llamadas a la API.
- Un modelo de dominio inicial sobre cómo integrar la información en la base de datos de ProactivaNET.

En este documento se refleja, en el apartado de diseño 9.1.3.- Estructura de la API de NVD, la información que se ha reutilizado de ese trabajo, relativa al uso de la API proporcionada por NVD para la obtención de los datos de seguridad.

7. Análisis de alternativas.

7.1.- ARQUITECTURA DEL PROGRAMA

NVD proporciona a través de la API más de 176.000 registros que se tendrán que procesar y cargar en la base de datos, aspecto que en función del diseño de la arquitectura puede afectar al rendimiento global del producto.

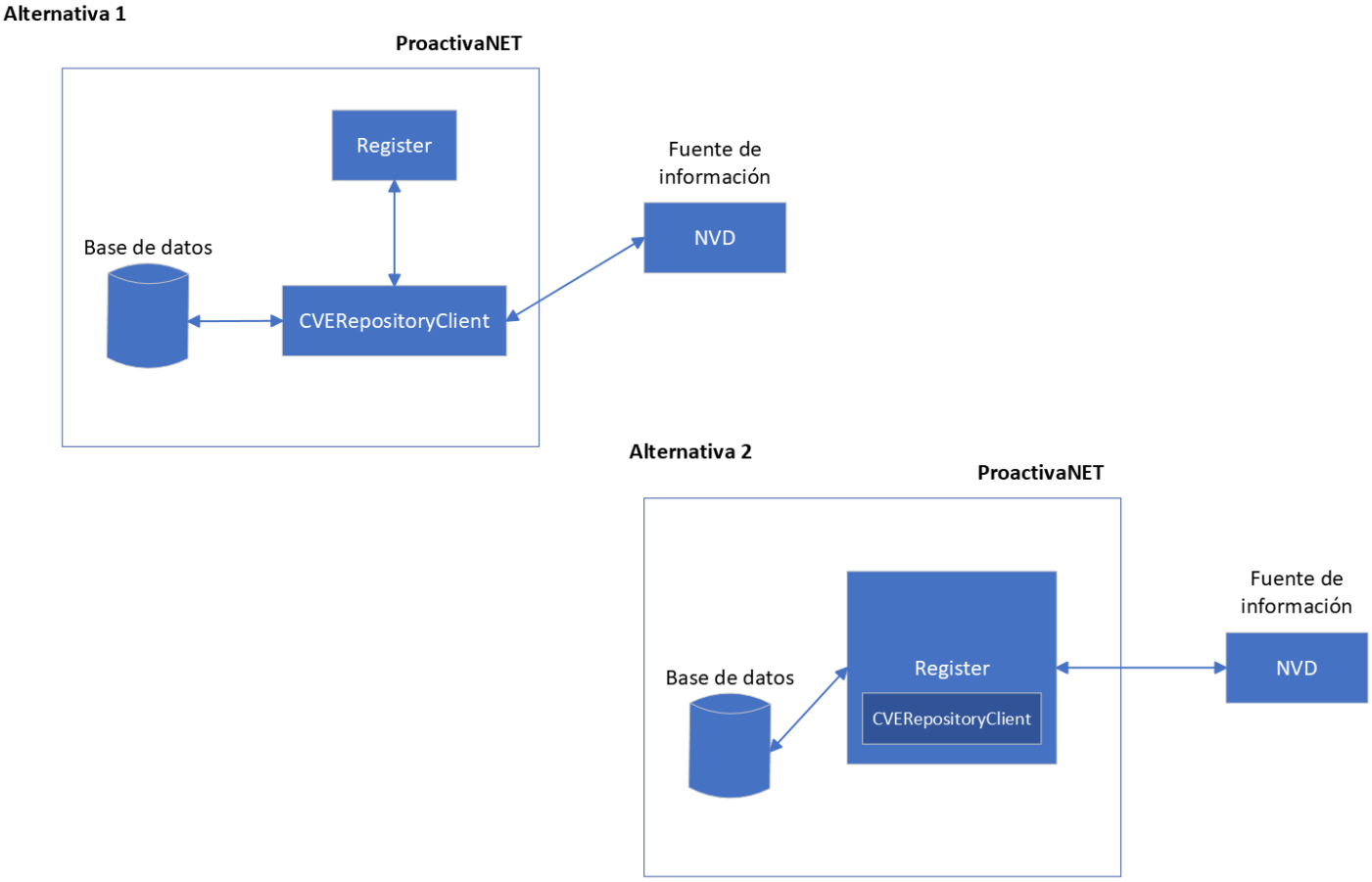


Figura 7.1.- Alternativas para el diseño de la arquitectura

- Alternativa 1: Externalización de los procesos de descarga y purga del *register*

Optando por la externalización de estos procesos, en primer lugar se consigue una distribución más eficiente de los recursos, liberando al *register*, proceso principal de la aplicación ProactivaNET, encargado de gestionar todas las tareas relacionadas con el inventariado y gestión de los activos, de la carga computacional derivada del procesamiento de la información de NVD, y, en segundo lugar, proporciona mayor flexibilidad a la hora de iniciar, parar o reanudar dichos procesos debido a actualizaciones en otros módulos de

ProactivaNET o liberaciones de nuevas versiones de la aplicación, que implican el bloqueo o reinicio de todos sus componentes.

- Alternativa 2: Integración de los procesos de descarga y purga en el *register*

En caso de integrar los procesos de descarga y purga de datos directamente en el *register*, podría llegar a sobrecargar dicho proceso, repercutiendo en todas las demás tareas de monitorización e inventariado de dispositivos, reduciendo en general la eficiencia de ProactivaNET debido a la gran cantidad de recursos que se tendrán que proveer para procesar esta cantidad de información.

- Alternativa seleccionada

Por tanto, el programa que llevará a cabo ambos procesos se ha diseñado de manera que sea independiente del *register*. Dicho programa se invocará desde el *register* como un ejecutable externo. El *register*, invocará tanto al proceso de descarga de datos como al proceso de purga diariamente, como una más de sus tareas de monitorización y gestión de activos.

7.2.- EFICIENCIA Y RENDIMIENTO DE LOS PROCESOS

Una vez solventado este aspecto en cuanto al reparto de los recursos para cada proceso, se deberá gestionar la tasa de conexiones que se generan con la base de datos.

La forma y estructura que se defina para establecer las conexiones con la base de datos para almacenar, modificar o eliminar registros afectará en gran medida al rendimiento del programa.

Las prestaciones generales se degradan por las dependencias que se generan con la base de datos debido al alto ratio de comunicaciones, provocando una mala distribución de la carga computacional. Por ello, el objetivo será limitar costes de creación y destrucción de comunicaciones, minimizando los retardos debidos a la interacción entre base de datos y procesos.

- Alternativa 1: Mayor número de conexiones de poco tamaño

En caso de que por cada registro que se obtenga de NVD, se establezca una conexión con la base de datos para realizar una determinada acción, se estará sobrecargando el ancho de banda disponible para todos los procesos de ProactivaNET, afectando gravemente al rendimiento general, de la misma forma que con la integración de los procesos en el *register*.

- Alternativa 2: Conexiones por bloques, menor número de conexiones de mayor tamaño

La técnica por bloques consiste en agrupar varias consultas o acciones sobre la base de datos y establecer una única conexión con varios comandos o acciones a realizar. De esta forma, a costa de aumentar los datos que se transfieren por cada conexión, se reducen el número de comunicaciones, reduciendo a su vez sus costes de creación y eliminación.

Para ello se utilizarán las estructuras de datos denominadas, DBInserter y DBUpdater, las cuales agrupan las inserciones y modificaciones sobre base de datos para llevar a cabo la transferencia de datos por bloques.

- Alternativa seleccionada

Por tanto, con el objetivo de minimizar el número de conexiones con la base de datos y así maximizar el rendimiento del programa, se optará por reducir la frecuencia de comunicaciones entre base de datos y procesos, aumentando el volumen de datos a transferir, estableciendo conexiones por bloques, en vez de individualmente por registro.

8. Análisis de requisitos del sistema.

A lo largo de este apartado se describirá detalladamente el sistema de información propuesto, a través del desarrollo y especificación de varios modelos, los cuales satisfacen las necesidades de las partes interesadas, determinadas en la fase de estudio de viabilidad del sistema. Estos modelos servirán de base para el posterior diseño e implementación del sistema.

8.1.- MODELO DE DOMINIO

Se ha definido un modelo de datos, el cual contempla todas las entidades, relaciones, atributos y reglas de negocio necesarias para satisfacer las necesidades y requisitos definidos previamente.

8.1.1.- Diccionario de datos

En este aparatado se describirán las entidades del modelo de dominio desarrollado.

- PanCPEs
    - Descripción: tabla que almacenará los datos relacionados con los CPEs asociados a una o varias vulnerabilidades. Estos CPEs además estarán asociados con el Sistema Operativo que representa, en caso de que esté inventariado en la base de datos de ProactivaNET.
    - Atributos:
        - id (Guid)
        - identifier (varchar) : valor del CPE.
            - Ej: “cpe:2.3:a:openssl:openssl:0.9.1c:*:*:*:*:*:*:*”
        - dateCreated (date) : fecha en la que se crea la entrada en la base de datos.
        - dateModifyToAssets (date) : fecha en la que el registro se vio modificado, bien por un cambio en algún atributo, o por la creación o eliminación de una relación con vulnerabilidades o sistemas operativos.

- PanCWEs
    - Descripción: tabla que almacenará los datos relacionados con las debilidades asociadas a una o varias vulnerabilidades.
    - Atributos:
        - id (Guid)
        - identifier (varchar) : valor del CWE.
            - Ej: “CWE-326”.
        - name (varchar) : nombre de la vulnerabilidad asociada al CWE.
            - Ej: “Inadequate Encryption Strength”
        - source (varchar) : fuente que determina la vulnerabilidad.
            - Ej: “MITRE”.
        - dateModifyToCVEs (date) : fecha en la que el registro se vio modificado, bien por un cambio en algún atributo, o por la creación o eliminación de una relación con vulnerabilidades.

- PanOs
    - Descripción: tabla existente en la base de datos de ProactivaNET que almacena la información sobre los sistemas operativos inventariados en la aplicación. Esta tabla será la conexión entre la base de datos actual de ProactivaNET y la ampliación desarrollada en este documento.
    - Atributos (se muestran únicamente los atributos relevantes para el desarrollo de la ampliación de la base de datos):
        - id (Guid)
        - panCPEs_id (Guid) : foreign key a panCPEs

- PanVulnerabilities
    - Descripción: tabla central del modelo de dominio. Almacena la información más relevante sobre vulnerabilidades.

- Atributos:
  - id (Guid)
  - identifier (varchar) : valor del CVE.
  - dateCreated (date) : fecha en la que se crea la entrada en la base de datos.
  - datePublished(date) : fecha en la que se publicó la vulnerabilidad.
  - dateLastUpdate (date) : fecha en la que se modificó el registro por última vez en la base de datos.
  - panVulnerabilitiesSources_id (Guid) : foreign key a panVulnerabilitiesSources
  - panVulnerabilitiesAssigners_id (Guid) : foreign key a panVulnerabilitiesAssigners
  - description (varchar) : cadena de texto que describe la vulnerabilidad

- PanVulnerabilitiesCPEs
  - Descripción: tabla creada para relacionar vulnerabilidades y CPEs.
  - Atributos:
    - id (Guid)
    - panVulnerabilities_id (Guid) : foreign key a panVulnerabilities
    - panCPEs_id (Guid) : foreign key a panCPEs
    - versionStartIncluding (varchar) : primera versión del activo con la vulnerabilidad
    - versionEndIncluding (varchar) : última versión del activo con la vulnerabilidad
    - versionEndExcluding (varchar) : última versión del activo sin la vulnerabilidad

- PanVulnerabilitiesCWEs
  - Descripción: tabla creada para relacionar vulnerabilidades y CWEs.
  - Atributos:
    - id (Guid)
    - panVulnerabilities_id (Guid) : foreign key a panVulnerabilities

- panCWEs_id (Guid) : foreign key a panCWEs

- PanVulnerabilitiesReferences
  - Descripción: tabla que almacenará información adicional relacionada con referencias asociadas a una vulnerabilidad.
  - Atributos:
    - id (Guid)
    - panVulnerabilities_id (Guid) : foreign key a panVulnerabilities
    - url (varchar) : enlace a referencias acerca de la vulnerabilidad
    - name (varchar) : enlace a referencias acerca del nombre de la vulnerabilidad
    - tags (varchar) : palabras clave que asociadas a la vulnerabilidad.
      - Ej: "Exploit","Third Party Advisory"

- PanVulnerabilitiesScore
  - Descripción: tabla que almacenará información adicional relacionada con las puntuaciones de severidad asociadas a una vulnerabilidad.
  - Atributos:
    - id (Guid)
    - panVulnerabilities_id (Guid) : foreign key a panVulnerabilities
    - version (int) : número de versión de la puntuación. Tomará los valores 2 y 3, para las versiones 2.0 y 3.0, respectivamente, dado que se calcula de forma distinta el valor de la severidad en cada versión.
    - severity (int) : nivel de severidad de la vulnerabilidad. Tomará los siguientes valores: {0:none,1:low,2:medium,3:high,4:critical}
    - score (float) : valor numérico que determina la severidad de la vulnerabilidad.
    - vector (varchar) : Lista de parámetros con la que se calcula la severidad de la vulnerabilidad.
      - Ej: “CVSS:X/AV:X/AC:X/Au:X/C:X/I:X/A:X”
        - CVSS es la versión que se utilizó para definir el vector
        - AV es accesVector

- AC es accessComplexity
- Au es authentication
- C es confidentialityImpact
- I es ubtegrityImpact
- A es availabilityImpact

- PanVulnerabilitiesSources
  - Descripción: tabla que almacenará información sobre las entidades de las cuales se obtiene información de seguridad relacionada con vulnerabilidades.
  - Atributos:
    - id (Guid)
    - name (varchar) : nombre de la entidad de la que se obtuvo la vulnerabilidad
    - apikey (varchar): clave necesaria para la identificación en las comunicaciones http con la fuente de información. Se debe incluir en cada petición http.
    - url (varchar) : enlace a la fuente de información.
    - active (boolean) : estado de la actividad de la fuente de información. True: En activo , False: No opera.

- PanVulnerabilitiesAssigners
  - Descripción: tabla que almacena la información relacionada con las entidades encargadas de detectar vulnerabilidades. NIST coopera con otras instituciones que comparten información de las vulnerabilidades que descubren.
  - Atributos:
    - id (Guid)
    - name (varchar) : nombre de la entidad que detectó la vulnerabilidad.
    - dateModifyToCVEs (date) : fecha en la que el registro se vio modificado, bien por un cambio en algún atributo, o por la creación o eliminación de una relación con vulnerabilidades.

- PanVulnerabilitiesSettings
    - Descripción: tabla que almacena información relacionada con la configuración de las peticiones a las fuentes de información.
    - Atributos:
        - id (Guid)
        - timeCVE (date) : tiempo máximo de una vulnerabilidad sin relaciones en la base de datos.
        - timeCPE (date) : tiempo máximo de un CPE sin relaciones en la base de datos.
        - resultsPerPage (int) : número de registros obtenidos por petición.

8.1.2.- Diagrama de clases

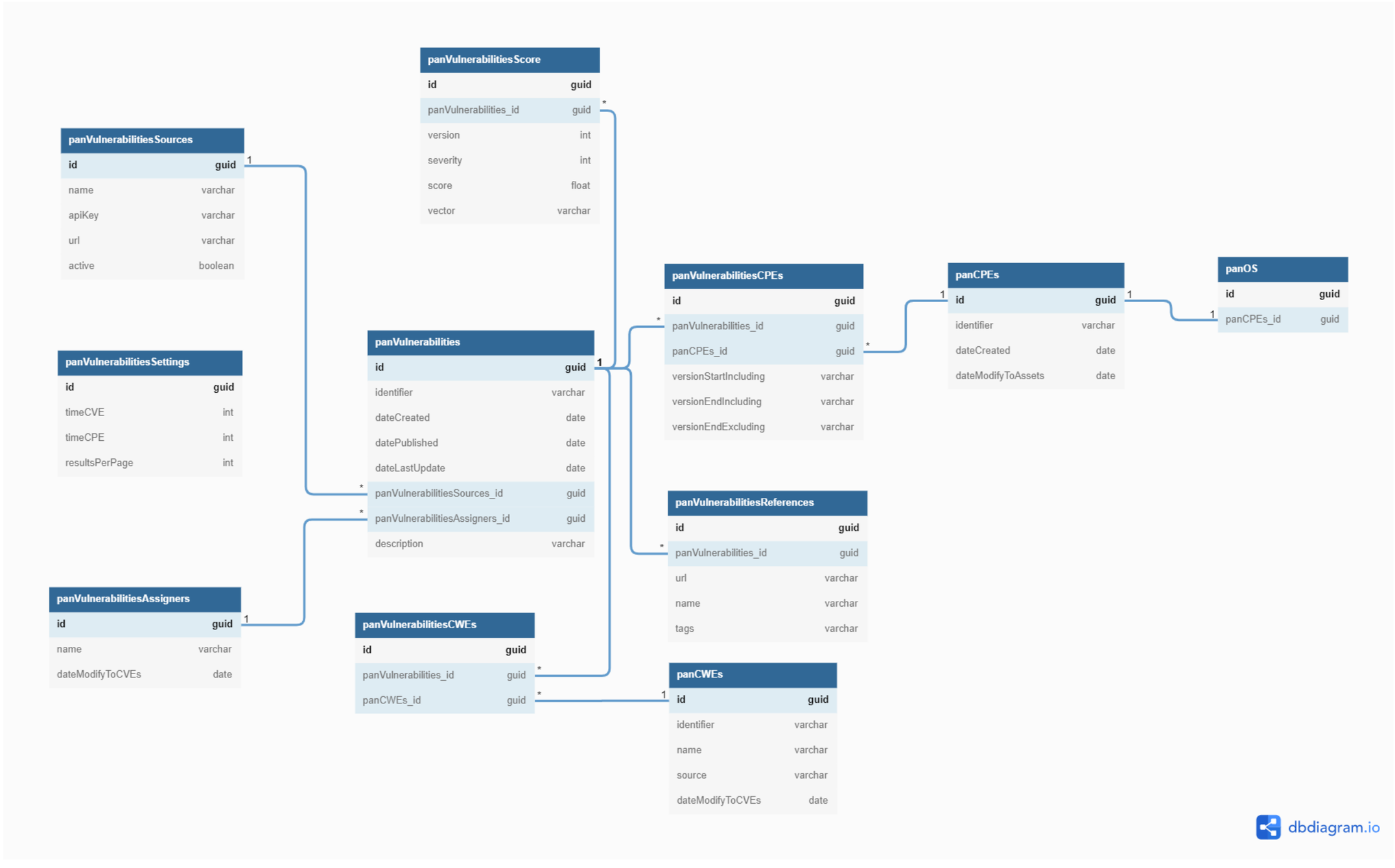


Figura 8.1.- Modelo de dominio de la ampliación de la Base de Datos

8.2.- MODELO FUNCIONAL DE LOS PROCESOS

En lugar de utilizar casos de uso en cada proceso que conforma el sistema de información, proceso de descarga de datos y proceso de purga, se ha definido un flujo de acciones, el cual permita identificar de manera clara y sencilla tanto las tareas y actividades que deberá realizar, como el ciclo de vida de los datos.

8.2.1.- Flujo de acciones – Proceso de descarga de datos

Proceso encargado de obtener datos de una determinada fuente de información , procesar dichos datos para filtrar aquella información que se quiere almacenar y guardarlos en la base de datos

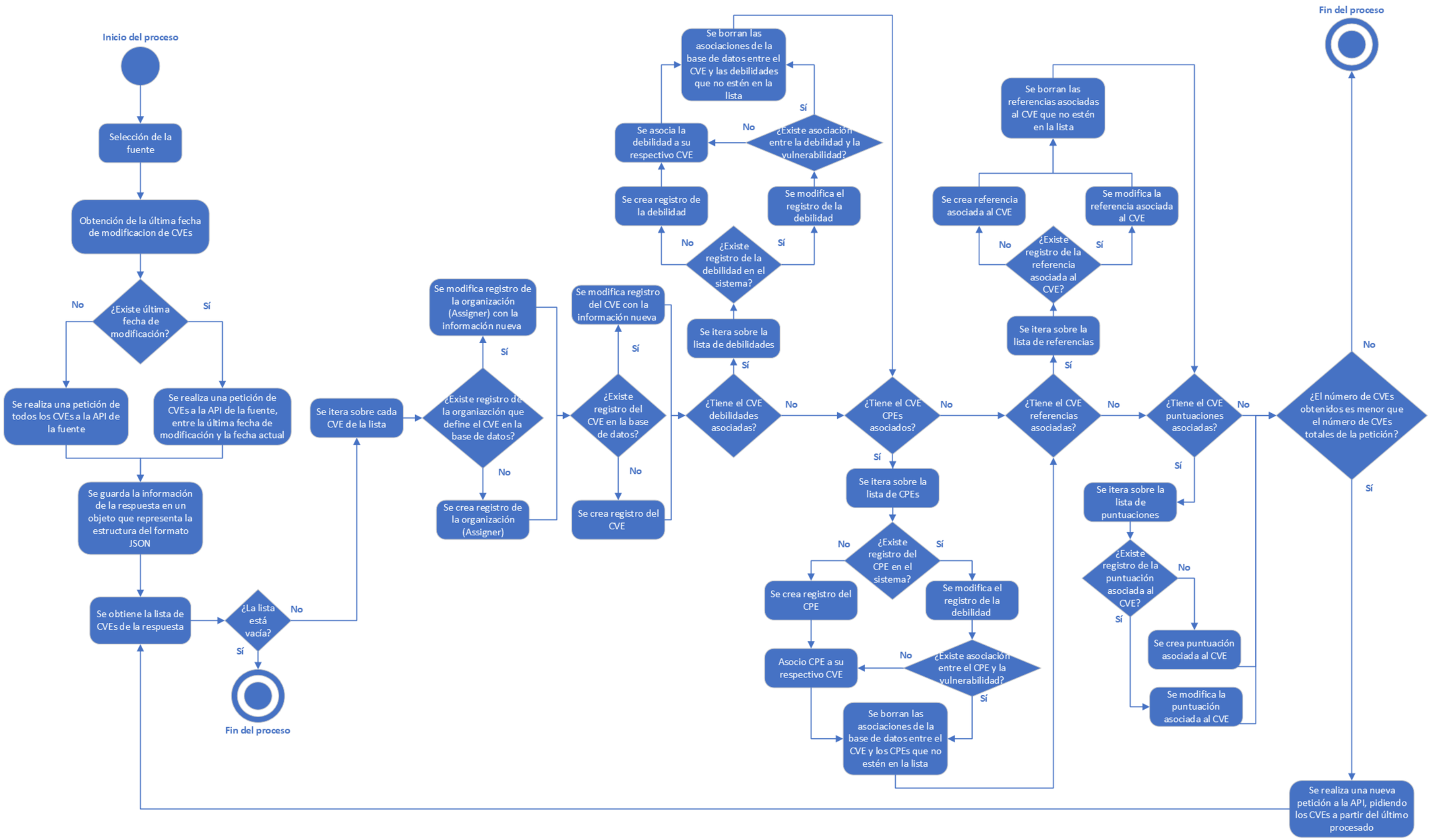


Figura 8.2.- Flujo de acciones - Proceso de descarga de datos

8.2.2.- Flujo de acciones – Proceso de purga

Proceso encargado de eliminar la información obsoleta de la base de datos al cumplir un determinado periodo de tiempo sin modificaciones y sin relaciones con otras tablas.

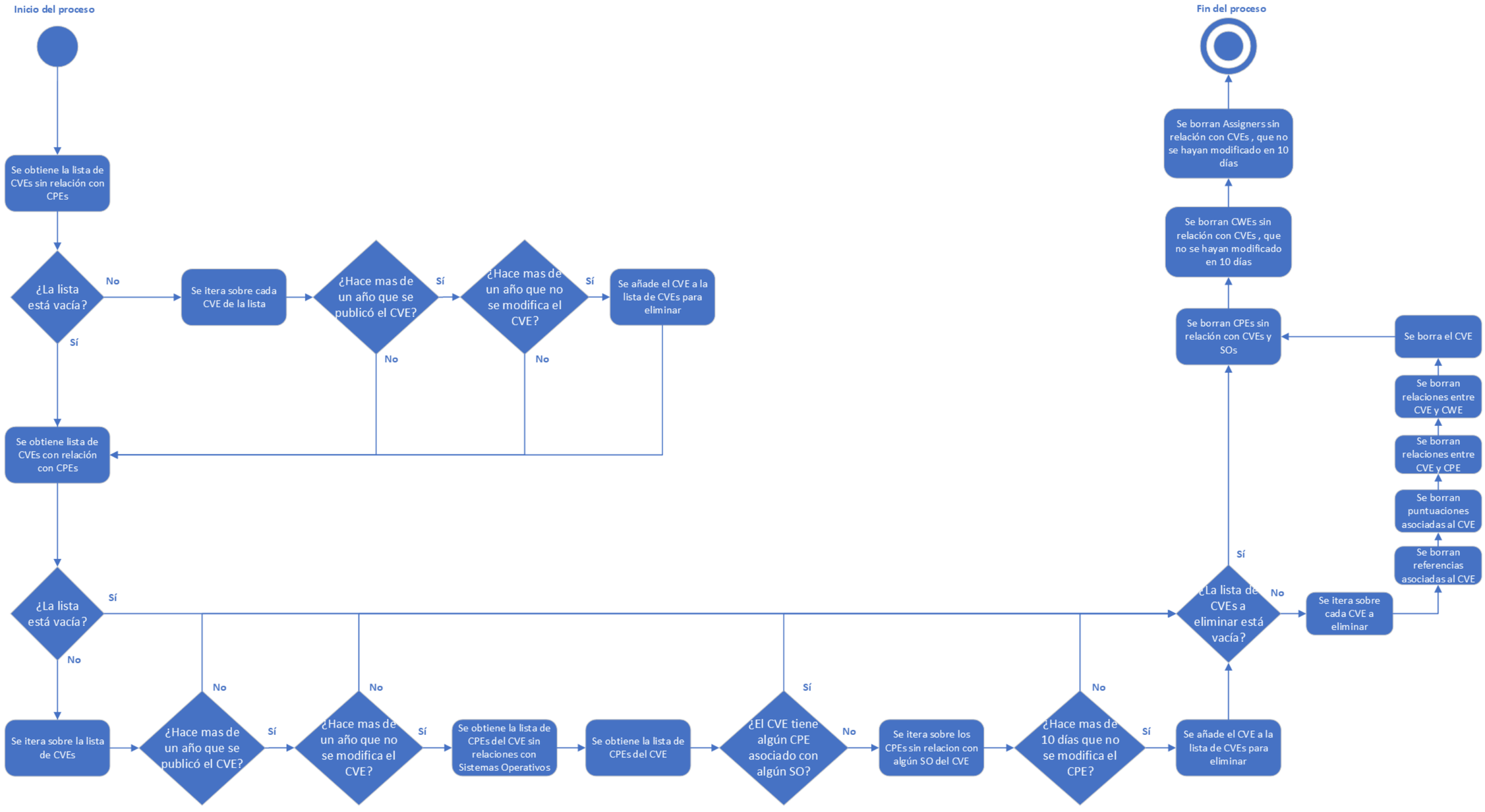


Figura 8.3.- Flujo de acciones - Proceso de purga

9. Diseño del sistema.

En este apartado se procederá a describir con detalle el diseño e implementación de las actividades y tareas establecidas en el alcance del proyecto, para elaborar una solución software que permita la integración de la información de NVD en ProactivaNET.

9.1.- DISEÑO DE ARQUITECTURA

9.1.1.- Descripción de la arquitectura

Tras el estudio de las alternativas planteadas para el diseño de la arquitectura del programa, se optó por la externalización de los procesos, con el fin de mejorar el rendimiento y reducir el impacto en los demás componentes del programa. Esta decisión implica que los procesos de descarga y purga de datos serán ejecutados desde el register como procesos externos, teniendo que establecer dicha conexión.

El *register* diariamente, llamará a los procesos de descarga y purgado para actualizar la información de seguridad. El proceso de descarga se conectará a través de la API a la base de datos de la fuente de información, obteniendo la información que se requiera y almacenándola, tras su procesamiento, en la base de datos de ProactivaNET, y , de la misma forma, el register ejecutará el proceso de purga, el cual eliminará la información obsoleta de la base de datos.

ProactivaNET desde sus última versiones adopta un modelo de distribución SaaS (Software as a Service). Este modelo implica que los datos y el soporte lógico del producto se alojan en unos servidores externos a la empresa, distribuidos por un proveedor, que en el caso de ProactivaNET es AWS (Amazon Web Service). De esta forma se permite a los clientes acceder a los servicios proporcionados por ProactivaNET a través de Internet de forma rápida y sencilla.

El despliegue de ambos procesos se realizará de forma óptima en los servidores de AWS, puesto que estos están en la nube. La ejecución de dichos procesos desde el register no se llevará a cabo en este proyecto, debido a que implica un impacto general que puede desestabilizar el producto global que se encuentra en producción proveyendo servicios a los clientes.

9.1.2.- Gráfico de componentes

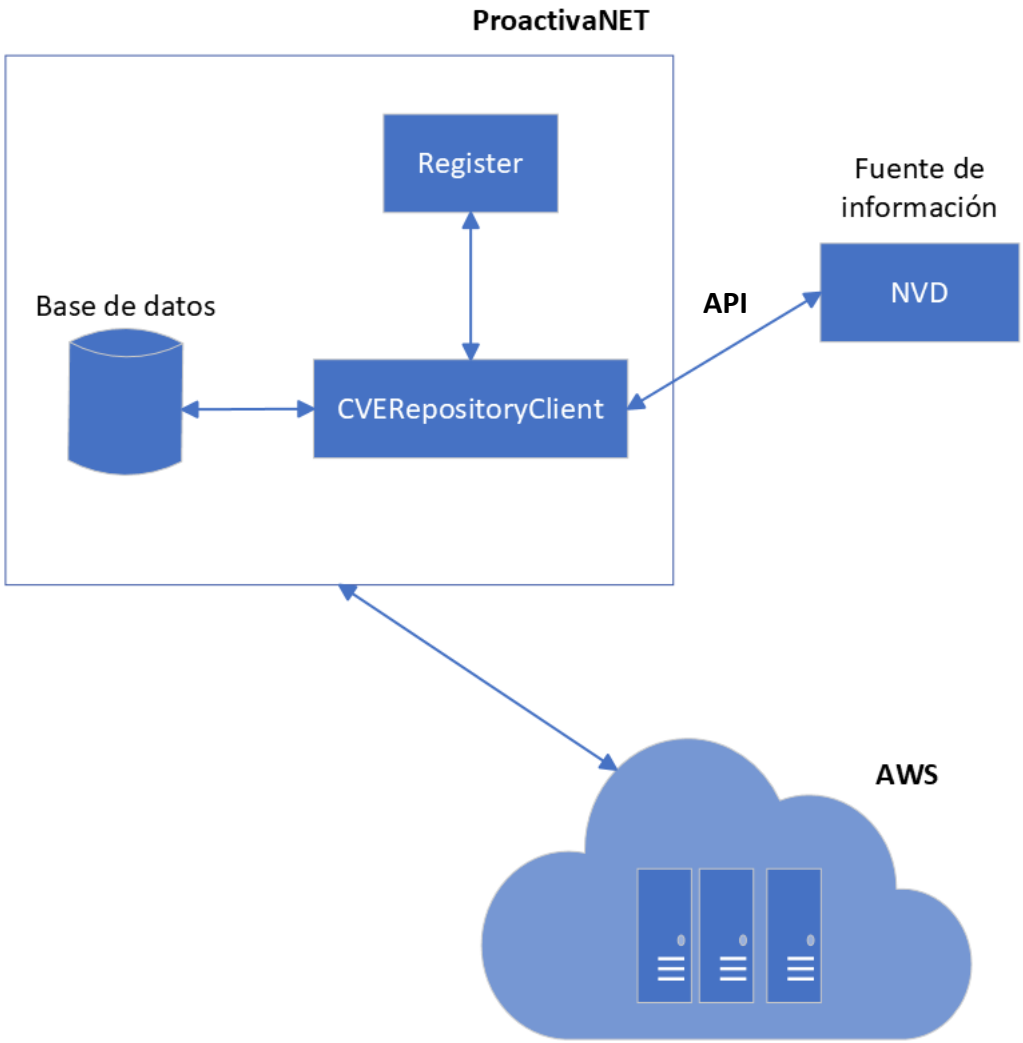


Figura 9.1.- Gráfico de componentes

9.1.3.- Estructura de la API de NVD

Este apartado es fruto del resumen del trabajo realizado previamente por José Carlos Díaz Ruénez, recogido en el documento: *Detección de vulnerabilidades en los activos inventariados en la herramienta Inventario de ProactivaNET - Integración entre ProactivaNET, NVD e INCIBE*.

NIST siguen una arquitectura REST. Este modelo permite el envío de peticiones, siguiendo el protocolo HTTP, para realizar consultas sobre NVD, donde un servidor de base de

datos enviará una respuesta con el resultado de la consulta. A través de la API que proporciona NVD se podrá establecer esta comunicación con sus servidores.

Con el objetivo de facilitar el envío de peticiones, por parte de instituciones que requieran una gran cantidad de información, NVD proporciona una clave, con la cual aumenta el número de peticiones HTTP por minuto, de 10 sin clave, a 100. Esta clave, además , actúa como token de sesión, permitiendo la autentificación y el registro de la entidad que solicita las peticiones.

Para el desarrollo de este proyecto se utilizará dicha clave, la cual está vinculada a un correo corporativo de ProactivaNET.

A través de la API se pueden realizar peticiones para obtener dos tipos de datos distintos, CVEs o CPEs. Dichas peticiones siguen un formato establecido por NVD, donde mediante una guía, describen como utilizar su API y como realizar las peticiones.

9.1.3.1.- Petición de CVE

La API nos permite solicitar una colección de CVEs o solo un único CVE. Siguiendo la documentación aportada por NVD [5], las peticiones HTTP para un único CVE seguirán la siguiente estructura, donde *cveId* es el identificador que representa a cada vulnerabilidad.

HTTP Method:	GET		
Content Type:	application/json		
URL:	https://services.nvd.nist.gov/rest/json/cve/1.0/<cveId>		
Parameters	**Type**	**Description**	**Required?**
cveId	URL path	CVE identifier.	Yes
addOns	URL query	See Include Official CPE Names.	No

Figura 9.2.- Estructura de petición HTTP para un único CVE

Un ejemplo de petición para un único CVE sería el siguiente, donde se pide el identificador ‘CVE-2021-43217’, vulnerabilidad de ejecución remota de código del sistema de cifrado de archivos (EFS).

https://services.nvd.nist.gov/rest/json/cve/1.0/CVE-2021-43217

En cambio, para la obtención de una colección de CVEs, las peticiones deberán seguir la siguiente estructura.

HTTP Method:	GET		
Content Type:	application/json		
URL:	https://services.nvd.nist.gov/rest/json/cves/1.0		
Parameters	**Type**	**Description**	**Required?**
startIndex	URL query	See Paging Results.	No
resultsPerPage			
pubStartDate		CVE publication or modification date range.	
pubEndDate			
modStartDate			
modEndDate			
includeMatchStringChange		See CPE Name.	
keyword		Free text keyword search.	
isExactMatch		See Keyword Search.	
cweId		CVE by CWE category.	
cvssV2Severity		CVE having base severity score.	
cvssV3Severity			
cvssV2Metrics		CVE having CVSS vectors.	
cvssV3Metrics			
cpeMatchString		CVE product applicability.	
addOns		See Include Official CPE Names.	

Figura 9.3.- Estructura de petición HTTP para una colección de CVEs

Un ejemplo de petición para una colección de CVEs sería el siguiente, donde se piden las cien primeras vulnerabilidades que se hayan modificado entre el 2021-08-04 y el 2021-10-22.

https://services.nvd.nist.gov/rest/json/cves/1.0?startIndex=0&resultsPerPage=100&modStartDate=2021-08-04T13:00:00:000%20UTC%2B01:00&modEndDate=2021-10-22T13:36:00:000%20UTC%2B01:00

9.1.3.2.- Petición de CPE

La API [6] nos permite solicitar únicamente una colección de CPEs, donde las peticiones HTTP seguirán la siguiente estructura.

HTTP Method:	GET		
Content Type:	application/json		
URL:	https://services.nvd.nist.gov/rest/json/cpes/1.0		
Parameters	**Type**	**Description**	**Required?**
startIndex	URL query	See Paging Results.	No
resultsPerPage			
modStartDate		CPE modification date range.	
modEndDate			
includeDeprecated		See deprecated CPEs	
keyword		Free text keyword search.	
cpeMatchString		CPE product applicability	
addOns		See Include Official CVEs	

Figura 9.4.- Estructura de petición HTTP para una colección de CPEs

Un ejemplo de petición para una colección de CPEs sería el siguiente, donde se piden los cien primeros activos que se hayan modificado entre el 2021-08-04 y el 2021-10-22.

https://services.nvd.nist.gov/rest/json/cpes/1.0?startIndex=0&resultsPerPage=100&modStartDate=2021-08-04T13:00:00:000%20UTC%2B01:00&modEndDate=2021-10-22T13:36:00:000%20UTC%2B01:00

9.1.3.3.- Gestión de colecciones de elementos en una petición

Para la gestión de peticiones que obtienen colecciones de elementos, ya sean CVEs o CPEs, hay que tener en cuenta los parámetros *startIndex* y *resultsPerPage*. Estos parámetros permiten iterar en cada petición sobre el conjunto de elementos.

El parámetro resultsPerPage permite seleccionar cuantos elementos se quiere obtener por cada petición, estando restringido a un máximo de 2000 elementos por petición, siendo el valor por defecto 20.

Por otra parte, *startIndex* permite seleccionar a partir de qué elemento de la colección se quiere obtener información, empezando por el elemento 0 como comportamiento por defecto.

Una petición puede obtener más elementos que el valor máximo para *resultsPerPage*, por lo que se deberán ir solicitando en distintas peticiones, cambiando el valor de startIndex, hasta obtener todos los elementos.

9.2.- DISEÑO DE DETALLE DE LOS PROCESOS

El programa deberá abarcar la obtención, procesamiento y carga de la información proporcionada por la API de NVD en la base de datos y una vez finalizado dicho proceso, el borrado de registros obsoletos de la base de datos.

El programa se ha desarrollará con el entorno de desarrollo Microsoft Visual Studio 2019, en el lenguaje de programación C# para mantener la coherencia con el código desarrollado para ProactivaNET. Para la implementación del modelo de datos, se ha utilizado la herramienta SQL Server Management Studio, donde se ha desarrollado los scripts de creación y modificación de la base de datos.

9.2.1.- Estructura del programa

La solución software que se ha desarrollado está dividida en dos proyectos. El proyecto denominado CVEsRepositoryClientCore contendrá toda la lógica de negocio, incluyendo ambos procesos de descarga y purgado de datos.

Debido a la arquitectura final del programa, el cual será invocado como un proceso externo por el *register*, proceso principal encargado del inventariado y de la gestión de activos, se definió un segundo proyecto denominado CVEsRepositoryClient, el cual únicamente tendrá la función de invocar los procesos de descarga y purga del proyecto CVEsRepositoryClientCore, con el fin de simular el comportamiento del *register*.

9.2.2.- Proceso de descarga de datos

Este proceso será el encargado de obtener los datos de la fuente de información, en este caso NVD, procesarlos y cargarlos en la base de datos.

El proceso comienza seleccionando la fuente a la que se realizaran las peticiones. Una vez seleccionada la fuente, en este caso NVD, para obtener la clave de la API que permitirá el aumento de peticiones por minuto a 200, se formará la petición que se va a realizar, la cual depende de varios factores.

En primer lugar, se obtendrá la última fecha en la que se modificó una vulnerabilidad en la base de datos. Esta fecha determinará el periodo a partir del cual se querrá solicitar información a NVD. Solicitando la información a partir de la última fecha en la que se modificó

una vulnerabilidad en la base de datos, se obtendrá toda la nueva información disponible desde la última descarga de datos. En este momento pueden ocurrir dos situaciones.

En caso de que sea la primera descarga de datos y no haya aún registros en la base de datos se realizará una petición sin filtro de fechas para obtener toda la información disponible. En otro caso, si existe un registro con una fecha de última modificación, la cual representa la fecha de la última descarga de datos, se realizará la petición aplicando dicho filtro.

Una vez realizada la petición, se obtendrá una respuesta en formato JSON. Dicho formato se caracteriza por estar formado por una colección de pares nombre-valor. Para poder manejar la información de la respuesta se ha diseñado un conjunto de clases, las cuales son abstracciones de estos pares nombre-valor, debido a la complejidad de la estructura de la respuesta por la cantidad de objetos y listas que contiene. Asignando a cada objeto de estas clases, la información de la respuesta JSON que le corresponde, facilitamos el acceso y su manejo. Un ejemplo de esta abstracción sería el siguiente:

```
1 reference | César del Río Fernández, 21 days ago | 1 author, 1 change
public class ResultCVE
{
    0 references | César del Río Fernández, 21 days ago | 1 author, 1 change
    public string CVE_data_format { get; set; }
    0 references | César del Río Fernández, 21 days ago | 1 author, 1 change
    public string CVE_data_version { get; set; }
    0 references | César del Río Fernández, 21 days ago | 1 author, 1 change
    public string CVE_data_timestamp { get; set; }
    1 reference | César del Río Fernández, 21 days ago | 1 author, 1 change
    public List<CVEItem> CVE_Items { get; set; }
}

3 references | César del Río Fernández, 21 days ago | 1 author, 1 change
public class RootCVE
{
    1 reference | César del Río Fernández, 21 days ago | 1 author, 1 change
    public int resultsPerPage { get; set; }
    0 references | César del Río Fernández, 21 days ago | 1 author, 1 change
    public int startIndex { get; set; }
    1 reference | César del Río Fernández, 21 days ago | 1 author, 1 change
    public int totalResults { get; set; }
    1 reference | César del Río Fernández, 21 days ago | 1 author, 1 change
    public ResultCVE result { get; set; }
}
```

Figura 9.5.- Clases para el manejo de estructuras JSON

```
{
  "resultsPerPage": 2000,
  "startIndex": 0,
  "totalResults": 177664,
  "result": {
    "CVE_data_type": "CVE",
    "CVE_data_format": "MITRE",
    "CVE_data_version": "4.0",
    "CVE_data_timestamp": "2022-06-09T17:20Z",
    "CVE_Items": [
      {
        "cve": {
          "data_type": "CVE",
          "data_format": "MITRE",
          "data_version": "4.0",
          "CVE_data_meta": {
            "ID": "CVE-2021-26633",
            "ASSIGNER": "vuln@krcert.or.kr"
```


Figura 9.6.- Formato JSON de una respuesta

Las respuestas a la API de NVD comienzan con los pares *resultsPerPage*, *startIndex*, *totalResults* y el objeto *result*, como se puede apreciar en la Figura 9.6. Esta información se almacenará en los atributos de la clase de C# RootCVE y de manera recursiva, la información del objeto *result* se almacenará en sus respectivos atributos dentro de la clase ResultCVE.

Una vez procesada la información para facilitar su manejo en forma de objetos de C#, se procede a la carga de la información en la base de datos.

Como se puede observar en la Figura 9.6, el número de registros de la respuesta a la petición inicial es de 177664, valor del atributo *totalResults* del JSON, pero solo se devuelven 2000, valor del atributo *resultsPerPage*, comenzando por el registro con índice 0, valor de *startIndex*. Una vez procesados los 2000 primeros registros, habrá que realizar una nueva petición, indicando que se quieren recibir los registros a partir del índice 2000. Esta iteración se tendrá que repetir hasta recibir todos los registros.

En cada petición, para cada elemento de la lista de vulnerabilidades, CVE_Items en la clase *ResultCVE* de la Figura 9.5, se comprueba en primer lugar si existe registro en la base de datos de la información relativa a una entidad del modelo de dominio. En caso de que exista, se comprueba si se ha modificado, y si es el caso se modifica dicho registro. En el caso de que no exista se crea el registro.

El orden en el que se va procesando la información relativa a cada entidad es de vital importancia debido a las relaciones que existen entre ellas. En primer lugar se maneja la información relacionada con los Assigners, organizaciones que detectan las vulnerabilidades, dado que para crear un registro de una vulnerabilidad se necesita conocer que organización la detectó. Sucesivamente se gestionan las vulnerabilidades, núcleo de todo el modelo de dominio. Y finalmente se gestiona la información adicional sobre las vulnerabilidades en este orden, para poder respetar la integridad referencial del modelo de dominio : CWEs (debilidades que conlleva la vulnerabilidad), CPEs (activos, en este caso sistemas operativos, a los que se aplica esta vulnerabilidad), referencias de la vulnerabilidad y puntuaciones de severidad.

Una vez gestionados todos los registros resultantes de la primera petición se finaliza el proceso de obtención y carga de datos.

9.2.3.- Proceso de purga

Tras haber cargado en la base de datos la información de NVD con el proceso anterior, se procede a eliminar aquellos registros que son obsoletos o que no aportan información relevante.

Cada vez que se realice este proceso se eliminarán aquellas vulnerabilidades que cumplan alguno de los siguientes requisitos:

- Una vulnerabilidad no tiene relación con algún CPE de la base de datos y hace más de un año que se publicó y que no se modifica. No tener relación con algún CPE determina que dicha vulnerabilidad no está asociada a ningún sistema operativo inventariado de la base de datos. Por lo tanto, en el caso de que una vulnerabilidad se mantenga en la base de datos durante un año sin tener alguna relación con un sistema operativo, se eliminará.

- La vulnerabilidad hace más de un año que se publicó y que no se modifica y tiene CPEs asociados, pero estos CPEs no tiene relación con algún sistema operativo desde hace más de 10 días. Un sistema operativo puede desaparecer de la base de datos por decisión de una organización debido a una actualización del software o cambios de versión. En este caso se eliminarían las relaciones con el CPE de dicho sistema operativo. Por lo tanto, en el caso de que una vulnerabilidad se mantenga en la base de datos durante un año con CPEs asociados,

pero sin que los CPEs tengan relación con ningún sistema operativo desde hace más de diez días, se eliminará.

De la misma forma que el orden de creación de los registros es importante para mantener la integridad referencial, el orden de eliminación de los registros también. En primer lugar se elimina la información relacionada con las vulnerabilidades, en el siguiente orden: referencias de la vulnerabilidad, puntuaciones de severidad, relaciones con CPEs (activos, en este caso sistemas operativos, a los que se aplica esta vulnerabilidad), relaciones con CWEs (debilidades que conlleva la vulnerabilidad) y finalmente el registro de la vulnerabilidad.

A continuación se comienzan a borrar los registros de CPEs que han quedado sin relación con vulnerabilidades y sistemas operativos y por último, se procede a eliminar aquellos registros de CWEs y Assigners que no tengan relaciones con vulnerabilidades y que no se hayan modificado desde hace más de diez días.

9.2.4.- Tiempo en base de datos de un registro

De la misma forma que una organización puede actualizar un sistema operativo y pasar a una nueva versión, eliminando de su inventario dicho sistema operativo, se puede dar el caso de que tras un periodo de pruebas, se retracten y vuelvan a la versión inicial.

Por este motivo, se determina un tiempo prudencial de diez días para confirmar la eliminación de la relación entre un sistema operativo y un CPE, para posteriormente, proceder a la eliminación de vulnerabilidades siguiendo el criterio anterior.

En el caso de las vulnerabilidades, se determina un tiempo de un año para la confirmación de su eliminación, dado que esta entidad es el eje central de todo el proyecto. El usuario final del programa podrá disponer de todas las vulnerabilidades, le afecten o no a sus activos, del último año. Esta funcionalidad permite a los encargados de seguridad de una organización consultar vulnerabilidades que se han detectado recientemente sobre activos que no están inventariados en su empresa, aportando información interesante sobre las vulnerabilidades que puede acarrear la actualización a una versión nueva o la migración a otro tipo de software.

9.2.5.- Función de los triggers en los procesos

Los triggers en el momento del almacenamiento y eliminación de la información en la base de datos serán los encargados de gestionar las fechas de última modificación. Esta fecha tomará el valor del día y hora del momento en el que se activen los triggers.

Se han implementado los siguientes triggers, agrupados por la tabla con la que están asociados:

- PanOs
    - Al eliminar un sistema operativo, se cambia la última fecha de modificación de los CPEs asociados a dicho sistema operativo, tomando como valor la fecha del momento en el que se elimina el registro de PanOs.
    - Al inventariar un sistema operativo, se cambia la última fecha de modificación de los CPEs asociados a dicho sistema operativo, tomando como valor la fecha del momento en el que se crea el registro en PanOs.
    - Al cambiar el valor del atributo CPE de un sistema operativo, se cambia la última fecha de modificación tanto del valor antiguo de CPE, como del nuevo valor, tomando como valor la fecha del momento en el que se modificó el atributo en PanOs.
- PanVulnerabilities
    - Al eliminar una vulnerabilidad, se cambia la última fecha de modificación de los Assigners asociados a dicha vulnerabilidad, tomando como valor la fecha del momento en el que se elimina el registro de PanVulnerabilities.
    - Al registrar una nueva vulnerabilidad, se cambia la última fecha de modificación de los Assigners asociados a dicha vulnerabilidad, tomando como valor la fecha del momento en el que se crea el registro en PanVulnerabilities.
    - Al cambiar el valor del atributo Assigners de una vulnerabilidad, se cambia la última fecha de modificación tanto del valor antiguo de la organización que detectó la vulnerabilidad, como del nuevo valor, tomando como valor la fecha del momento en el que se modificó el atributo en PanVulnerabilities.

- PanVulnerabilitiesCWEs

    - Al eliminar una relación entre una debilidad (CWE) y una vulnerabilidad (CVE), se cambia la última fecha de modificación del CWE asociado a dicha vulnerabilidad, tomando como valor la fecha del momento en el que se elimina el registro de PanVulnerabilitiesCWEs.
    - Al registrar una nueva relación entre una debilidad y una vulnerabilidad, se cambia la última fecha de modificación del CWE asociado a dicha vulnerabilidad, tomando como valor la fecha del momento en el que se crea el registro en PanVulnerabilitiesCWEs.
    - Al cambiar la debilidad asociada en un registro a una vulnerabilidad, se cambia la última fecha de modificación tanto del valor antiguo de la debilidad, como del nuevo valor, tomando como valor la fecha del momento en el que se modificó el atributo en PanVulnerabilitiesCWEs.

Por tanto, para el correcto funcionamiento del proceso de purga, es necesario la utilización de triggers que mantengan actualizada la última fecha de modificación de un registro, para poder tomar la decisión de eliminarlo en caso de que supere un tiempo determinado.

9.3.- DISEÑO FÍSICO DE DATOS

La base de datos que utiliza el software ProactivaNET está desarrollada bajo unos estándares y una nomenclatura específicos. Estos se tomarán como referencia para el desarrollo e implementación de la ampliación que permitirá el almacenamiento de la información obtenida de NVD.

A partir del modelo de dominio y del diccionario de datos, definidos en la fase de análisis del sistema, se ha elaborado un script que llevará acabo la creación de la base de datos.

Para la implementación de la base de datos , cada clase del modelo de dominio definido en el apartado 8.1.- MODELO DE DOMINIO , durante la fase de análisis del sistema, se ha implementado como una tabla.

9.3.1.- Integración con la base de datos de ProactivaNET

La integración de la ampliación de la base de datos que se desarrollará para este proyecto se realizará a través de la tabla PanOs. Esta tabla existente en la base de datos actual de ProactivaNET registra todos los sistemas operativos de una determinada organización. Cada sistema operativo tendrá un CPE asociado, por lo que la ampliación de base de datos se integrará relacionando la tabla que almacene los CPEs con el atributo de PanOs que determina su CPE. De esta forma se relacionará un sistema operativo, con toda la información de seguridad que se recoge a través de NVD, respecto a vulnerabilidades y debilidades.

9.3.2.- Estándares y Nomenclatura

En una base de datos de gran tamaño como la desarrollada en ProactivaNET, formada por más de 1300 tablas, la utilización de una nomenclatura y estándares bien definidos desde el inicio es de vital importancia. En primer lugar para mantener la coherencia entre todas las entidades y relaciones, que de otra manera seria imposible de gestionar debido al tamaño de la base de datos, y en segundo lugar, para que los nombres de tablas y atributos sean autodescriptivos. De esta forma, un usuario que no esté familiarizado con la base de datos puede comprender a simple vista qué tipo de entidad se almacena en una tabla y qué tipo de información describe un atributo.

A continuación se procederá a describir los estándares y la nomenclatura utilizada para la realización de la ampliación que se integrará en la base de datos de ProactivaNET.

9.3.2.1.- Nomenclatura

Para conseguir que todo nombre de entidad o atributo de la base de datos sea autodescriptivo se ha seguido una nomenclatura definida para toda la base de datos de ProactivaNET.

Todos los nombres de entidades comienzan con un prefijo, el cual determina a qué aplicación de ProactivaNET pertenece dicha entidad. ProactivaNET tiene dos aplicaciones principales:

- Inventario: componente de ProactivaNET encargado de recopilar y visualizar la información de todos los componentes tecnológicos de una red, ya sean hardware o software, que se han detectado, controlado y monitorizado.
- ServiceDesk: componente de ProactivaNET cuya labor es la gestión de los servicios TI de una organización, el cual abarca servicios como la gestión del catálogo de servicios, de problemas e incidencias o de peticiones de los usuarios.

Los prefijos usados a la hora de nombrar una entidad en base de datos son:

- pan (ProActivanet iNventory) : determina las entidades pertenecientes al Inventario.
- pad (ProActivanet serviceDesk): determina las entidades pertenecientes al ServiceDesk.
- paw (ProActivanet Web): determina las entidades pertenecientes a componentes de librerías de ProactivaNET.

Dado que la información obtenida desde NVD afecta a la seguridad de los activos monitorizados de una organización y estos se gestionan a través del inventario, todas las tablas de la ampliación de la base de datos que se definirán en el modelo de dominio comienzan con el prefijo *pan*.

Tanto para el nombrado de las tablas, como para el nombrado de atributos, se utiliza una técnica denominada notación camelizada, la cual consiste en que la primera letra de cada palabra es mayúscula, excepto para la primera palabra. Un ejemplo de esta técnica sería *panVulnerabilitiesSettings*, un nombre de una de las tablas del modelo de dominio de la ampliación, donde la primera letra del prefijo no se modifica y las primeras letras de las palabras vulnerabilitites y settings son mayúsculas.

En relación con las claves ajenas, atributo que se corresponde con la clave primaria de otra tabla, su nombre se construye concatenando el nombre de la tabla de la clave primaria que se relaciona y el nombre de dicha clave primaria, separados por un guion bajo. Un ejemplo sería *panVulnerabilities_id*, clave ajena que se encuentra en *panVulnerabilitiesScore*, tabla del modelo de dominio de la ampliación. El nombre de este atributo, al ser una clave ajena, está formado por el nombre de la tabla que se relaciona, *panVulnerabilities* y el nombre de la clave primaria de esa tabla que se relaciona, *id*, separado por un guion bajo.

Por último en cuanto a nomenclatura, los nombres de las tablas resultantes de relaciones n a n, donde varios registros de una tabla se relacionan con varios registros de otra tabla, se conforman con los nombres de las dos entidades que se relacionan, sin el prefijo que determina a que aplicación pertenecen. Un ejemplo de esta técnica sería *panVulnerabilitiesCPEs*, un nombre de una de las tablas del modelo de dominio de la ampliación. Esta tabla es resultante de una relación n a n entre panVulnerabilities y panCPEs, por lo que su nombre se construye concatenando, en primer lugar el prefijo pan, del inventario, y los nombres de las tablas que relaciona sin el prefijo, Vulnerabilities y CPEs.

9.3.2.2.- Estándares

Con el objetivo de unificar la creación de claves primarias, toda clave primaria tendrá como nombre *id*, y será de tipo Guid (Globally Unique Identifier). Este tipo de dato binario tiene un tamaño de 16 bytes y se caracteriza por tener un valor único para todas las tablas, por lo que el tipo Guid se puede considerar una clave primaria global.

De esta forma, asignando el tipo Guid a las clave primarias, se solucionan problemas de redundancia a la hora de trabajar con distintas bases de datos, dado que podrían repetirse valores de claves primarias para registros de ambas bases de datos.

Por otra parte, con el mismo objetivo de reducir la redundancia de los datos, la base de datos de ProactivaNET ha seguido un proceso de normalización, aunque debido a su tamaño, esto puede repercutir en términos de eficiencia, por lo que en algunos casos, se opta por la repetición de datos, con el fin de aumentar la eficiencia del acceso a estos mismos.

Por el mismo motivo, se han aplicado un conjunto de reglas y procesos para normalizar el modelo de dominio que se ha desarrollado, sin perder de vista la eficiencia.

Por último, la base de datos de ProactivaNET, desde su creación está en continuo desarrollo y una de las propiedades que la caracteriza es el soporte al mismo tiempo de dos gestores de base de datos, SQL-Server y Oracle. Esto dificulta el diseño de querys y comandos debido a que deben de ser soportados por ambos gestores, los cuales tienen lenguajes de programación de bases de datos distintos. Por lo tanto, se debe tener en cuenta en todo momento esta dualidad a la hora de desarrollar los scripts de creación de la base de datos y las querys.

9.3.3.- Script de creación de base de datos

Una vez definido el modelo de dominio, el siguiente paso es la creación de un script con el cual poder crear e integrar dicho dominio en la base de datos actual de ProactivaNET. Como se ha nombrado anteriormente, para el diseño de este script se ha tenido en cuenta el soporte de SQL-Server y Oracle como gestores de base de datos.

El archivo resultante denominado *CreateDataBase.sql* se adjuntará a este documento.

El script está estructurado en cinco partes por una cuestión de organización, con el objetivo de facilitar la comprensión y localización de los comandos. Dichas partes son:

0. Tables: en esta sección del archivo se crean las tablas y atributos del modelo de dominio, determinando el tipo de los atributos y la posibilidad de que puedan tomar el valor null o no.
1. Primary keys : en esta sección del archivo se determina para cada tabla sus claves primarias.
2. Defaults: en esta sección del archivo se determinan los valores por defecto que tomarán algunos atributos de las tablas en caso de no ser introducidos en el momento de creación del registro.
3. Foreign keys: en esta sección del archivo se determina para cada tabla sus claves ajenas.
4. Triggers: en esta sección se crean triggers que modifican atributos de un registro al realizarse una determinada acción, ya sea de creación, borrado o modificación. La implementación de los triggers es de vital importancia para mantener la integridad referencial en la base de datos y para el correcto funcionamiento de los procesos de descarga de datos y de purga, como se verá en los siguientes apartados.

10. Plan de pruebas.

Tras finalizar el desarrollo del programa, se ha llevado a cabo un plan de pruebas, con el objetivo de mejorar la calidad del producto final, gracias al buen cumplimiento de los requisitos funcionales y de rendimiento.

Para ello se ha realizado un diseño de situaciones de prueba que cubra todos los requisitos de los procesos de negocio del programa, el proceso de descarga y almacenamiento de datos y el proceso de purga. A partir del diseño, se ha implementado un conjunto de casos de prueba, cuyo objetivo es encontrar fallos en el funcionamiento del programa.

Todos los fallos detectados a partir de los casos de prueba han sido corregidos y posteriormente testeados de nuevo.

10.1.- ENTORNO DE PRUEBAS

El siguiente plan de pruebas se ha ejecutado en un entorno de pruebas con unas características y propiedades concretas, con el objetivo de especificar aspectos relacionados con la funcionalidad y la eficiencia de los procesos. El sistema en el cual se ha llevado a cabo el plan de pruebas dispone de un procesador Intel Xeon Gold 6140 y una memoria RAM de 8 gb.

Las características del entorno de pruebas se deberán de tener en cuenta a la hora de replicar los casos de prueba, en cuanto a resultados de eficiencia.

10.2.- SITUACIONES DE PRUEBA

Partiendo de la especificación y requisitos del programa, se diseñarán las siguientes situaciones de prueba, agrupándolas en función de que aspecto o característica del programa se quiere probar.

Se realizará un único diseño de situaciones de prueba utilizando diversas técnicas. El uso de varias técnicas añade profundidad al plan de pruebas, aumentando la probabilidad de encontrar fallos en el software.

Las técnicas utilizadas en el diseño del conjunto de situaciones de prueba son las siguientes:

- Clases de equivalencia : diseño de situaciones de prueba determinando conjuntos de datos para los cuales el programa se comporta de manera similar.
- AVL : técnica que se utiliza para complementar el diseño de clases de equivalencia, la cual consiste en crear situaciones de prueba donde se tomen los valores límite de una clase de equivalencia. Proporciona mayor profundidad a la técnica de clases de equivalencia al ampliar el rango de valores testeados para un mismo conjunto de datos.
- MCDC (condición/decisión modificada) : dentro de las técnicas basadas en condiciones, esta técnica proporciona una gran nivel de efectividad en cuanto a la detección de fallos, reduciendo considerablemente el número de situaciones de prueba necesarios. Consiste en diseñar situaciones de prueba con el objetivo de que cada condición afecte de forma independiente al resultado de la decisión.

10.2.1.- Situaciones de prueba para proceso de descarga de datos

CLASES DE EQUIVALENCIA

1. Seleccionar una url determinada
   1.1. Obtención de url automática
      1.1.1. No existe última fecha de modificación en base de datos
      1.1.2. Existe última fecha de modificación en base de datos
   1.2. Url determinada
2. Número total de elementos de la respuesta
   2.1. > número de elementos máximos por respuesta
   2.2. <= número de elementos máximos por respuesta
3. Número de elementos de la lista de CVEs
   3.1. 0
   3.2. > 0
      3.2.1. 1
      3.2.2. Todos los registros de la fuente
4. Registro del Assigner
   4.1. No existe en base de datos
   4.2. Existe
5. Registro del CVE

5.1. No existe en base de datos

5.2. Existe

6. Número de debilidades (CWE) asociadas a un CVE
   6.1. 0
   6.2. > 0
      6.2.1. 1
         6.2.1.1. Existe registro
            6.2.1.1.1. Existe relación entre CWE y CVE
            6.2.1.1.2. No existe relación entre CWE y CVE
         6.2.1.2. No existe registro
7. Número de CPEs asociadas a un CVE
   7.1. 0
   7.2. > 0
      7.2.1. 1
         7.2.1.1. Existe registro
            7.2.1.1.1. Existe relación entre CPE y CVE
            7.2.1.1.2. No existe relación entre CPE y CVE
         7.2.1.2. No existe registro
8. Número de referencias asociadas a un CVE
   8.1. 0
   8.2. > 0
      8.2.1. 1
         8.2.1.1. Existe registro
         8.2.1.2. No existe registro
9. Número de puntuaciones asociadas a un CVE
   9.1. 0
   9.2. > 0
      9.2.1. 1
      9.2.2. 2
10. Registro puntuaciones
   10.1. Existe registro
   10.2. No existe registro
11. Seleccionar una fuente

11.1. Registrada en base de datos

11.2. No registrada en base de datos (CI)

10.2.2.- Situaciones de prueba para proceso de purga

En las tablas para representar las situaciones de prueba diseñadas mediante la técnica MCDC se remarcará en cada caso las condiciones que influyen independientemente en el resultado de la prueba.

CLASES DE EQUIVALENCIA

12. Número de elementos de la lista de CVEs sin relación con CPEs
    12.1. 0
    12.2. > 0
        12.2.1. 1
13. Número de elementos de la lista de CVEs con relación con CPEs
    13.1. 0
    13.2. > 0
        13.2.1. 1
14. Número de CVEs para borrar
    14.1. 0
    14.2. > 0
        14.2.1. 1
        14.2.2. 2
15. Número de referencias asociadas al CVE para borrar
    15.1. 0
    15.2. > 0
        15.2.1. 1
16. Número de puntuaciones asociadas al CVE para borrar
    16.1. 0
    16.2. > 0
        16.2.1. 1
17. Número de relaciones con CPEs asociadas al CVE para borrar
    17.1. 0
    17.2. > 0

17.2.1. 1

18. Número de relaciones con CWEs asociadas al CVE para borrar

18.1. 0

18.2. > 0

18.2.1. 1

19. Número de CPEs sin relaciones para borrar

19.1. 0

19.2. > 0

19.2.1. 1

20. CWEs sin relaciones para borrar

20.1. Número

20.1.1. 0

20.1.2. > 0

20.1.2.1. 1

20.1.2.1.1. Última fecha de modificación > 10 días

20.1.2.1.2. Última fecha de modificación <= 10 días

21. Assigners sin relaciones para borrar

21.1. Número

21.1.1. 0

21.1.2. > 0

21.1.2.1. 1

21.1.2.1.1. Última fecha de modificación > 10 días

21.1.2.1.2. Última fecha de modificación <= 10 días

MCDC

22. Eliminar CVE sin relaciones con CPEs

- C1: Fecha de publicación del CVE - Fecha actual > 365 días
- C2: Fecha de última modificación del CVE - Fecha actual > 365 días
- Salida esperada : Se añade el CVE a la lista de CVEs para eliminar
- Decisión: C1 and C2 = Salida

Situación de prueba	C1	C2	Salida
22.1	C	C	C
22.2	F	C	F
22.3	C	F	F

Tabla 10-1.- MCDC para eliminar CVEs sin relaciones con CPEs

23. Eliminar CVE con relaciones con CPEs

- C1: Fecha de publicación del CVE - Fecha actual > 365 días
- C2: Fecha de última modificación del CVE - Fecha actual > 365 días
- C3: Todos los CPEs del CVE no están relacionados con sistemas operativos
  CPEs del CVE – CPEs del CVE sin relación con sistemas operativos == conjunto vacío
- C4: Fecha de última modificación de todos los CPEs – Fecha actual > 10 días
- Salida esperada : Se añade el CVE a la lista de CVEs para eliminar

- Decisión: C1 and C2 and C3 and C4 = Salida.

Situación de prueba	C1	C2	C3	C4	Salida
23.1	C	C	C	C	C
23.2	F	C	C	C	F
23.3	C	F	C	C	F
23.4	C	C	F	C	F
23.5	C	C	C	F	F

Tabla 10-2.- MCDC para eliminar CVEs con relaciones con CPEs

10.3.- BASES DE DATOS PARA PRUEBAS

Para la realización de los casos de uso se ha utilizado la base de datos en distintos estados. En este apartado se describirá cada una de los distintas situaciones en las que se encontraba la base de datos, con el objetivo de facilitar la comprensión y trazabilidad de los casos de prueba.

Junto a este documento se adjuntarán varios scripts de bases de datos, los cuales permitirán replicar una situación concreta para un caso de prueba determinado. A continuación se describirán estos scripts.

1. Archivo “CreateDataBase.sql” : script que tiene como labor la creación de la base de datos. Únicamente tendrá almacenada la información relacionada con la fuente NVD para poder realizar las peticiones. Se debe ejecutar en primer lugar.
2. Archivo “ResetDataBase.sql” : script que elimina todos los datos excepto los relacionados con fuente NVD para poder realizar las peticiones. Deja la base de datos en su estado inicial, después de ejecutar el script “CreateDataBase.sql”
3. Archivo “CP_2.sql” : script que reinicia la base de datos y almacena toda la información relacionada con el CVE : ‘CVE-2020-7668’. Este archivo registra su Assigner, el propio CVE, su único CWE y CPE y sus respectivas relaciones, su única referencia y sus dos puntuaciones en función de la versión.
4. Archivo “CP_3.sql” : script que reinicia la base de datos y almacena parte de la información relacionada con el CVE : ‘CVE-2020-7668’. Este archivo registra solo su único CWE y CPE aunque no sus respectivas relaciones con el CVE, de manera que no almacena su Assigner, el propio CVE, su referencia y sus dos puntuaciones en función de la versión.
5. Archivo “CP_9.sql” : script que reinicia la base de datos y registra un CVE cuya última fecha de modificación sea la fecha del momento en el que se ejecuta el script.
6. Archivo “CP_10.sql” : script que reinicia la base de datos y registra dos CVEs. El CVE-1 tiene asociado un CPE que no tiene relaciones con sistemas operativos, y el CVE-2 , sin relaciones con CPEs. El objetivo de este script es comprobar la eliminación de vulnerabilidades sin relaciones con CPEs o con

CPEs que no estén asociados a sistemas operativos, cuando ha pasado un año desde su última fecha de modificación.

7. Archivo “CP_11.sql” : script que reinicia la base de datos y registra un Assigner y un CWE sin relaciones. El objetivo de este script es la comprobación del borrado únicamente de dichas entidades, el Assigner con el atributo ‘name’ que toma el valor ‘prueba’ y el CWE con el atributo ‘identifier’ que toma el valor ‘CWE-prueba’.
8. Archivo “CP_13.sql” : script que reinicia la base de datos y registra un Assigner y un CVE sin relaciones. El objetivo de este script es comprobar la eliminación de un CVE sin relaciones.
9. Archivo “CP_14.sql” : script que reinicia la base de datos y registra dos CVEs. El CVE-1 tiene asociado un CPE que no tiene relaciones con sistemas operativos, y el CVE-2 , sin relaciones con CPEs. El objetivo de este script es comprobar la conservación de vulnerabilidades sin relaciones con CPEs o con CPEs que no estén asociados a sistemas operativos, debido a que no ha pasado un año desde su última fecha de modificación.
10. Archivo “CP_15.sql” : script que reinicia la base de datos y registra dos CVEs. El CVE-1 tiene asociado un CPE que tiene una relación con un sistemas operativo, y el CVE-2 tiene asociado un CPE si relaciones con sistemas operativos cuya fecha de última modificación es de menos de 10 días. El objetivo de este script es comprobar la conservación de vulnerabilidades con CPEs que están asociados a sistemas operativos, y con CPEs que hace menos de 10 días que no están asociados a sistemas operativos.

10.4.- CASOS DE PRUEBA

En este apartado se procederá a determinar casos de prueba que abarquen todas las situaciones de prueba diseñadas en el apartado anterior.

Para cada caso de prueba se detallará el objetivo con el cual se realiza, las situaciones de prueba que cubre, las entradas con las que se realizará y tanto las salida que se espera obtener como la obtenida en su realización.

Los casos de prueba cuya salida obtenida no concuerde con la salida esperada se remarcarán de rojo como fallos.

Con el objetivo de poder replicar los casos de prueba se proporcionan varias archivos para replicar los estados iniciales de la base de datos para cada caso.

Para poder implementar los casos de prueba se debe ejecutar en primer lugar el archivo "CreateDataBase.sql" para crear la base de datos y a continuación, en función del caso de prueba, sus determinadas entradas.

Identificador	Objetivo	Situaciones de prueba que cubre	Entradas	Salida esperada	Salida obtenida
1	Comprobar la carga inicial de información en la base de datos, la cual no tiene ningún registro almacenado, por lo que no dispone de fecha de última modificación para añadir un filtro a partir del cual obtener registros. Además se comprobará la creación de una url automática en función de la información que haya en la base de datos y el envío de varias peticiones para obtener todos los datos, debido a que el número de registros totales de la petición supera el número máximo de elementos de una respuesta, 2000.	1.1.1 2.1 3.2.2 11.1	Parámetro del programa Fuente: NVD Ejecutar en base de datos: ResetDataBase.sql	Se han almacenado correctamente todos los registros disponibles de NVD de manera eficiente sin afectar en el rendimiento del proceso principal	Se han almacenado correctamente todos los registros disponibles de NVD de manera eficiente sin afectar en el rendimiento del proceso principal

Identificador	Objetivo	Situaciones de prueba que cubre	Entradas	Salida esperada	Salida obtenida
2	Comprobar la carga de información relativa a un único CVE que ya está almacenado en la base de datos. Tanto el Assigner, como las referencias, puntuaciones, CPEs y CWEs están ya registrados y relacionados con el CVE en la base de datos. Además se comprobará la introducción de una url determinada, seleccionando como fuente NVD	2.2 3.2.1. 4.2 5.2 6.2.1.1.1 7.2.1.1.1 8.2.1.1 9.2.2 10.1 1.2	Parámetro del programa Url: https://services.nvd.nist.gov/rest/json/cve/1.0/CVE-2020-7668?addOns=dictionaryCpes Fuente: NVD Ejecutar en base de datos: CP_2.sql	Actualiza toda la información en todas las tablas de la base de datos relacionada con el CVE.	Actualiza toda la información en todas las tablas de la base de datos relacionada con el CVE, excepto para las referencias, donde borra el registro y luego lo crea de nuevo, en vez de actualizarlo
3	Comprobar la carga de información relativa a un único CVE, el cual no está almacenado en la base de datos. Tanto el Assigner, como las referencias y puntuaciones no están registrados. En cambio, los CPE y CWE están ya registrados pero no relacionados con el CVE en la base de datos	4.1 5.1 6.2.1.1.2 7.2.1.1.2 8.2.1.2 10.2	Parámetro del programa Url: https://services.nvd.nist.gov/rest/json/cve/1.0/CVE-2020-7668?addOns=dictionaryCpes Fuente: NVD Ejecutar en base de datos: CP_3.sql	Crea registros para toda la nueva información y actualiza la ya existente en todas las tablas de la base de datos.	Crea registros para toda la nueva información y actualiza la ya existente en todas las tablas de la base de datos.

Identificador	Objetivo	Situaciones de prueba que cubre	Entradas	Salida esperada	Salida obtenida
4	Comprobar el comportamiento del programa al realizar una petición cuya respuesta no tiene ningún registro	3.1	Parámetro del programa Url: https://services.nvd.nist.gov/rest/json/cves/1.0/?resultsPerPage=1&apikey=be00d657-d463-4892-8885-b6267a72af0c&modStartDate=2000-01-01T00:00:00:000 Z&modEndDate=2000-01-01T17:35:00:000 Z Fuente: NVD Ejecutar en base de datos: ResetDataBase.sql	No se realiza ninguna acción sobre la base de datos	No se realiza ninguna acción sobre la base de datos
5	Comprobar la carga de la información relativa a un CVE, el cual no está registrado en la base de datos. Tanto el Assigner, como las referencias, puntuaciones, y su único CPE y CWE no están registrados.	6.2.1.2 7.2.1.2	Parámetro del programa Url: https://services.nvd.nist.gov/rest/json/cve/1.0/CVE-2020-7668?addOns=dictionaryCpes Fuente: NVD Ejecutar en base de datos: ResetDataBase.sql	Se almacena toda la información correctamente en la base de datos	Se almacena toda la información correctamente en la base de datos

Identificador	Objetivo	Situaciones de prueba que cubre	Entradas	Salida esperada	Salida obtenida
6	Comprobar la carga de la información relativa a un CVE, el cual no tiene CPEs, ni referencias asociadas. Únicamente tiene una única puntuación de severidad, un CWE y su Assigner	7.1 8.1 9.2.1	Parámetro del programa Url: https://services.nvd.nist.gov/rest/json/cve/1.0/CVE-1999-0935?addOns=dictionaryCpes Fuente: NVD Ejecutar en base de datos: ResetDataBase.sql	Se almacena correctamente la información del CVE, Assigner, CWE y puntuación	Se almacena correctamente la información del CVE, Assigner, CWE y puntuación
7	Comprobar la carga de la información relativa a un CPE el cual no tiene ni puntuaciones, ni CWEs asociados.	6.1 9.1	Parámetro del programa Url: https://services.nvd.nist.gov/rest/json/cve/1.0/CVE-2022-20207?addOns=dictionaryCpes Fuente: NVD Ejecutar en base de datos: ResetDataBase.sql	Se almacena correctamente toda la información del CVE	Se almacena correctamente toda la información del CVE

Identificador	Objetivo	Situaciones de prueba que cubre	Entradas	Salida esperada	Salida obtenida
8	Comprobar la introducción de una fuente que no esté registrada en la base de datos	11.2	Fuente: MITRE	No se realiza ninguna petición y se guarda en el log del programa un aviso	Se acaba el proceso debido a una excepción no controlada
9	Comprobar la obtención de una url automática cuando haya un registro con fecha de última modificación	1.1.2	Parámetro del programa Fuente: NVD Ejecutar en base de datos: CP_9.sql	Se crea correctamente y realiza la petición con dicho filtro	Se crea correctamente y realiza la petición con dicho filtro
10	Comprobar la eliminación de dos registros de CVE. Un CVE que no tiene relaciones con CPEs, y otro que sí la tiene, pero cuyo CPE no está asociado a ningún sistema operativo	12.2.1 13.2.1 14.2.2 15.2.1 16.2.1 17.2.1 18.2.1 19.2.1 20.1.2.1.1 21.1.2.1.1 22.1 23.1	Parámetro del programa Fuente: NVD Url: https://services.nvd.nist.gov/rest/json/cves/1.0/?resultsPerPage=2&apikey=be00d657-d463-4892-8885-b6267a72af0c&modStartDate=2022-01-01T00:00:00:000 Z&modEndDate=2022-01-01T00:00:01:000 Z Ejecutar en base de datos: CP_10.sql	Elimina los registros para la vulnerabilidad CVE_1 y CVE_2, junto con toda la información asociada a ellas, excepto el CWE y el Assigner	Elimina los registros para la vulnerabilidad CVE_1 y CVE_2, junto con toda la información asociada a ellas, excepto el CWE y el Assigner

Identificador	Objetivo	Situaciones de prueba que cubre	Entradas	Salida esperada	Salida obtenida
11	Comprobar la eliminación de un Assigner y un CWE sin relaciones, que llevan más de 10 días sin modificar. Además se comprobará como no se purgan una vulnerabilidad sin asociaciones con CPEs y otra con CPEs sin relaciones con sistemas operativos, ya que no ha pasado un año desde su fecha de publicación.	14.1 19.1 20.1.2.1.2 21.1.2.1.2 22.2 23.2	Parámetro del programa Fuente: NVD Url: https://services.nvd.nist.gov/rest/json/cves/1.0/?resultsPerPage=2&apikey=be00d657-d463-4892-8885-b6267a72af0c&modStartDate=2022-01-01T00:00:00:000 Z&modEndDate=2022-01-01T00:00:01:000 Z Ejecutar en base de datos: CP_11.sql	Elimina los registros para el Assigner cuyo valor para el atributo name es ‘prueba’ y el CWE cuyo valor para el atributo identifer es ‘CWE-prueba’. Conserva ambos registros de CVEs y su información asociada en otras tablas	Elimina los registros para el Assigners cuyo valor para el atributo name es ‘prueba’ y el CWE cuyo valor para el atributo identifer es ‘CWE-prueba’. Conserva ambos registros de CVEs y su información asociada en otras tablas
12	Comprobar el comportamiento del proceso de purga ante una base de datos sin registros	12.1 13.1 20.1.1 21.1.1	Parámetro del programa Fuente: NVD Url: https://services.nvd.nist.gov/rest/json/cves/1.0/?resultsPerPage=2&apikey=be00d657-d463-4892-8885-b6267a72af0c&modStartDate=2022-01-01T00:00:00:000 Z&modEndDate=2022-01-01T00:00:01:000 Z Ejecutar en base de datos: ResetDataBase.sql	No realiza ninguna acción sobre la base de datos y el programa finaliza correctamente	No realiza ninguna acción sobre la base de datos y el programa finaliza correctamente

Identificador	Objetivo	Situaciones de prueba que cubre	Entradas	Salida esperada	Salida obtenida
13	Comprobar la eliminación de un CVE sin más información adicional en otra tabla de la base de datos	14.2.1 15.1 16.1 17.1 18.1	Parámetro del programa Fuente: NVD Url: https://services.nvd.nist.gov/rest/json/cves/1.0/?resultsPerPage=2&apikey=be00d657-d463-4892-8885-b6267a72af0c&modStartDate=2022-01-01T00:00:00:000 Z&modEndDate=2022-01-01T00:00:01:000 Z Ejecutar en base de datos: CP_13.sql	Solo elimina el registro de CVE, conservando en base de datos su Assigner.	Solo elimina el registro de CVE, conservando en base de datos su Assigner.
14	Comprobar como el proceso de purga no borra ningún registro de CVEs, ya que el CVE-1, sin relaciones con CPEs, y el CVE-2, con relaciones con CPEs, los cuales no tienen asociaciones con sistemas operativos, hace menos de un año que se modificaron por última vez	22.3 23.3	Parámetro del programa Fuente: NVD Url: https://services.nvd.nist.gov/rest/json/cves/1.0/?resultsPerPage=2&apikey=be00d657-d463-4892-8885-b6267a72af0c&modStartDate=2022-01-01T00:00:00:000 Z&modEndDate=2022-01-01T00:00:01:000 Z Ejecutar en base de datos: CP_14.sql	No se borra ningún registro de base de datos	No se borra ningún registro de base de datos

Identificador	Objetivo	Situaciones de prueba que cubre	Entradas	Salida esperada	Salida obtenida
15	Comprobar como el proceso de purga no borra ningún registro de CVEs, ya que el CVE-1 tiene un CPE que sí está asociado a un sistema operativo y el CVE-2 tiene CPEs que han dejado de relacionarse con sistemas operativos hace menos de 10 días. Los CVEs no se borrarán aunque sus fechas de publicación y modificación hayan sucedido hace más de un año.	23.4 23.5	Parámetro del programa Fuente: NVD Url: https://services.nvd.nist.gov/rest/json/cves/1.0/?resultsPerPage=2&apikey=be00d657-d463-4892-8885-b6267a72af0c&modStartDate=2022-01-01T00:00:00:000 Z&modEndDate=2022-01-01T00:00:01:000 Z Ejecutar en base de datos: CP_15.sql	No se borra ningún registro de base de datos	No se borra ningún registro de base de datos

10.5.- FALLOS DETECTADOS

Tras realizar los casos de prueba detallados en el apartado anterior, se procederá a realizar un análisis de los fallos que se han encontrado, marcados en rojo. Todos los fallos detectados se han corregido y testeado en la versión final entregada.

- Caso de prueba 2:
  Con el objetivo de comprobar el registro y modificación de un único CVE que ya está almacenado en la base de datos, cuyo Assigner, referencias, puntuaciones, CPEs y CWEs asociados también están ya registrados y relacionados con el CVE, se ha detectado un comportamiento incorrecto a la hora de almacenar la información de las referencias.

  En la tabla panVulnerabilitiesReferences, encargada de almacenar la información relativa a las referencias asociadas a una determinada vulnerabilidad, en vez de actualizar un registro con la información nueva, se borraba el registro antiguo con la información obsoleta y se creaba uno nuevo.

Este comportamiento ocurría debido a que se realizaba incorrectamente la comprobación de la existencia de un registro en dicha tabla, utilizando el atributo url, en vez del atributo name de la referencia. Por ello, tras no encontrar el registro, se creaba uno nuevo y se borraba el antiguo.

Este fallo repercute en la eficiencia del programa, añadiendo más peticiones sobre base de datos, y en la integridad referencial, ya que se borra un registro que puede estar asociado con otra entidad, sin tener en cuenta el orden de borrado.

- Caso de prueba 8:
  Con el objetivo de comprobar la introducción de una fuente que no esté registrada en la base de datos, se ha detectado un fallo, debido a que en el proceso de desarrollo no se tuvo en cuenta el requisito *R.F.1.1 – Selección de una fuente de información*, el cual determinaba que el usuario podrá seleccionar la fuente de información a la que se quiere realizar las peticiones, por lo que no se consideró dicha situación.

10.6.- RESUMEN EJECUTIVO

Tras la realización de los casos de prueba se han encontrado los siguientes fallos, ordenados en función de su gravedad:

1. Fallo al modificar un registro de la tabla panVulnerabilitiesReferences, la cual almacena referencias asociadas a una determinada vulnerabilidad. Caso de prueba 2.
2. Fallo al no manejar la introducción de fuentes de información que no estén registradas en base de datos. Caso de prueba 8

10.7.- OTRAS PRUEBAS

Las pruebas relacionadas con los requisitos no funcionales, en cuanto a rendimiento y continuidad del negocio, encargadas de comprobar que se cumplen y satisfacen dichas necesidades, están en desarrollo aunque no se incluyan en este proyecto.

Estas pruebas requieren un entorno real de un producto que se encuentra en producción dando servicio a sus clientes, por lo que se debe tener precaución en cuanto al impacto que generan dichas pruebas en los demás componentes que se encuentran en funcionamiento.

Por estos motivos, dichas pruebas se encuentran en desarrollo, pero debido a la naturaleza de ProactivaNET , se requiere mayor tiempo y precaución, con el fin de no afectar a los servicios prestados a sus clientes.

11. Planificación.

La fase de planificación consiste en la organización de las tareas que componen el proyecto, de manera que se define su duración y orden de ejecución. Como resultado, se obtiene un plan de trabajo inicial elaborado a partir de la programación y estructuración de estas actividades, de manera que se identifican claramente tanto las relaciones temporales, como el calendario o los momentos en los que deben desarrollarse cada una de ellas.

Por lo tanto, el desarrollo de la planificación al comienzo de un proyecto permite determinar la organización y plazos de este, para facilitar la definición del alcance, resultado en función de la combinación de coste, calidad y tiempo.

El desarrollo de este proyecto se ha realizado siguiendo un modelo en cascada o modelo Waterfall. Este modelo, dentro de las metodologías predictivas, se basa en un desarrollo secuencial, es decir, la ejecución de varias etapas que se van realizando una tras otra en un orden determinado. Esta técnica, aunque aporta poca flexibilidad frente a los cambios a diferencia de las metodologías ágiles, permite realizar un seguimiento estricto de cada fase del proyecto, lo que favorece la corrección y monitorización de los entregables que se van desarrollando durante cada fase. Debido a esto, la planificación del proyecto es de vital importancia para definir los plazos de cada tarea, ya que los resultados de cada fase deben documentarse de forma exhaustiva.

Para la planificación de este proyecto se han definido las siguientes fases, correspondientes a procesos definidos en metodología Métrica V3, descritos a lo largo de este proyecto:

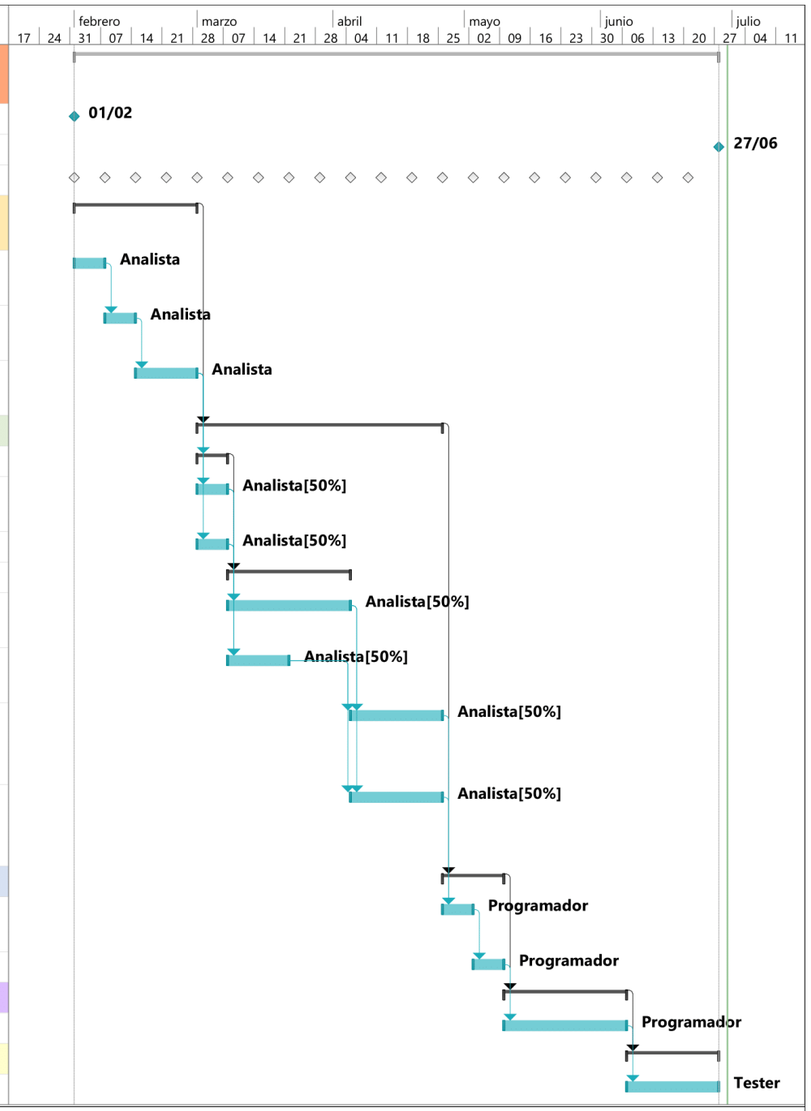
1. Estudio de Viabilidad del Sistema: proceso en el cual se realiza un análisis del proyecto con el objetivo de definir sus características generales en función de los requisitos del cliente y de la situación actual de la empresa, para proponer una solución a corto plazo. Este proceso dará inicio al desarrollo del proyecto y tendrá una duración de 20 días, comenzado el 01/02/22.
2. Análisis del Sistema: proceso en el cual se desarrollará la solución propuesta en el EVS, especificando detalladamente el sistema, con la finalidad de definir un conjunto de requisitos y modelos que cubran las necesidades de los usuarios. Este proceso tendrá una duración de 40 días, comenzando el 01/03/22.

3. Diseño del sistema: proceso en el cual se define detalladamente tanto la arquitectura del sistema como el entorno y los componentes que lo forman. Este proceso tendrá una duración de 10 días, comenzando el 26/04/22.
4. Desarrollo del software : proceso en el cual se lleva a cabo la construcción del sistema de información, a partir del conjunto de requisitos y especificaciones de los procesos de análisis y diseño anteriores. Este proceso tendrá una duración de 20 días, comenzando el 10/05/22.
5. Pruebas del sistema: proceso en el cual se realizará un plan de pruebas con el objetivo de descubrir fallos para mejorar la calidad del software. Este proceso tendrá una duración de 15 días, comenzando el 07/06/22.

La planificación del proyecto, dado que se ha utilizado una metodología predictiva, se representará con un gráfico de Gantt. Este diagrama facilita el control del presupuesto y los plazos del proyecto.

11.1.- DIAGRAMA DE GANTT

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
0		**Proyecto Integración NVD-ProactivaNET**	**105 días**	**mar 01/02/22**	**lun 27/06/22**		
1		Comienzo del proyecto	0 días	mar 01/02/22	mar 01/02/22		
2		Fin del proyecto	0 días	lun 27/06/22	lun 27/06/22		
3		**Reunión de seguimiento**	**100 días**	**mar 01/02/22**	**mar 21/06/22**		
25		**Estudio de Viabilidad del Sistema**	**20 días**	**mar 01/02/22**	**lun 28/02/22**		
26		Establecimiento del alcance del Sistema	5 días	mar 01/02/22	lun 07/02/22		Analista
27		Definición de Requisitos de Usuario	5 días	mar 08/02/22	lun 14/02/22	26	Analista
28		Estudio y Valoración de las alternativas	10 días	mar 15/02/22	lun 28/02/22	27	Analista
29		**Análisis del Sistema**	**40 días**	**mar 01/03/22**	**lun 25/04/22**	**25**	
30		**Definición del sistema**	**5 días**	**mar 01/03/22**	**lun 07/03/22**	**28**	
31		Descripción de los actores	5 días	mar 01/03/22	lun 07/03/22	28	Analista[50%]
32		División en subsistemas	5 días	mar 01/03/22	lun 07/03/22	28	Analista[50%]
33		**Análisis de clases**	**20 días**	**mar 08/03/22**	**lun 04/04/22**	**30**	
34		Definición de Modelo de dominio	20 días	mar 08/03/22	lun 04/04/22	31;32	Analista[50%]
35		Definición de diccionario de datos	10 días	mar 08/03/22	lun 21/03/22	31;32	Analista[50%]
36		Definición de modelo dinámico - Proceso de descarga de datos	15 días	mar 05/04/22	lun 25/04/22	35;34	Analista[50%]
37		Definición de modelo dinámico - Proceso de purga	15 días	mar 05/04/22	lun 25/04/22	35;34	Analista[50%]
38		**Diseño del sistema**	**10 días**	**mar 26/04/22**	**lun 09/05/22**	**29**	
39		Diseño de la arquitectura del sistema	5 días	mar 26/04/22	lun 02/05/22	36;37	Programador
40		Diseño físico de datos	5 días	mar 03/05/22	lun 09/05/22	39	Programador
41		**Desarrollo del software**	**20 días**	**mar 10/05/22**	**lun 06/06/22**	**38**	
42		Desarrollo del programa	20 días	mar 10/05/22	lun 06/06/22	40	Programador
43		**Pruebas del sistema**	**15 días**	**mar 07/06/22**	**lun 27/06/22**	**41**	
44		Pruebas del sistema	15 días	mar 07/06/22	lun 27/06/22	42	Tester



Proyecto: Proyecto Integració
Fecha: jue 30/06/22

Tarea | Resumen del proyecto | Tarea manual | solo el comienzo | Fecha límite
División | Tarea inactiva | solo duración | solo fin | Progreso
Hito | Hito inactivo | Informe de resumen manual | Tareas externas | Progreso manual
Resumen | Resumen inactivo | Resumen manual | Hito externo

Página 1

Figura 11.1.- Diagrama de Gantt

11.2.- DIAGRAMA PERT

Tras realizar el diagrama de Gantt, con el fin de proporcionar más información sobre las relaciones de precedencia entre las tareas definidas en la planificación y la flexibilidad de estas ante cambios de plazos, se ha realizado un diagrama Pert, donde se representa dicha información.

Este diagrama está compuesto de nodos rectangulares, los cuales representan las tareas del proyecto y de nodos hexagonales, los cuales representan hitos del proyecto. Un hito es un evento que tiene importancia para el desarrollo del proyecto. Estos nodos se relacionan mediante flechas que determinan el orden en el que se realizarán. La punta de flecha indica que la tarea señalada se realizará a continuación de la tarea de la cual sale la flecha. Una tarea no se podrá comenzar a desarrollarse hasta que todas sus predecesoras hayan finalizado. Finalmente, el diagrama estará formado por una red, constituida por nodos y caminos resultantes de las relaciones de precedencia.

En el diagrama se remarcará de rojo aquellos nodos que compongan el camino crítico del proyecto y de azul, los nodos no críticos. Denominamos camino crítico al recorrido de mayor duración del proyecto, el cual define el tiempo mínimo que se necesitará para llevar a cabo la ejecución de dicho proyecto. Por lo tanto, una reducción del plazo total de ejecución del proyecto solo será posible si se consigue reducir la duración de estas actividades críticas, ya que el tiempo de ejecución total del proyecto viene dado por la suma de las duraciones de las actividades que forman el camino crítico.

Debido a la metodología en cascada que se ha llevado a cabo, se puede observar en el diagrama que el camino crítico está compuesto de la mayor parte de las tareas del proyecto de manera secuencial. Por ello, este tipo de metodologías no aportan flexibilidad a la hora de reducir los plazos de un proyecto.

Únicamente la tarea “Definición del diccionario de datos” no es crítica, debido a que dispone de un periodo de tiempo de 5 días de retraso que se puede permitir hasta que finalice su actividad complementaria y paralela “Definición del modelo de dominio”.

A este periodo de tiempo se le denomina holgura y determina el aplazamiento que puede tolerar la ejecución de una actividad, sin que este afecte a la fecha de fin del proyecto.

El diagrama de PERT se añadirá junto con la documentación para poder examinarlo con más detalle.

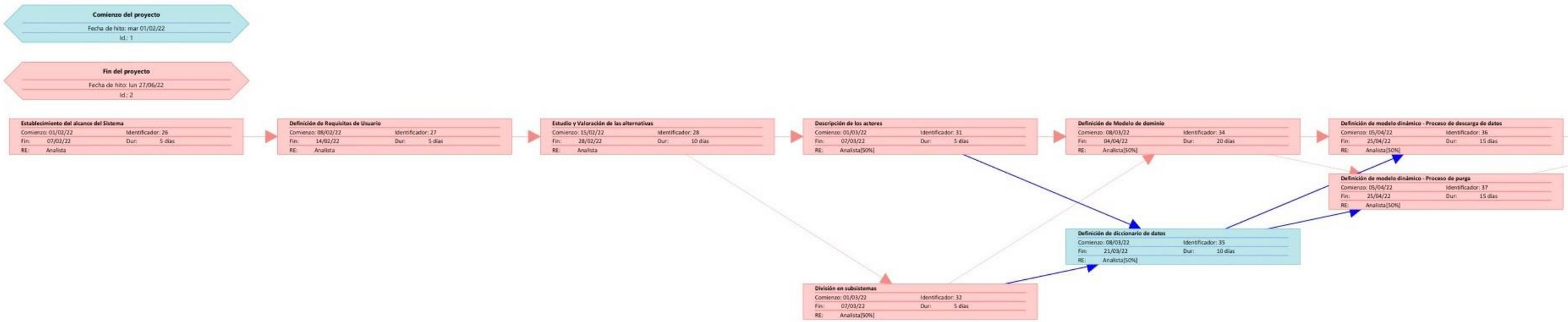


Figura 11.2.- Diagrama de PERT - Parte 1

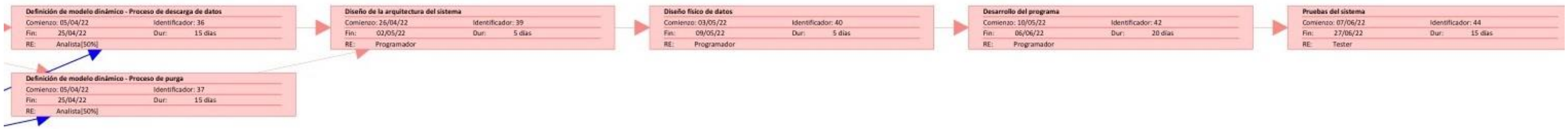


Figura 11.3.- Diagrama de PERT - Parte 2

11.3.- HISTOGRAMA DE RECURSOS

Finalmente, tras representar mediante el diagrama de Gantt la organización de las tareas y mediante el diagrama de Pert la flexibilidad de las actividades que conforman dicha planificación, se ha realizado un diagrama que representa la distribución de los recursos a lo largo del proyecto.

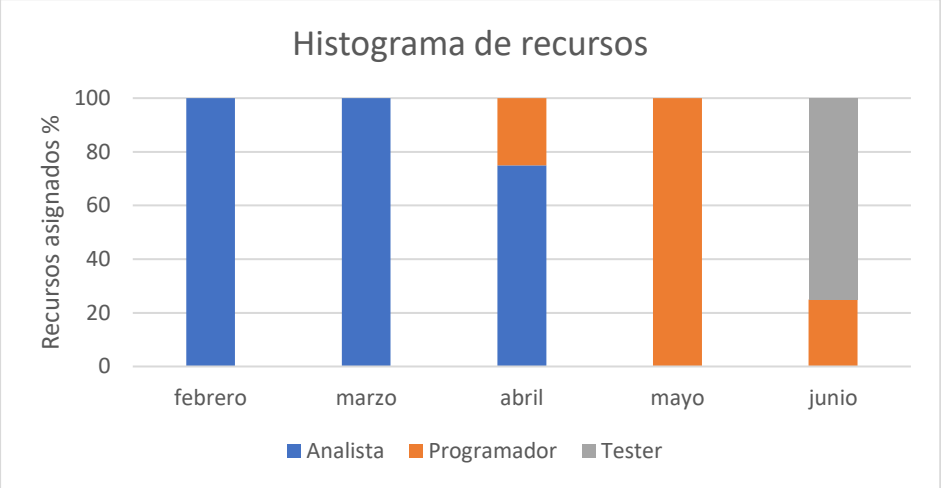


Figura 11.4.- Histograma de Recursos

Para la ejecución del proyecto se necesitarán tres perfiles de empleado distintos, en función de la tarea que se llevará a cabo. Las fases de "Estudio de Viabilidad" y "Análisis del sistema" las llevará a cabo un analista, perfil especializado en el estudio, descripción y especificación de sistemas de información. Por otra parte, en las fases de "Diseño del Sistema de Información" y "Desarrollo del software" se necesitará un perfil de programador, encargado de construir y codificar todo el software y la infraestructura del proyecto. Finalmente, en la fase de "Pruebas del sistema" se necesitará un perfil de tester, profesional cualificado para el desarrollo de un plan de pruebas siguiendo una serie de técnicas y metodologías para encontrar fallos, en búsqueda de mejorar la calidad del software. Estos perfiles se han ido obteniendo a lo largo de los estudios, por lo que todo el desarrollo del proyecto recaerá en el alumno.

A partir de esta distribución de recursos se puede concluir que para la ejecución del proyecto por parte del alumno, la distribución sería idealmente lineal, teniendo una ocupación completa desde el inicio hasta el final del proyecto. En otro caso esta distribución sería poco óptima, debido a los tiempo de inactividad de cada perfil, pudiendo ser mejor opción llevar a cabo el proyecto con una metodología ágil.

12. Presupuesto

En este apartado se desarrollará el presupuesto de ejecución del proyecto, en el cual se determina y justifica su coste económico.

El presupuesto estará dividido en capítulos, los cuales agruparán unidades de recursos presupuestadas, denominadas partidas. Dichas partidas se agruparán en función de la fase del proyecto a la que pertenezcan, con el fin de facilitar su comprensión.

La unidad monetaria empleada en la elaboración de los presupuestos es el euro y la unidad de trabajo es la hora.

12.1.- CAPÍTULO 1 - ESTUDIO DE VIABILIDAD DEL SISTEMA

En este apartado se describirá los costes de ejecución del estudio y análisis del proyecto, con el objetivo de definir sus características generales en función de los requisitos del cliente y de la situación actual de la empresa.

Id	Descripción	Unidad	Cantidad	Precio	Total
1	Analista	Hora	160	20,00 €	3200,00 €

Los costes del proceso “Estudio de Viabilidad del Sistema” ascienden a tres mil doscientos euros.

12.2.- CAPÍTULO 2 - ANÁLISIS DEL SISTEMA

En este apartado se describirá los costes del desarrollo de una especificación detallada del sistema, con la finalidad de definir un conjunto de requisitos y modelos que cubran las necesidades de los usuarios.

Id	Descripción	Unidad	Cantidad	Precio	Total
1	Analista	Hora	280	20,00 €	5600,00 €

Los costes del proceso “Análisis del Sistema” ascienden a cinco mil seiscientos euros.

12.3.- CAPÍTULO 3 – DISEÑO DEL SISTEMA

En este apartado se describirá los costes de definición tanto de la arquitectura del sistema como del entorno y los componente que lo forman.

Id	Descripción	Unidad	Cantidad	Precio	Total
1	Programador	Hora	80	30,00 €	2400,00 €

Los costes del proceso “Diseño del Sistema” ascienden a dos mil cuatrocientos euros.

12.4.- CAPÍTULO 4 – DESARROLLO DEL SOFTWARE

En este apartado se describirá los costes de construcción del sistema de información, a partir del conjunto de requisitos y especificaciones de los procesos de análisis y diseño

Id	Descripción	Unidad	Cantidad	Precio	Total
1	Programador	Hora	160	30,00 €	4800,00 €

Los costes de la fase “Desarrollo del software” ascienden a cuatro mil ochocientos euros.

12.5.- CAPÍTULO 5 – PRUEBAS DE SISTEMA

En este apartado se describirá los costes de realización de un plan de pruebas con el objetivo de descubrir fallos para mejorar la calidad del software.

Id	Descripción	Unidad	Cantidad	Precio	Total
1	Tester	Hora	120	30,00 €	3600,00 €

Los costes del proceso “Pruebas del Sistema” ascienden a tres mil seiscientos euros.

12.6.- PRESUPUESTO DE EJECUCIÓN

DESIGANCIÓN DE LAS OBRAS	TOTALES
CAPÍTULO 1 - Estudio de Viabilidad del Sistema	3.200,00 €
CAPÍTULO 2 - Análisis del Sistema	5.600,00 €
CAPÍTULO 3 - Diseño del sistema	2.400,00 €
CAPÍTULO 4 - Desarrollo del software	4.800,00 €
CAPÍTULO 5 - Pruebas del sistema	3.600,00 €
TOTAL	19.600,00 €
IVA 21%	4.116,00 €
TOTAL CON IVA	23.716,00 €

El total del presupuesto de ejecución asciende a veintitrés mil setecientos dieciséis euros.

13. Futuras ampliaciones.

La integración de vulnerabilidades relacionadas con aplicaciones y hardware no se abarcará en este proyecto, aunque su implementación seguirá el mismo proceso, una vez solventado el problema relacionado con la creación de CPEs para estos tipos de activos.

El NIST ,dentro de toda la información relacionada con una vulnerabilidad, proporciona una lista de identificadores que referencian las debilidades que causa. Estas debilidades son detectadas y registradas por el MITRE, una organización estadounidense relacionada con la seguridad informática que trabaja conjuntamente con el NIST.

Actualmente se encuentra en desarrollo una API del MITRE que permitirá, de la misma forma que la API de NVD, el establecimiento de una comunicación vía REST para la obtención de información relacionada con las debilidades, tomando como referencia el CWE (Common Weakness Enumeration), un sistema de categorías para las debilidades del software.

Por ello, en el desarrollo de este proyecto se tendrá en cuenta esta futura ampliación, almacenando la lista de CWEs (debilidades), para cada CVE (vulnerabilidad), aunque no se abarcará la integración de la información referente a esas debilidades, dado que aún no se dispone de la API proporcionada por el MITRE.

Por último, la ejecución de los procesos desde el *register* no se implementará durante el transcurso de este proyecto, debido a que esta actividad implica un proceso de prueba extenso para comprobar el impacto en términos de eficiencia en los demás componentes del sistema, teniendo que realizar dichas pruebas en la versión de ProactivaNET que se encuentra en producción para obtener resultados en un entorno con carga real, pudiendo afectar a la calidad de los servicios prestados a los clientes.

14. Aplicación instalable.

Los servidores de datos de ProactivaNET se encuentran alojados en la nube, distribuidos por la empresa Amazon a través de sus servicios de AWS. Para establecer una conexión con ellos se debe levantar una conexión VPN desde un dispositivo controlado que se encuentre dentro la red privada de la empresa.

Por este motivo, no se proporcionará una aplicación instalable, aunque se adjuntará a este documento el código fuente tanto de los procesos principales como de los scripts que se utilizan para el plan de prueba y los scripts que crean y modifican la base de datos.

En su defecto se realizará una demostración del comportamiento de los procesos integrados en la herramienta ProactivaNET, desde uno de estos dispositivos controlados integrados en la red de la empresa. En esta demostración se realizará una serie de ejemplos que demuestren el comportamiento de los procesos una vez integrados en el producto oficial de la empresa. De esta forma se podrá observar el resultado final que obtendrá el cliente, apreciando el valor que aporta la nueva información a los servicios de gestión de activos e inventariado que proporciona ProactivaNET.

15. Manual de configuración

El programa diseñado en este proyecto dispone de varios parámetros configurables, los cuales permiten cambiar algunos aspectos del funcionamiento de los procesos

Todos los parámetros configurables se consultan sobre la base de datos, almacenados en la tabla panVulnerabilitiesSettings. En esta tabla se almacenan los valores para dichos parámetros, por lo que en caso de modificar un valor, se debe modificar a través de la base de datos.

Estos valores son:

- timeCVE (date) : tiempo máximo que se conservará en base de datos una vulnerabilidad sin relaciones.
- timeCPE (date) : tiempo máximo que se conservará en base de datos un CPE sin relaciones en la base de datos.
- resultsPerPage (int) : número de registros obtenidos por petición.

Además, el programa permite seleccionar una fuente de información y una url para la obtención de unos datos concretos. En el caso de la selección de una fuente de información, se introducirá como argumento del programa el nombre de dicha fuente. Para ello, la fuente debe estar almacenada en base de datos en la tabla panVulnerabilitiesSources, donde el atributo name determinará el valor que habrá que introducir como argumento. En el caso de la url, si la url no concuerda con la estructura de petición de la fuente de información no se continuará con el proceso de descarga.

Para el seguimiento de la trazabilidad del programa, toda la información relevante en la ejecución de este se volcará a un archivo de log, para consultar el origen de una excepción o las fechas en las que se realizan los procesos. Este fichero en la siguiente ruta, CVERepositoryClient/EspiralMS.Proactivanet.CVEsRepositoryClient/bin/Debug/log., creándose uno nuevo para cada día de ejecución.

16. Conclusión

Tras la finalización de este trabajo, ProactivaNET, por medio del programa final resultante ha aportado mayor valor a su aplicación, proporcionando a sus clientes más información en el ámbito de la seguridad, respecto a las posibles vulnerabilidades y debilidades que afectan a sus dispositivos.

El alumno de la misma forma ha afianzado y adquirido nuevos conocimientos que a lo largo de sus estudios se han ido introduciendo en las siguientes asignaturas:

- Bases de datos
- Gestión de Tecnologías de la Información
- Seguridad
- Teoremas y Paradigmas de la programación
- Arquitectura de computadores
- Programación concurrente y paralela
- Sistemas de Información
- Ingeniería del software
- Proyectos
- Pruebas y Despliegue

Este trabajo es un compendio y resumen de todo el conocimiento, técnica , métodos y normativas aprendidos durante toda la trayectoria académica en las anteriores asignaturas y por tanto, es el resultado final consecuencia de aplicar todo lo aprendido.

17. Bibliografía.

[1] Grupo Espiral MS : https://www.grupoespiralms.com/

[2] Proactivanet : https://www.proactivanet.com/

[3] NIST (National Institute of Standards and Technology) . U.S. Department of Commerce: https://www.nist.gov/

[4] NVD (National Vulnerability Database), Página principal: https://nvd.nist.gov/general

[5] Uso de la API para peticiones de CVEs: https://nvd.nist.gov/developers/vulnerabilities

[6] Uso de la API para peticiones de CPEs: https://nvd.nist.gov/developers/productsç

[7] Detección de vulnerabilidades en los activos inventariados en la herramienta Inventario de ProactivaNET Integración entre ProactivaNET, NVD e INCIBE , Autor: José Carlos Díaz Ruénez

[8] Métrica v.3, Portal de la Administración Electrónica del Gobierno de España: https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html