

Universidad de las Ciencias Informáticas

Facultad 2



Título: “Sistema para la ejecución de pruebas de seguridad y procesamiento de los resultados en una arquitectura distribuida.”

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Yamila Izquierdo Vázquez

Erenio Ramos Medina

Tutor(es): Ing. Edgar González Blanco

Co-tutor: Ing. Rogfel Thompson Martínez

La Habana, Cuba
Junio, 2012

Pensamiento

Las ideas no duran mucho, hay que hacer algo con ellas.

Anónimo.

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos al Centro de Telemática de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Yamila Izquierdo Vazquez

Firma del Autor:

Erenio Ramos Medina

Firma del Autor:

Ing. Rogfel Thompson Martínez

Firma del Co-Tutor:

Ing. Edgar González Blanco

Firma del Tutor:

DATOS DE CONTACTO

Yamila Izquierdo Vazquez

Correo: yivazquez@estudiantes.uci.cu

Ciudad de La Habana, Cuba

Erenio Ramos Medina

Correo: ermedina@estudiantes.uci.cu

Ciudad de La Habana, Cuba

Ing. Edgar González Blanco

Correo: egblanco@uci.cu

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

Ing. Rogfel Thompson Martínez

Correo: rthompson@uci.cu

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

AGRADECIMIENTOS

A nuestras madres y padres por darnos la vida, por tantos sacrificios, por su infinito amor. A nuestras familias y amigos por su comprensión su apoyo y su invaluable ayuda, sobre todo en los momentos más difíciles de la realización de la tesis. A nuestros tutores Edgar Gonzales y Rogfel Thompson, por su guía, su sabiduría y su apoyo. A nuestros compañeros del laboratorio por sus oportunos consejos y su ayuda. A todos los que de alguna forma han contribuido a la culminación de este trabajo; a todos gracias.

DEDICATORIA

Yamila Izquierdo Vázquez: A mis padres.

Erenio Ramos Medina: A mis viejos que son mis héroes.

RESUMEN

En la presente investigación se propone la implementación de un sistema capaz de ejecutar herramientas de pruebas de seguridad en aplicaciones de forma distribuida, así como estandarizar y almacenar en una base de datos la información de los reportes generados, los cuales pueden tener diferentes formatos. Dicho sistema se integrará a un entorno *web (front end)* para así conformar una plataforma de pruebas de seguridad en aplicaciones informáticas. Desde el entorno *web* se realizarán solicitudes de pruebas, que serán enviadas hacia un servidor que alojará el sistema Bambú (sistema distribuidor de tareas). Una vez recibida la solicitud de ejecución de la prueba, se distribuye esta hacia los agentes que alojarán las herramientas que realizan la prueba especificada. Una vez realizado el envío hacia el agente se ejecutará la herramienta de pruebas de seguridad, generándose archivos de reporte, de los cuales se extraerá la información relevante, mediante la ejecución del normalizador de información haciendo uso de expresiones regulares definidas en los archivos XML de estandarización. Este resultado de la prueba será enviado hacia el servidor y almacenado en una base de datos. Luego se le enviará una notificación al entorno *web* para que muestre la información del resultado de la prueba realizada.

PALABRAS CLAVE

Servidor, Agente, Capa, Entorno Web, Distribuir, Normalizador

TABLA DE CONTENIDOS

DECLARACIÓN DE AUTORÍA	I
DATOS DE CONTACTO.....	I
AGRADECIMIENTOS.....	1
DEDICATORIA	2
RESUMEN.....	3
TABLA DE CONTENIDOS.....	4
ÍNDICE DE FIGURAS.....	7
ÍNDICE DE TABLAS.....	8
INTRODUCCIÓN.....	9
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA	13
Introducción	13
1.1. Marco Conceptual.....	13
1.2. Compañías que desarrollan y brindan servicios de seguridad informática.....	15
1.2.1. NGSsecure.....	15
1.2.2. BASE4 Security	15
1.2.3. Trustware.....	16
1.3. Exploración de soluciones posibles a reutilizar	16
1.3.1. Open Source Security Information Management (OSSIM)	16
1.3.2. Sistema para Distribución de Tareas (Bambú)	17
1.4. Tecnologías y herramientas utilizadas	18
1.4.1. Metodología de desarrollo	19
1.4.2. Lenguaje de modelado: Lenguaje Unificado de Modelado (UML)	19
1.4.3. Herramienta case: Visual Paradigm.....	19
1.4.4. Herramienta de Desarrollo: Eclipse.....	20
1.4.5. Sistema Gestor de Bases de Datos: PostgreSQL	20
1.4.6. Administrador de Bases de Datos: PgAdmin.....	20
1.4.7. Lenguaje de programación: Python	20
1.4.8. SQLite.....	21
1.4.9. Middleware	21
1.4.10. ICE	22
1.4.11. Glacier2.....	22
1.4.12. WebSocket.....	22
1.5. Análisis de posibles soluciones estandarización de reportes	22
1.5.1. Parser.....	23
1.5.2. Pyparsing.....	23
1.5.3. Capa de abstracción de pyparsing.....	23
1.5.4. Expresiones regulares.....	23
1.5.5. XML con expresiones regulares	24
1.6. Conclusiones	24
CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.....	25

2.1. Objeto de automatización	25
2.2. Propuesta de solución	25
2.3. Modelo del dominio	26
2.3.1. Conceptos:	26
2.4. Requisitos funcionales	27
2.5. Requisitos no funcionales	28
2.6. Modelo de Caso de uso del sistema	29
2.6.1. Actores del sistema.....	29
2.6.2. Diagrama de Casos de uso del sistema.....	30
2.6.3. Descripción de casos de uso del sistema	30
CAPÍTULO 3: DISEÑO DEL SISTEMA.....	35
3.1. Modelo de diseño	35
3.1.1. Diagrama de paquetes	35
3.1.2. Diagrama de clases	36
3.1.3. Cliente	37
3.1.4. Servidor.....	38
3.2. Arquitectura del sistema.....	39
3.4.1. Arquitectura n-capas.....	42
3.4.2. Descripción de las capas	42
3.4.3. Patrones de diseño	42
3.4.3.1. Experto	42
3.4.3.2. Alta cohesión.....	43
3.4.3.3. Bajo acoplamiento.....	43
3.4.3.4. Singleton	43
3.4.3.5. Creador.....	43
3.5. Modelo de datos	44
3.5.1. Modelo lógico de datos	44
3.5.2. Modelo físico de la base de datos	45
3.6. Conclusiones	45
CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA.....	46
4.1. Introducción.....	46
4.2. Extracción de información mediante XML y expresiones regulares.....	46
4.3. Diagrama de despliegue	48
4.4. Diagrama de componentes.....	49
4.4.1. Diagrama general del sistema	49
4.4.2. Diagrama de componentes del servidor paquete Acceso a Datos	50
4.4.3. Diagrama de componentes del servidor paquete Dominio	52
4.4.4. Diagrama de componentes del servidor paquete ImplementClass	52
4.4.5. Diagrama de componentes del servidor paquete Balancer	53
4.4.6. Diagrama de componentes del cliente paquete ImplementClass	53
4.4.7. Diagrama de componentes del cliente paquete Standard.....	54
4.5. Pruebas del sistema	55

4.5.1. Estrategia de Prueba	55
4.5.2. Pruebas de Unidad	55
4.5.3. Pruebas de integración	56
4.6. Conclusiones	57
CONCLUSIONES GENERALES	58
RECOMENDACIONES	59
REFERENCIA BIBLIOGRÁFICA	60
BIBLIOGRAFÍA.....	62
ANEXOS	64
ANEXO 2 DESCRIPCIÓN DEL MODELO LÓGICO DE LA BASE DE DATOS.	66
ANEXO 3 DESCRIPCIÓN DEL MODELO FÍSICO DE LA BASE DE DATOS.....	68
GLOSARIO.....	74

ÍNDICE DE FIGURAS

Fig. 1: Diagrama de dominio	26
Fig. 2: Diagrama de casos de uso del sistema	30
Fig. 3: Diagrama de paquetes.....	36
Fig. 4: Diagrama de clases cliente	37
Fig. 5: Diagrama de clases servidor	38
Fig. 6: Arquitectura de la Plataforma de Pruebas de Seguridad en los software.....	39
Fig. 7: Diagrama del modelo lógico de datos	44
Fig. 8: Diagrama del modelo físico de datos.....	¡Error! Marcador no definido.
Fig. 9: Diagrama de Despliegue de la Plataforma.....	49
Fig. 10: Diagrama de componentes general del sistema	50
Fig. 11: Diagrama de componentes del servidor paquete Acceso a Datos.....	51
Fig. 12: Diagrama de componentes del servidor paquete Dominio.....	52
Fig. 13: Diagrama de componentes del servidor paquete ImplementClass.....	53
Fig. 14: Diagrama de componentes del servidor paquete Balancer.....	53
Fig. 15: Diagrama de componentes del cliente paquete ImplementClass.....	54
Fig. 16: Diagrama de componentes del cliente paquete Standard.....	54
Fig. 17: Ejemplo de prueba de unidad.....	56
Fig. 18: Resultado de las pruebas de unidad.....	56

ÍNDICE DE TABLAS

Tabla 1: Descripción de los actores del sistema	30
Tabla 2: Descripción del caso de uso distribuir tarea en el servidor	31
Tabla 3: Descripción del caso de uso procesar el contenido del reporte	32
Tabla 4: Descripción del caso de uso enviar resultado de prueba	33
Tabla 5: Descripción del caso de uso Notificar culminación de la prueba	33
Tabla 6: Tabla de pruebas de integración	57
Tabla 7: Descripción de del diagrama lógico de la Base de Datos.	67
Tabla 8: Descripción del modelo físico de la Base de Datos.....	¡Error! Marcador no definido.

INTRODUCCIÓN

El mundo se encuentra ante una nueva era, conocida como la era de la información siendo resultado del acelerado avance de las Tecnologías de la Informática y las Comunicaciones (TIC), lo que se traduce en la aparición de diversos sistemas digitales encargados de gestionar grandes volúmenes de información. Aunque las TIC tienen una visión integral, pues es conveniente que estén presentes en todas las actividades y procesos que se realicen en las instituciones, como herramienta necesaria para aumentar la productividad y organización del trabajo. Se debe tener en cuenta que con la aparición de la red de redes, la posibilidad de interconectarse y a medida que la información es más creciente e ilimitada surgen nuevas amenazas de seguridad para los sistemas informáticos siendo relevante entonces, priorizar y controlar el acceso a la documentación manejada en las empresas para lograr el aseguramiento de la confidencialidad, disponibilidad e integridad de la información. (1)

Cuba, al igual que el resto del mundo, se ha manifestado en los últimos años por el desarrollo de la actividad informática. Aunque aún es incipiente la industria del software cubana se lleva a cabo un proceso de informatización, del cual debe entenderse que no es más que una forma de gestionar la información y el conocimiento.

Ante el creciente número de amenazas que aparecen diariamente en las redes se hace necesario velar cada día más por la seguridad informática. Por tanto es necesario en la naciente industria del software cubano, tener en cuenta la seguridad de las aplicaciones desarrolladas.

La Universidad de las Ciencias Informáticas (UCI), como caso particular, está insertada actualmente en el proceso de producción de *software* y servicios informáticos. La misma se encuentra organizada a través de una red de centros que desarrollan *software* en diferentes ramas de la informática para la exportación y la necesaria informatización de la sociedad cubana.

La UCI como entidad desarrolladora de *software* tiene como responsabilidad el fortalecimiento de la seguridad de los sistemas antes de ser entregados al cliente. En la UCI se trabaja por consolidar los logros alcanzados, por erradicar las deficiencias que aún existen y por profundizar en el tema de la seguridad en las aplicaciones informáticas, con la intención de eliminar oportunamente cada nueva brecha.

En el Centro de Telemática (TLM), específicamente en el proyecto Laboratorio de Seguridad Informática (LabSI) se realizan una serie de pruebas con herramientas de *hacking* ético y análisis de código, que tienen como objetivo detectar problemas de seguridad en los productos de *software* desarrollados en la institución. LabSI concibe la idea de la creación de una plataforma para la ejecución de pruebas de seguridad en aplicaciones informáticas la cual contará con un *front end* y un *back-end*. El *front end* será el sistema encargado de la gestión de la información en la plataforma. Por su parte el *back-end* se ocupará de la ejecución de las herramientas de prueba de seguridad.

En la actualidad para la ejecución de las pruebas, cada especialista debe tener instaladas en su puesto de trabajo las herramientas que usa, teniendo que ejecutarlas de forma local. En ocasiones el laboratorio cuenta con estaciones de trabajo que poseen recursos computacionales disponibles, los cuales quedan sin explotar cuando no se encuentra un especialista realizando pruebas en estas, por lo que no se hace un uso óptimo de los recursos existentes.

Una vez terminada una prueba, se genera un reporte cuyo formato varía en dependencia de la herramienta usada, incluso una misma herramienta puede generar sus resultados en varios formatos. Los reportes generados deben ser minuciosamente examinados de forma manual por el especialista que realizó la prueba, para extraer de este la información relevante y así confeccionar un informe de la misma. Teniendo en cuenta que los reportes pueden ser extensos, con formatos diversos, en ocasiones poco entendibles y además suelen contener cierta cantidad de información no relevante para el proyecto LabSI, se hace evidente que el proceso resulta costoso en tiempo y esfuerzo.

Actualmente la información resultante de la ejecución de cada prueba solo queda en archivos de texto lo que dificulta la recuperación a posteriori de una determinada información contenida en dichos archivos. Otro problema asociado a la forma en que se almacenan los reportes generados por las herramientas es que no hay modo factible de relacionar los datos automáticamente con vista a obtener informaciones estadísticas que apoyen la toma de decisiones a partir de las pruebas realizadas.

Luego de hacer un análisis de la **problemática** se obtiene el siguiente **problema a resolver**: ¿Cómo solucionar las deficiencias del proceso de pruebas de seguridad informática del proyecto LabSI?

Como **objeto de estudio** de la presente investigación se plantea: el proceso de pruebas de seguridad informática. Se define como **campo de acción** el proceso de pruebas de seguridad en aplicaciones

informáticas de forma distribuida para proyecto LabSI.

Para dar solución al problema expuesto anteriormente se traza como **objetivo general**: desarrollar un sistema basado en una arquitectura distribuida para la ejecución de pruebas de seguridad, permitiendo además la estandarización y almacenamiento de los resultados.

Las siguientes **tareas de la investigación** darán cumplimiento al objetivo general:

- ✓ Estudio de las herramientas y tecnologías que tributen a la solución.
- ✓ Estudio de sistemas de ejecución de tareas distribuidas con vista a adoptar uno sobre el cual implementar la solución.
- ✓ Definición de los requisitos funcionales y no funcionales que debe cumplir el sistema.
- ✓ Diseño de un modelo representativo de la solución.
- ✓ Implementación de un conjunto de clases capaces de procesar y almacenar los distintos resultados de las herramientas de seguridad.
- ✓ Estudio de técnicas para analizar contenidos de archivos y extraer información de estos.
- ✓ Diseño de la base de datos que permita la persistencia organizada de la información.
- ✓ Análisis de los resultados de las herramientas de pruebas de penetración Nikto, Wapiti y Nmap, propuestas para ser integradas al sistema, con el objetivo de determinar la información relevante que brindan sus reportes.
- ✓ Realización de pruebas a la solución obtenida.
- ✓ Integración de la solución con el sistema *front end* para materializar así la plataforma pruebas de seguridad en aplicaciones informáticas del proyecto LabSI.

Para darle cumplimiento a la investigación se utilizaron los siguientes **métodos científicos de investigación**:

Analítico-Sintético

Análisis de la bibliografía disponible para realizar un estudio del estado del problema a resolver. Posibilita definir los conceptos principales y analizar otras soluciones existentes. Se sintetizan las principales características de las herramientas para el desarrollo del sistema y las ventajas del uso de las mismas. (3)

Histórico-Lógico

Se utiliza con el fin de conocer soluciones ya existentes, semejantes a la que se pretende realizar, tanto a nivel nacional como internacional. A través del mismo se logrará detectar todas las deficiencias y problemas que presentaban estas soluciones. (3)

El presente documento está estructurado en 4 capítulos:

Capítulo 1: Fundamentación teórica. Incluye un estudio del estado del arte del tema tratado, donde se muestran los principales conceptos manipulados en el transcurso de la investigación. Se hace una breve referencia a las herramientas, técnicas, metodología y *software* usados en el mundo para dar solución a problemas similares que apoyen a la construcción de la solución del problema que se enfrenta.

Capítulo 2: Características del sistema. Describe el modelado del negocio y se abordan las características del sistema. Se describe la solución propuesta, definiendo los requerimientos funcionales y no funcionales.

Capítulo 3: Diseño del sistema. Comienzo de una arquitectura que se ajuste a la solución del problema y sea flexible a cambios. Además muestra los diagramas de clases modelados para la solución así como el diseño del flujo de trabajo. Logrando transformar los requisitos del usuario a una especificación que describe cómo implementar el mismo. Todo lo anterior como resultado de la aplicación acertada de los patrones de diseño correctos.

Capítulo 4: Implementación y Prueba. Presenta los distintos componentes que conforman al producto y se realiza la validación del sistema a través de las pruebas de unidad y de integración.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En Cuba, específicamente en la UCI se realizan pruebas de seguridad informática para detectar vulnerabilidades en las aplicaciones desarrolladas. En el presente capítulo se abordan los principales conceptos referentes a la seguridad informática analizados a lo largo de la investigación. Se realiza además un análisis de los sistemas que existen actualmente a nivel internacional y nacional con similitud a los procesos ya mencionados que se pretenden automatizar en el proyecto LabSI y se realiza una breve descripción de las herramientas y tecnologías que se utilizan para dar cumplimiento al objetivo general de la presente investigación.

1.1. Marco Conceptual

Con la automatización de la sociedad moderna y de sus empresas aparecen diariamente amenazas para los activos informáticos, de esta forma parte la necesidad de la aplicación de medidas de Seguridad Informática, que no es más que un conjunto de acciones metodológicas y herramientas cuyo objetivo principal es proteger la información y por ende a los sistemas informáticos que la generan, la procesan, la transmiten y la almacenan, ante vulnerabilidades o amenazas internas y externas. (3)

De igual forma para el aseguramiento de que las políticas de seguridad son establecidas correctamente en las empresas o instituciones, son realizadas las Auditorías de Seguridad Informática que se encargan de verificar el control interno de la función informática. Asegurar a la alta dirección y al resto de las áreas de la empresa que la información que les llega es la necesaria en el momento oportuno, y es fiable, ya que les sirve de base para tomar decisiones importantes. Eliminar o reducir al máximo la posibilidad de pérdida de la información por fallos en los equipos, en los procesos o por una gestión inadecuada de los archivos de datos. Detectar y prevenir fraudes por manipulación de la información o por acceso de personas no autorizadas. (4)

No obstante a todas las medidas que se establecen de Seguridad Informática aún así puede que existan amenazas y vulnerabilidades en los sistemas de cómputo, las cuales pueden ser descubiertas a través de penetraciones controladas en los sistemas informáticos de una empresa, de la misma forma que lo haría un pirata informático pero de forma ética, con previa autorización por escrito; a lo que se le llama *Hacking ético*. El resultado es un informe donde se identifican los sistemas en los que se ha logrado penetrar y la información confidencial y/o secreta conseguida. (5)

Tipos de hacking ético

✓ Hacking Ético Externo Caja Blanca

Facilita información para poder realizar la intrusión (normalmente las direcciones IP a testar). Se analizan en profundidad y extensión todas las posibles brechas de seguridad al alcance de un atacante de los sistemas de comunicaciones sometidos a estudio. Opcionalmente, el ámbito de actuación se puede ampliar a máquinas no perimetrales. El resultado es un informe amplio y detallado de las vulnerabilidades, así como las recomendaciones para solucionar cada una de ellas. (5)

✓ Hacking Ético Externo Caja Negra

Es esencialmente lo mismo que en el de Caja Blanca con la dificultad añadida de que no se nos facilita ningún tipo de información inicial. (5)

✓ Hacking Ético Interno

El ámbito de esta auditoría es la red interna de la empresa, para hacer frente a la amenaza de intento de intrusión, bien por un empleado que pueda realizar un uso indebido o una persona con acceso a los sistemas o un *hacker* que hubiera conseguido penetrar en la red. Para este servicio se hace necesaria la presencia de nuestros especialistas en las instalaciones de la empresa que se va a auditar. El resultado es un informe amplio y detallado de las vulnerabilidades, así como las recomendaciones para solucionar cada una de ellas. (5)

✓ Hacking Ético de Aplicaciones Web

Se simulan los intentos de ataque reales a las vulnerabilidades de una o varias aplicaciones determinadas, como pueden ser: sistemas de comercio electrónico, de información, o de acceso a bases de datos. No es necesaria la entrega del código fuente de la aplicación. El resultado es un informe amplio y detallado de las vulnerabilidades, así como las recomendaciones para solucionar cada una de ellas. (5)

✓ Hacking Ético de Sistemas de Comunicaciones

En esta auditoría se analiza la seguridad de las comunicaciones tales como: redes de datos, *hardware* de red, comunicaciones de voz, fraude en telecomunicaciones (uso fraudulento de centralitas, telefonía pública, acceso no autorizado a Internet, redes de transmisión de datos por radio, etc.) principalmente para estudiar la disponibilidad de los sistemas, la posibilidad de una interceptación o introducción no

autorizada de información. El resultado es un informe amplio y detallado de las vulnerabilidades, así como las recomendaciones para solucionar cada una de ellas. (5)

1.2. Compañías que desarrollan y brindan servicios de seguridad informática

A continuación se realiza una descripción de algunas compañías que brindan y desarrollan servicios de seguridad informática a nivel mundial.

1.2.1. NGSsecure

Es una compañía independiente proveedora de pruebas de penetración y experta en servicios de seguridad. (6)

Algunos servicios brindados

- ✓ *PenetrationTesting* (Pruebas de Penetración) con este servicio la empresa ayuda a sus clientes a comprender su exposición ante las amenazas.
- ✓ *Information Forencics*(Informática Forense) brinda investigación forense y asesoramiento llevado a cabo en línea con las directrices de la ACPO¹
- ✓ *Software* de seguridad informática: con este servicio tienen el objetivo de mantener al día a los clientes acerca del *software* de seguridad informática disponible en el mercado.
- ✓ Garantía de la Seguridad informática: este es un servicio para ayudar al cliente a aplicar las medidas adecuadas de seguridad. (7)

1.2.2. BASE4 Security

Es una empresa argentina que brinda servicios de Seguridad de la Información a toda Latinoamérica. Es una empresa innovadora en productos y servicios, y además, cuenta con profesionales con una extensa y reconocida trayectoria en el campo de la seguridad de la información. (8)

Algunos servicios brindados

- ✓ Gestión de la seguridad de la información (MSO por sus siglas en inglés) es un recurso especializado en la seguridad de la información, trabaja en sus instalaciones para operar y administrar los sistemas de seguridad así como asistir al cliente en la verificación, cumplimiento y control de las políticas de seguridad existentes en su organización. (8)

¹Asociación de jefes de Policía

- ✓ Monitoreo de seguridad administrado (MSM por sus siglas en inglés) este servicio es el encargado de realizar el monitoreo, operación remota y administración de los sistemas de seguridad que el cliente ya posee en su red. (8)
- ✓ Servicios de Evaluaciones de Seguridad de la Información es el servicio que utilizan especialistas de esta compañía para identificar las debilidades en la infraestructura informática de cualquier organización, estos se encargan de realizar ataques controlados, utilizando las mismas técnicas, herramientas y metodologías que utilizan los hackers para hacer vulnerable la seguridad lógica de una organización. (8)

1.2.3. Trustware

Compañía privada con oficinas en Mountain View, California y Tel-Aviv, Israel, ha desarrollado un *software* innovador de seguridad en Internet utilizando su tecnología patentada de virtualización para usuarios domésticos y de oficina en casa y empresas. El *software* desarrollado por esta compañía nombrado BufferZone es un programa para *Windows* que permite que los programas sospechosos realicen todas las operaciones necesarias para su correcto funcionamiento, pero sin dañar ficheros fiables o el sistema operativo. (9)

De acuerdo a que todas las compañías antes mencionadas son privativas, el proyecto LabSI no puede hacer uso de los servicios brindados por las mismas.

1.3. Exploración de soluciones posibles a reutilizar

1.3.1. Open Source Security Information Management (OSSIM)

Es una herramienta destinada a la gestión de la seguridad de la información basada en el monitoreo. Su objetivo es proporcionar una compilación comprensiva de los instrumentos que, trabajando juntos, conceden a un administrador de red una vista detallada sobre todo y cada uno de sus dispositivos, servidores de acceso de redes, etc., mostrándolos en una interfaz web única, desde donde el administrador puede observar todo lo que sucede. Esta herramienta proporciona una correlación de las interfaces de visualización y herramientas de gestión de incidentes que facilitan la presentación de reportes. Además posee la capacidad de actuar como un *Sistema de Detección/Protección de Intrusos* (IDS) basado en información correlacionada desde prácticamente cualquier fuente. (10)

OSSIM presenta un conjunto amplio de herramientas de seguridad, pero solo una de ellas (Nmap) se encuentra entre las que usa el proyecto LabSI en el proceso de pruebas. Las herramientas que tiene integradas pueden ser eliminadas o se le pueden añadir nuevas siempre y cuando se les construya el correspondiente *plugin*² para que puedan integrarse. Los resultados de las herramientas de OSSIM son recepcionados, procesados y estandarizados por la capa de estandarización la cual los entrega al servidor en una forma estándar.

Al parecer OSSIM posee características que se presentan como candidatas a la solución de los problemas en el proceso de pruebas de seguridad de LabSI, pero en realidad su finalidad difiere de las necesidades del proyecto. El hecho de que OSSIM esté orientado al monitoreo de la red mientras que el proyecto LabSI persigue la detección de vulnerabilidades en aplicaciones informáticas representa una diferencia considerable. Si bien es cierto que OSSIM permite la integración de nuevas herramientas, la mencionada diferencia implica que para la integración de las usadas por el proyecto sea necesario modificar la interfaz visual y el motor de correlación, así como crear los *plugins* integradores de dichas herramientas, lo que conlleva a la asimilación del código fuente de la aplicación cuya documentación es escasa, dificultándose su comprensión.

1.3.2. Sistema para Distribución de Tareas (Bambú)

Bambú es un sistema genérico que permite la ejecución de tareas de forma distribuida, así como la obtención y almacenamiento de sus resultados. Está compuesto por dos subsistemas, el Servidor y conectados a este uno o varios Agentes. El Servidor es el encargado de recibir cada tarea y asignársela a un Agente que pueda realizarla. Por su parte los Agentes son las PC que ejecutan finalmente las tareas y envían hacia el servidor el resultado obtenido. (11)

Bambú fue desarrollado en el centro TLM con vista a ser usado como marco de trabajo para la implementación de sistemas que trabajen de forma distribuida. Se implementó usando el lenguaje de programación Python del cual se tiene dominio, haciendo uso del estándar de codificación definido en la Universidad para el mencionado lenguaje, lo que facilita la asimilación y entendimiento de su código fuente. Se cuenta también con la documentación y demás artefactos generados durante su desarrollo.

Por las condiciones anteriores se propone el uso de Bambú como base para la implementación del sistema que responda a la solución del problema que compete a la presente investigación. Si bien es

² es un componente de software que añade capacidades específicas para una mayor aplicación de software

cierto que Bambú facilitaría el desarrollo de un sistema para la automatización del proceso de pruebas de seguridad en aplicaciones informáticas del proyecto LabSI, aún existe un conjunto de problemas que deben ser resueltos.

Entre los problemas existentes se tienen que:

- ✓ Bambú no es capaz de recibir las solicitudes de pruebas de seguridad, debido a que este predefine una serie de parámetros por cada tarea a ejecutar, los cuales son insuficientes para una prueba de seguridad del proyecto LabSI.
- ✓ Estrechamente relacionado con el problema anterior, la base de datos de Bambú no dispone de las tablas adecuadas para almacenar los datos de una prueba de seguridad del proyecto LabSI.
- ✓ La forma en que se distribuyen las tareas hacia los agentes no tiene en cuenta la capacidad de procesamiento de estos.
- ✓ Una vez que culmina la ejecución de una tarea es incapaz de notificar de dicho suceso.
- ✓ Carece de la capacidad de procesar los resultados generados por las herramientas para que sean almacenados de forma estándar en la base de datos.

Solucionando los inconvenientes encontrados, se logrará transformar el distribuidor de tareas genérico (Bambú) en un sistema basado en una arquitectura distribuida para la ejecución de pruebas de seguridad, que permita además la estandarización y almacenamiento de los resultados.

1.4. Tecnologías y herramientas utilizadas

Con el objetivo de lograr el sistema antes mencionado, que solucione los problemas existentes, se realiza un estudio sobre las tecnologías y herramientas a utilizar, las cuales fueron definidas por el proyecto Laboratorio de Seguridad Informática del Centro de Telemática. Seguidamente se realiza una breve descripción de las mismas:

1.4.1. Metodología de desarrollo

El Proceso Unificado de Rational (Rational Unified Process, por sus siglas en inglés RUP) es un proceso de desarrollo de *software* genérico que puede ser utilizado para varios tipos de sistemas, diferentes áreas de aplicación, diferentes tipos de organizaciones, niveles de competencia y diferentes tamaños de proyectos. (12)

Provee un enfoque disciplinado en la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Proporciona un entorno de proceso de desarrollo configurable, basado en estándares, permite tener claro y accesible el proceso de desarrollo que se sigue, permite ser configurado a las necesidades de la organización y el proyecto, además es un proceso orientado a objetos (13)

Además de las posibilidades que brinda RUP antes expuestas es utilizado porque proporciona un volumen significativo de información, que sirve de apoyo a futuros usuarios del sistema en cuestión.

1.4.2. Lenguaje de modelado: Lenguaje Unificado de Modelado (UML)

El Lenguaje de Modelado Unificado (UML) es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema de *software*. El UML ofrece una forma estándar para escribir los planos del sistema, incluyendo conceptualmente cosas tales como los procesos de negocio y las funciones del sistema, así como esencias concretas, como declaraciones de lenguaje de programación, esquemas de bases de datos y *software* reutilizables. (14) . Mediante UML es posible establecer la serie de requisitos y estructuras necesarias para plasmar un sistema de *software* previo al proceso intensivo de escribir código. En la presente investigación es usado con el fin de modelar los diagramas que representen las diferentes vistas de la solución que se pretende implementar, para lograr la adecuada comprensión de la misma.

1.4.3. Herramienta case: Visual Paradigm

Visual Paradigm es una herramienta CASE para el diseño UML que ayuda al desarrollo de *software*. Está diseñada para usuarios como Ingenieros de *Software*, Analistas de Sistemas, Arquitectos de Sistemas y otros que estén interesados en el diseño de *software* orientado a objetos. (15)

Se utiliza para el desarrollo del sistema en cuestión porque soporta el ciclo de vida completo del desarrollo de *software*, por su estabilidad de ejecución en diferentes sistemas operativos y posee una interfaz agradable y fácil de usar. Con el uso de esta herramienta será posible modelar los artefactos propuestos

por RUP que se ajusten a la presente investigación, con el objetivo de lograr un claro entendimiento de lo que se desea hacer.

1.4.4. Herramienta de Desarrollo: Eclipse

Eclipse es un entorno de desarrollo integrado (IDE, *Integrated Development Environment*) que facilita enormemente las tareas de edición, compilación y ejecución de programas durante su fase de desarrollo. Es una aplicación gratuita de código abierto, multi-idioma que aunque se ha ganado su popularidad con el lenguaje java, existen *plugins* para varios lenguajes entre los cuales se encuentra python. (16)

Para la implementación del sistema en cuestión se usará el IDE Eclipse con el correspondiente *plugin* para el lenguaje de programación python, con el objetivo de que el código se fácil de entender.

1.4.5. Sistema Gestor de Bases de Datos: PostgreSQL

PostgreSQL es un Sistema Gestor de Bases de Datos, distribuido bajo licencia BSD³ (Berkeley Software Distribución) y con su código fuente disponible libremente. Utiliza un modelo cliente/servidor y usa multiprocesos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto. (17)

El sistema Gestor de Base de Datos PostgreSQL es usado con el fin de realizar el almacenamiento del contenido estandarizado de los reportes generados por las herramientas de prueba del proyecto LabSI.

1.4.6. Administrador de Bases de Datos: PgAdmin

Es una interfaz de administración para gestionar bases de datos PostgreSQL. Es multiplataforma y puede funcionar bajo *GNU/Linux, Windows, etc...*

Permite desde ejecución de consultas SQL simples hasta la elaboración de bases de datos complejas. El *software* es liberado con un instalador y no requiere ningún controlador adicional para comunicarse con el servidor de base de datos. El interfaz gráfico soporta todas las características de PostgreSQL y facilita enormemente la administración. (18)

1.4.7. Lenguaje de programación: Python

Es un lenguaje independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones *Windows* a servidores de red o incluso, páginas *web*. Es un lenguaje

³Es un sistema operativo derivado del sistema Unix, que presenta una licencia de software libre permisivo la cual impone pocas restricciones sobre la forma de uso, alteraciones y redistribución del software.

interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo. El creador del lenguaje es un europeo llamado Guido Van Rossum. (19)

Este lenguaje tiene varias características positivas por las cuales es usado para la implementación del sistema en cuestión, es un lenguaje multiplataforma, orientado a objetos por lo que ofrece en varios casos una manera sencilla de crear programas con componentes reutilizables. Tiene además una sintaxis muy visual gracias a la notación con márgenes de obligado cumplimiento. Para separar las porciones de código en Python se debe tabular hacia dentro, colocando un margen al código que iría dentro de una función o un bucle. Esto ayuda a que todos los programadores adopten las mismas notaciones y que los programas de cualquier persona tengan un aspecto similar.

1.4.8. SQLite

SQLite es un proyecto de dominio público el cual implementa una librería de aproximadamente 500kb, programado en el lenguaje C y totalmente libre. SQLite es independiente, simplemente se realizan llamadas a sub rutinas o funcionales de las propias librerías de SQLite, lo cual reduce ampliamente la latencia en cuanto al acceso a las base de datos compuestas por la definición de las tablas, índices, y los propios datos son guardados por un solo fichero estándar y en un solo ordenador. Esta herramienta además tiene una pequeña memoria y una única biblioteca necesaria para acceder a bases de datos, lo que la hace ideal para aplicaciones de bases de datos incorporadas. Es multiplataforma y sus bases de datos pueden ser fácilmente portadas sin ninguna configuración o administración. Cuenta con diferentes interfaces del API⁴, las cuales permiten trabajar con C++, PHP, Python etc. (20)

En la presente investigación se hace uso de SQLite con el objetivo de almacenar en los agentes datos referentes a la ejecución de las pruebas.

1.4.9. Middleware

Es un *software* orientado a proporcionar conectividad o integración entre diferentes aplicaciones, normalmente distribuidas, y en el peor de los casos sobre recursos heterogéneos. Para ello suele ser una capa de *software* que se encuentra entre las aplicaciones y los sistemas operativos. (21)

⁴Interfaz de Programación de aplicaciones

1.4.10.ICE

Internet Communication Engine (ICE) es un estándar desarrollado por ZeroC para el desarrollo de aplicaciones basadas en objetos distribuidos, es decir, ICE proporciona herramientas, APIs, y soporte de bibliotecas para construir aplicaciones cliente-servidor orientadas a objetos. Una aplicación ICE se puede usar en entornos heterogéneos, los clientes y los servidores pueden escribirse en diferentes lenguajes de programación, pueden ejecutarse en distintos sistemas operativos y en distintas arquitecturas, y pueden comunicarse empleando diferentes tecnologías de red. Además el código fuente de estas aplicaciones puede portarse de manera independiente al entorno de desarrollo. (22)

Este estándar es utilizado en la presente investigación para establecer comunicación entre cliente y servidor.

1.4.11.Glacier2

Es una solución de *firewall* para ICE que permite que clientes y servidores se comuniquen de forma segura. Donde el tráfico entre cliente y servidor queda completamente encriptado. Además proporciona soporte para la autenticación mutua y para la gestión segura de sesiones. (23)

1.4.12.WebSocket

WebSocket es una tecnología que hace posible abrir una sesión de comunicación interactiva entre el navegador del usuario y un servidor. Utilizando una conexión *WebSocket*, las aplicaciones *Web* pueden realizar comunicaciones en tiempo real en lugar de tener que sondear a los cambios de ida y vuelta. (24)

Esta tecnología será usada con el objetivo de realizar notificaciones al *front end*.

1.5. Análisis de posibles soluciones estandarización de reportes

Después de haber realizado un estudio del arte y de las herramientas a utilizar, se ha realizado un análisis de las posibles soluciones a tener en cuenta para la implementación de la extracción y estandarización de la información relevante de los reportes de las herramientas de pruebas de seguridad en aplicaciones.

1.5.1. Parser⁵

Una posible solución sería implementar un parser al menos para cada herramienta o para cada formato de reporte generado por cada herramienta. Esto presupone un arduo trabajo de codificación por la diversidad de formatos y herramientas existentes. Además la inclusión de una nueva herramienta implicaría la implementación de un nuevo parser que interprete los resultados de esta.

1.5.2. Pyparsing

Dado que el sistema Bambú está implementado en python una posible solución sería usar pyparsing, la cual es una biblioteca de clases que permite definir *parsers* con solo implementar la gramática usando las mismas clases que propone. (25)

La presente podría ser una solución, más la inclusión de nuevas herramientas implica codificar usando el lenguaje python, las respectivas gramáticas que usaría pyparsing. Para la implementación del sistema en cuestión resultaría trabajoso conformar gramáticas tipo pyparsing para extraer contenidos de los reportes, pues estas resultan ser complejas y además no permiten obtener la información en forma de instancias de clases.

1.5.3. Capa de abstracción de pyparsing

Otra posible solución podría ser la utilización del componente Capa de Abstracción de Pyparsing implementada en el centro de Telemática el cual permite definir gramáticas para pyparsing usando el metalenguaje XML y que los resultados sean devueltos en forma de instancias de clase. Esta presenta como inconveniente que es necesario definir todos los literales y palabras posibles a encontrarse en el contenido a *parsear*, lo cual no es factible pues los reportes de las herramientas pueden tener literales y palabras muy variados que incluso no son parte de la información relevante. La Capa de abstracción de Pyparsing no permite definir una pequeña gramática para encontrar coincidencias y extraerlas en un contenido a parsear.

1.5.4. Expresiones regulares

Con el uso de expresiones regulares al igual que con pyparsing se pueden extraer de un contenido las coincidencias de un patrón definido pero presenta el mismo inconveniente de no poder devolver el

⁵ analizador sintáctico (en inglés parser) es una de las partes de un compilador que transforma su entrada en un árbol de derivación.

resultado como instancias de clases creadas por el usuario, por lo que habría que implementar dicha transformación para cada herramienta que se desee incluir.

1.5.5. XML con expresiones regulares

Como solución se propone la implementación de un paquete que permita extraer la información relevante de uno o varios archivos y devuelva esta en forma de instancias de clases. Haciendo uso del metalenguaje XML y de las expresiones regulares. Para tal propósito se ha propuesto una forma de definir la información que se desea extraer y la clase en la cual se obtendrá el resultado como instancia. La forma de definir la información será un XML en el cual se especificará por ejemplo el nombre de la clase que será instanciada, el nombre de sus parámetros y las expresiones regulares que serán usadas para extraer la información y así dar valores a los atributos de la clase. Para cada herramienta será necesario escribir el archivo XML que define la información relevante a extraer.

1.6. Conclusiones

Luego de definir el marco conceptual es posible un mayor entendimiento de las definiciones relacionadas con la seguridad informática. Después de realizar el estudio de sistemas similares existentes, se constató que las principales empresas que realizan pruebas de seguridad no brindan detalles de la realización de las mismas. Además se comprobó que OSSIM no satisface las expectativas del proyecto LabSI, ya que se hace más difícil para el mismo hacer las modificaciones que requiere OSSIM de acuerdo a las necesidades del proyecto que implementar una solución propia. También se realizó un análisis acerca del sistema Bambú, en el cual se llegó a la conclusión de que es factible hacer uso del mismo como marco de trabajo para dar cumplimiento al objetivo general, aunque éste presente un conjunto de deficiencias, las mismas pueden ser solucionadas pues está implementado en el lenguaje de programación Python del cual se tiene dominio, y además se cuenta con la documentación y los artefactos generados durante la implementación de Bambú, lo que ayuda en la comprensión de dicho sistema.

De acuerdo a lo anteriormente mencionado se deriva la necesidad de la implementación de un sistema que sea capaz de satisfacer las expectativas esperadas. Teniendo en cuenta que con el empleo de las herramientas anteriormente argumentadas, se da la posibilidad de desarrollar la solución requerida.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

Introducción

Es importante para LabSI la realización de las pruebas de forma distribuida así como la estandarización y almacenamiento de los reportes que generan las pruebas de seguridad en aplicaciones. Puesto que este proceso en la actualidad es realizado manualmente sin permitir mantener el control adecuado sobre los reportes generados por las pruebas de seguridad realizadas. En este capítulo es realizada una descripción del sistema, además son expuestos los requisitos funcionales y no funcionales necesarios para el desarrollo del *software*.

2.1. Objeto de automatización

Para eliminar los problemas existentes con el proceso de pruebas en el proyecto LabSI, es necesario realizar la implementación de un sistema que realice las pruebas de forma distribuida y que además procese y almacene dichos reportes de forma automática. Proporcionando persistencia de la información obtenida de las pruebas de seguridad, de forma estándar y simplificada.

2.2. Propuesta de solución

Se propone la implementación de un sistema que sea capaz de distribuir las pruebas de seguridad de forma automática y que además procese y almacene en una base de datos común los resultados de las mismas, que al ser integrado a un entorno web conformará la Plataforma de Pruebas de Seguridad en el *software*. Dicho sistema tendrá funcionalidades tanto en el servidor⁶ como en la(s) máquina(s) agente⁷(s), además tendrá la responsabilidad en el servidor de recibir del *front end* la solicitud de prueba y asignársela al agente correspondiente, así como asignar la solicitud de ejecución de herramientas de pruebas de seguridad a los hilos de ejecución, y una vez asignada la solicitud el mismo debe ser capaz de ejecutar la herramienta correspondiente a la solicitud realizada.

Se implementará un conjunto de clases que sean capaces de extraer la información importante de los reportes generados por las herramientas una vez realizada la prueba mediante el uso de expresiones regulares y el metalenguaje XML, para enviarlos hacia el servidor y almacenarlos en la base de datos. Cuando sea realizado todo este proceso, se procede a enviar una notificación al *front end* desde el servidor para que muestre un informe con los datos relevantes de las pruebas realizadas.

⁶ Servidor es una computadora que, formando parte de una red, provee servicios a otras computadoras denominadas clientes.

⁷ Son las computadoras donde estarán instaladas las herramientas de pruebas de seguridad.

2.3. Modelo del dominio

El objetivo del modelo de dominio es ayudar a comprender los conceptos que utilizan los usuarios, los conceptos con los que trabajan y con los que deberá trabajar la aplicación. En él se representan los conceptos del dominio que interesan, sus características y las relaciones entre los mismos. (26)

En el presente diagrama se presentan los conceptos a ser tratados en la solución. A continuación se hace una breve descripción del mismo:

Una prueba es hecha por una o varias herramientas, una herramienta genera uno o varios reportes, el reporte tiene contenido y formato, el formato puede ser XML o txt.

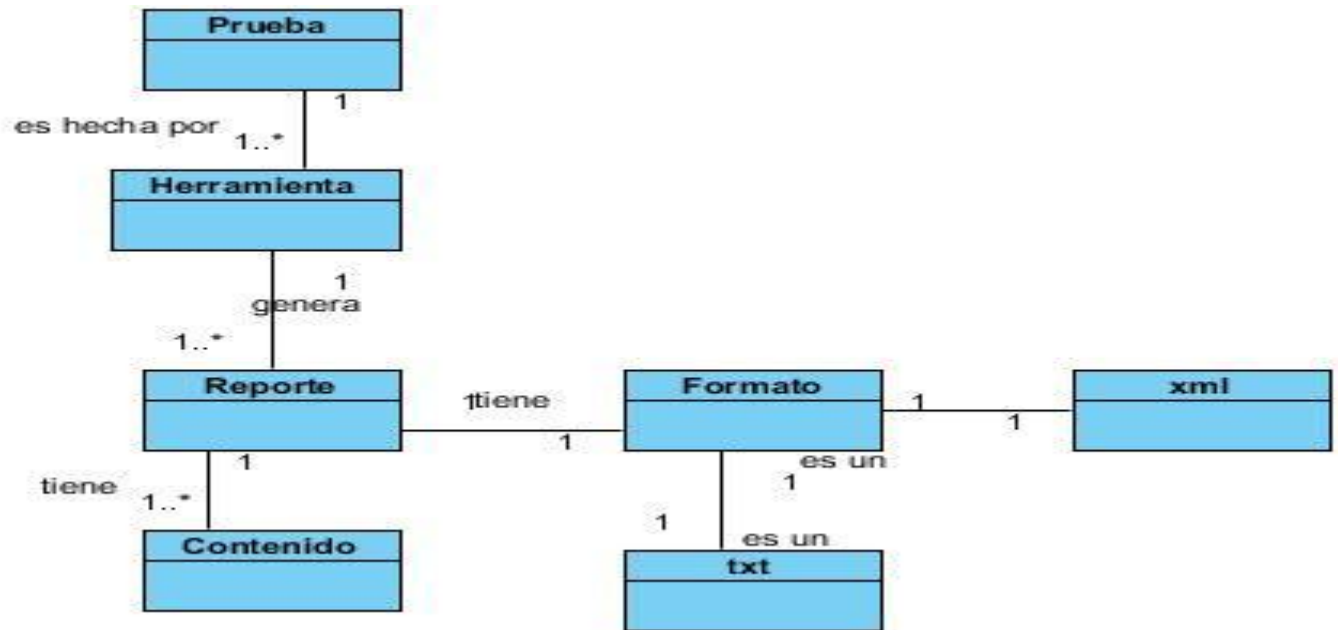


Fig. 1: Diagrama de dominio

2.3.1. Conceptos:

Herramienta: son las herramientas de prueba que se utilizan en el proyecto LabSI.

Reporte: los reportes son los resultados generados por las herramientas de prueba.

Contenido: es lo que se encuentra dentro del reporte.

Formato: es la extensión en la que puede ser generado el reporte.

Prueba: es una actividad más en el proceso de aseguramiento de la calidad.

Txt: es un archivo de texto plano.

XML: Es un estándar abierto, flexible y ampliamente utilizado para almacenar, publicar e intercambiar cualquier tipo de información.

2.4. Requisitos funcionales

Un requisito funcional define el comportamiento interno del *software*: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica. (27)

Servidor

RF1. Atender solicitud de ejecución de herramientas de pruebas de seguridad.

RF1.1. Atender solicitud de ejecución de herramientas de pruebas de seguridad de tarea Nikto.

RF1.2. Atender solicitud de ejecución de herramientas de pruebas de seguridad de tarea Wapiti.

RF1.3. Atender solicitud de ejecución de herramientas de pruebas de seguridad de tarea Nmap.

RF2. Distribuir tareas atendiendo a los criterios de hilos de ejecución disponibles.

RF3. Almacenar el contenido del reporte estandarizado en la base de datos.

RF4. Notificar estado de culminación de la prueba.

Agente

RF1. Procesar contenido del reporte de las herramientas de pruebas de seguridad Nikto, Wapiti y Nmap.

RF1.1 Cargar contenido del archivo de definición de información.

RF1.2 Leer el fichero del reporte de la herramienta de pruebas de seguridad.

RF1.3 Generar resultado estandarizado.

RF2. Enviar resultado estandarizado al servidor.

2.5. Requisitos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Normalmente están vinculados a requisitos funcionales. Dichos requisitos forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto.

Requisitos de Software

Los ordenadores donde va a ser instalado el servidor o el cliente deben contar con sistema operativo *GNU/Linux*, también debe tener instalado el intérprete de Python 2.6 y el middleware ZeroC Ice. Además el ordenador donde resida el servidor debe tener instalado el Gestor de base de datos PostgreSQL y el del cliente debe contar con SQLite.

Requisitos de hardware del servidor y el agente.

Las características de hardware con que debe contar la estación desde la cual se ejecutará el servidor estarán en función del número de agentes asociados a este. Para lograr un desempeño estable con una cantidad reducida de agentes conectados al servidor, bastaría con una PC que sea capaz de soportar la ejecución estable de un sistema operativo Ubuntu 10.04. De igual forma los requerimientos de hardware de los agentes dependerán de las herramientas que se pretendan ejecutar en ellos. Los requerimientos mínimos tanto para el servidor como para los agentes serían:

- 512 Mb de memoria RAM
- Microprocesador a 2.6 GHz

Requisitos de Soporte

Debe ser implementado siguiendo el estándar de codificación definido por la Dirección técnica de la UCI.
(28)

Escalabilidad

El sistema debe permitir agregar nuevas herramientas fácilmente.

Restricciones en el diseño y la implementación

Para la implementación del sistema se usa como lenguaje de programación Python con la versión 2.6 de su intérprete o superior.

Se utiliza para la implementación en Python la herramienta de *software* libre Eclipse con el correspondiente *plugin* para dicho lenguaje. La implementación en Python está apoyada en el Framework del departamento de Seguridad Informática del centro TLM para el mencionado lenguaje. Se usa como gestor de base de datos PostgreSQL y como herramienta de administración para el mismo se utiliza PgAdminIII.

Requisitos de Seguridad

En el sistema a desarrollar se pretende realizar una comunicación segura entre cliente y servidor haciendo uso de los servicios de ICE Glacier2.

Requisitos Legales

El sistema será propiedad exclusiva de la Universidad de Ciencias Informáticas.

2.6. Modelo de Caso de uso del sistema

2.6.1. Actores del sistema

Descripción	Actor
Agente	El actor agente representa una PC, donde están instaladas las herramientas de prueba, y es además donde se va a realizar el procesamiento de la información del reporte para ser enviada al servidor.
Servidor	El sistema servidor representa una PC en la cual se van a realizar una serie de funciones tales como distribuir tareas, recepcionar el resultado de la prueba y realizar el almacenamiento del contenido del reporte en la base de datos.

Tabla 1: Descripción de los actores del sistema

2.6.2. Diagrama de Casos de uso del sistema

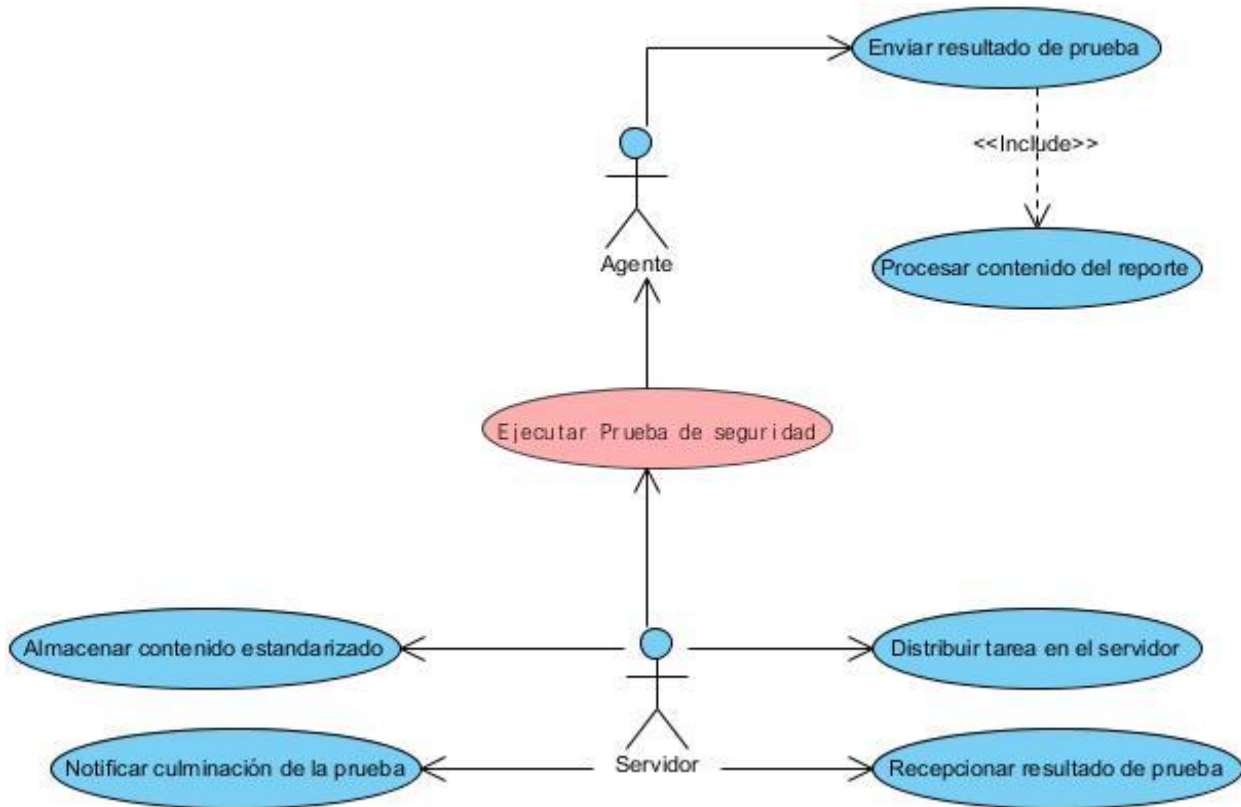


Fig. 2: Diagrama de casos de uso del sistema

2.6.3. Descripción de casos de uso del sistema

Caso de Uso:	Distribuir tarea en el servidor.
Actores:	
Resumen:	Se distribuye la tarea hacia el agente que contenga la herramienta de prueba teniendo en cuenta los procesos que se llevan a cabo en este agente.
Precondiciones:	La tarea debe haber sido enviada al servidor.
Referencias	

Prioridad	Crítica
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
	1. Se asigna la tarea al agente que contenga la herramienta de prueba correspondiente y la mayor cantidad de hilos de ejecución disponibles.
Flujos Alternos	
1.1. No hay agente disponible	
Acción del actor	Respuesta del Sistema
	La tarea se mantiene en la base de datos en estado de espera, hasta que exista un agente que la pueda ejecutar.

Tabla 2: Descripción del caso de uso distribuir tarea en el servidor

Caso de Uso:	Procesar el contenido del reporte
Actores:	
Resumen:	Una vez generado el reporte por la herramienta de prueba, se lleva a cabo el procesamiento del mismo, haciéndole un parseo para normalizar el contenido y almacenarlo en la base de datos
Precondiciones:	El reporte debe estar generado
Referencias	
Prioridad	Crítica
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
	1. El sistema lee el archivo que contiene las definiciones de información.

	<p>2. El sistema lee el contenido de los archivos de reportes generados.</p> <p>El sistema realiza la extracción de la información relevante contenida en los archivos del reporte.</p>
Flujos Alternos	
1.1. No se encuentra el archivo	
Acción del Actor	Respuesta del Sistema
	Si no se encuentra el archivo de reporte generado, se levanta una excepción.

Tabla 3: Descripción del caso de uso procesar el contenido del reporte

Caso de Uso:	Enviar resultado de prueba.	
Actores:		
Resumen:	Se envía del agente hacia el servidor el contenido del reporte estandarizado y el resultado de la prueba.	
Precondiciones:	Se debe haber procesado el contenido del reporte.	
Referencias		
Prioridad	Crítica	
Flujo Normal de Eventos		
Acción del Actor	Respuesta del Sistema	
	1. El sistema envía al servidor el fichero del resultado de la prueba y del contenido del reporte estandarizado.	
Flujos Alternos		
1.1. Servidor sin conexión		
Acción del actor	Respuesta del Sistema	
	Si no se puede enviar el resultado hacia el servidor, se almacena la tarea y el resultado en la base de datos Sqlite.	

--	--

Tabla 4: Descripción del caso de uso enviar resultado de prueba

Caso de Uso:	Almacenar el contenido del reporte en la base de datos	
Actores:		
Resumen:	Después de haber realizado el procesamiento del reporte y enviado al servidor se procede a realizar el almacenamiento del este resultado normalizado.	
Precondiciones:	Deben estar hechas las tablas de contenido estandarizado	
Referencias		
Prioridad	Crítica	
Flujo Normal de Eventos		
Acción del Actor	Respuesta del Sistema	
	1. El sistema almacena en la base de datos el contenido del reporte estandarizado.	

Caso de Uso:	Notificar culminación de la prueba	
Actores:		
Resumen:	Una vez terminado todo el proceso de prueba y el almacenamiento del resultado en la base de datos, se le hace una notificación al front end para que pueda acceder a la base de datos y mostrar este resultado.	
Precondiciones:	Debe haber terminado el proceso de almacenamiento.	
Referencias		
Prioridad	Crítica	
Flujo Normal de Eventos		
Acción del Actor	Respuesta del Sistema	
	1. El sistema avisa al Front-End de la culminación de todo el proceso.	

Tabla 5: Descripción del caso de uso Notificar culminación de la prueba

2.7. Conclusiones

Con la realización de este capítulo es posible dar inicio al desarrollo del sistema para la ejecución de pruebas de seguridad de forma distribuida, procesamiento y almacenamiento automático de los reportes generados por las herramientas de prueba, después de haber realizado un análisis profundo de los procesos que intervienen en el modelo de dominio. Fueron definidas las funcionalidades del sistema mediante el levantamiento de requisitos funcionales y no funcionales. Además de las descripciones de los casos de uso y los actores del sistema.

CAPÍTULO 3: DISEÑO DEL SISTEMA

Introducción

Cada paso durante el ciclo de desarrollo del *software* tiene un papel importante para la obtención del producto final. En este capítulo se describe la arquitectura empleada y los patrones de diseño utilizados para la implementación del sistema. Así como el diagrama de clases persistentes, diagramas de clases del diseño y de paquetes.

3.1. Modelo de diseño

Con el objetivo de realizar una representación previa del sistema a desarrollar se realiza el modelo de diseño, el cual sirve de abstracción a la implementación.

3.1.1. Diagrama de paquetes

Los diagramas de paquetes agrupan lógicamente las divisiones de un sistema, representando las dependencias entre sus agrupaciones. En el presente diagrama fueron agregados los paquetes `ImplementClass`, `Standard` y `Balancer`.

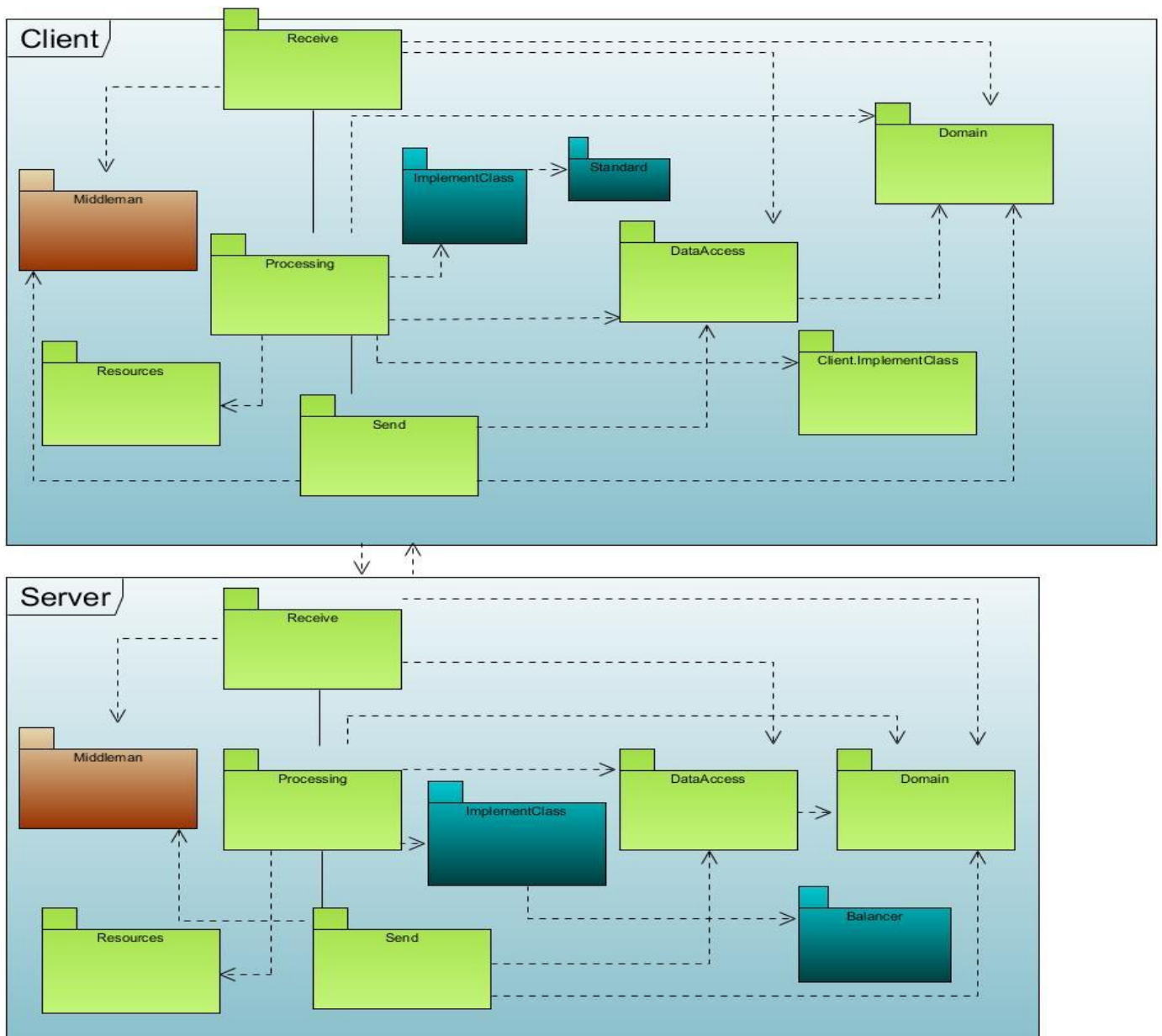


Fig. 3: Diagrama de paquetes

3.1.2. Diagrama de clases

A través del flujo de trabajo de diseño uno de los artefactos más importantes a obtener son los diagramas de clases, donde se exponen las clases del diseño que intervienen en las realizaciones de los casos de uso del sistema.

3.1.3. Cliente

En el diagrama de clases del cliente fueron agregados los paquetes: Standard que es el que contiene las clases que tendrán la responsabilidad de realizar la extracción de la información relevante de los reportes de prueba y el paquete ImplementClass que es el que contiene las clases encargadas de vincular las herramientas con el sistema.

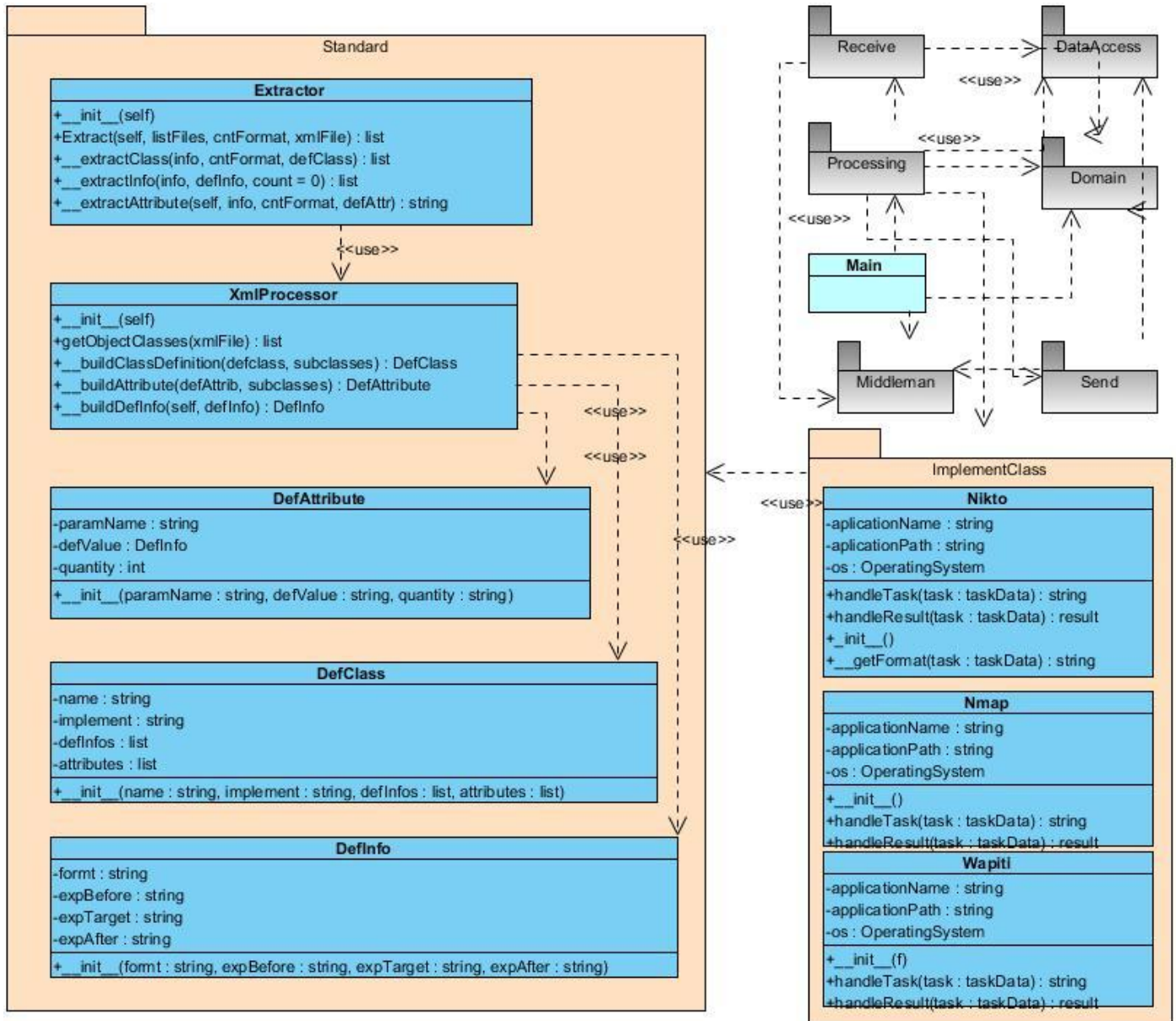


Fig. 4: Diagrama de clases cliente

3.1.4. Servidor

En el diagrama de clases del servidor fueron agregados los paquetes Balancer e ImplementClass, además de las clases agregadas en el paquete Domain y DataAcces.

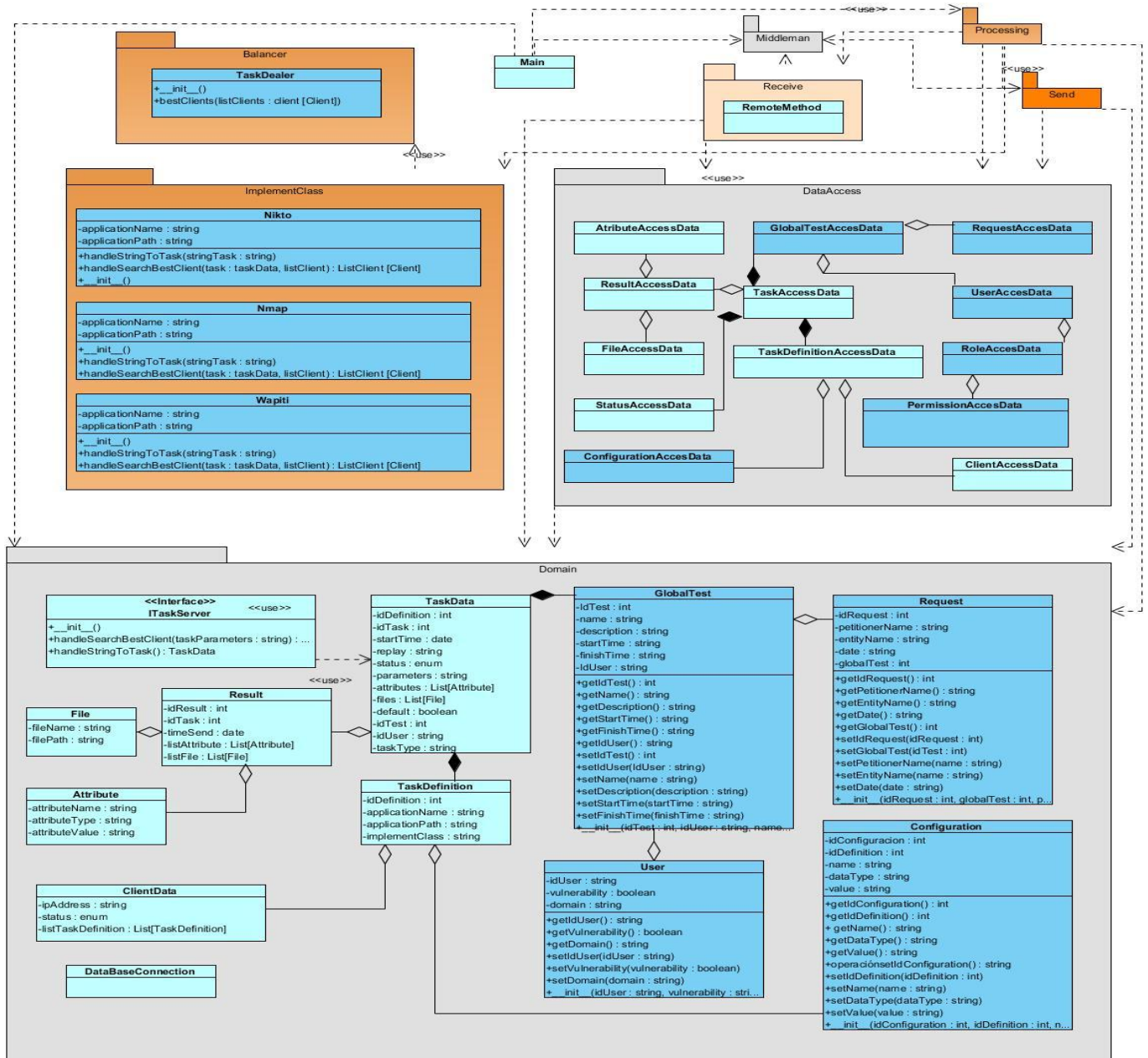


Fig. 5: Diagrama de clases servidor

3.2. Arquitectura del sistema

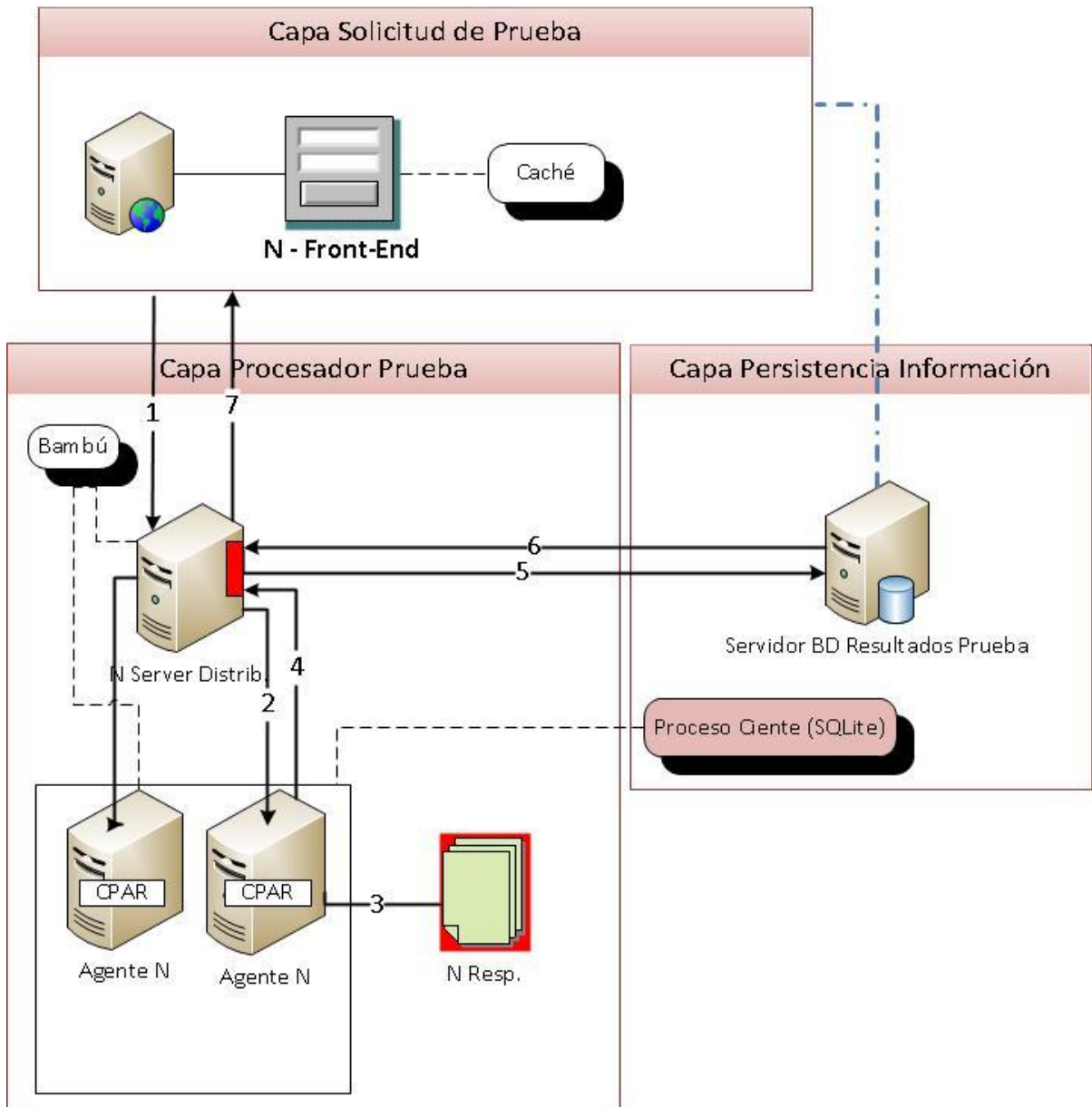

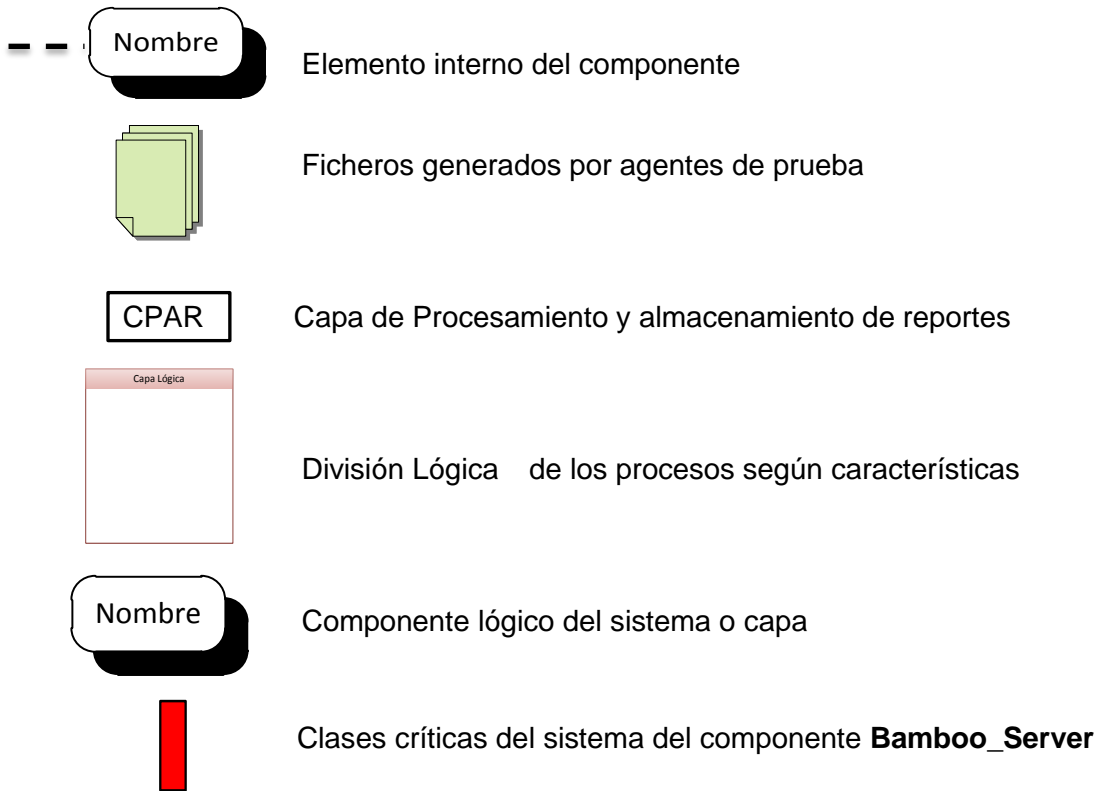


Fig. 6: Arquitectura de la Plataforma de Pruebas de Seguridad en los software.

Detalles de los componentes del diagrama:

N
 Dirección del flujo de proceso y orden que corresponde



Descripción de los procesos del diagrama

Flujo de proceso 1: Capa Solicitud de Prueba → Capa procesador de Prueba (N Servidor distribuidor)

El proceso se inicia en la capa de Solicitud de Prueba (Front end) donde se configuran las pruebas a realizar con los parámetros e informaciones necesarias por un especialista. De forma estándar los parámetros e informaciones son enviados al (los) servidor (es).

Flujo de proceso 2: Capa Procesador Prueba (N Servidor distribuidor – Agente N)

El proceso va a estar asociado directamente al componente Bambú_Server (modificado) que estará alojado en los servidores distribuidores. Cuando el servidor distribuidor reciba la configuración de las pruebas a realizar, el componente Bambú_Server será el encargado de distribuir las tareas a cada uno de los agentes de prueba, donde cada agente cuenta con el componente Bambú_Client y de este modo se realiza la comunicación, este proceso debe tener en cuenta el rendimiento de los agentes para las tareas a realizar (Versión 2.0). El componente Bambú_Server accede a una base de datos donde va a estar

registrado el (los) agente (s) que puede utilizar para las tareas a realizar según especificaciones del flujo anterior.

Cada **agente de prueba** al recibir una tarea asigna la ejecución de esta a uno de sus hilos de proceso, esta asignación debe realizarse de la forma más conveniente posible (**Versión 2.0**)

Una vez que **Bambú_Client** recibe una tarea almacena en una **base de datos de SQLite** información temporal referente a la tarea y a su estado de ejecución, dicha información se actualiza durante la ejecución y es eliminada una vez que **Bambú_Client** crea el resultado de la ejecución de la tarea.

Flujo de proceso 3: Capa Procesador Prueba (Agente N – Fichero de respuesta)

La CPAR, una vez concluida la tarea que genera un fichero de formatos diferentes según las pruebas, se encarga de normalizar dichas pruebas para obtener una estructura estándar para luego insertar en base de datos y asegurar la permanencia de los resultados de prueba.

Flujo de proceso 4: Capa Procesador Prueba (Agente N – N Servidor distribuidor)

Culminada la tarea de procesamiento y normalización de la información en la CPAR, a través de la comunicación Bambú_Client - Bambú_Server, primeramente el fichero obtenido de la tarea del proceso ejecutado en el agente es enviado al (los) servidor (es) distribuidos (es) (Versión 2.0, n servidores) y luego los datos normalizados en la CPAR son enviados igualmente en forma de la estructura definida al servidor N distribuidor, donde el conjunto de clases críticas adicionadas al Bambú_Server, se encargará de enviarlas a la base de datos correspondiente.

Flujo de proceso 5: Capa Procesador Prueba - > Capa Persistencia de la información (servidor BD PostgreSQL)

Los datos estandarizados y recibidos de los agentes, son enviados por las clases críticas a la base de datos para ser almacenados.

Flujo de proceso 6: Capa Persistencia de la información (servidor BD PostgreSQL)- > Capa Procesador Prueba

El servidor de base de datos informa sobre la acción de guardar datos, en correspondencia con lo sucedido, si hubo un error u ocurrió el proceso eficientemente.

3.4.1. Arquitectura n-capas

Se utiliza la arquitectura n-capas, donde las capas aparecen tanto en el agente como en el servidor. Este estilo arquitectónico es un estilo de llamada y retorno, en el mismo cada capa proporciona servicios a la capa superior y se sirve de las prestaciones que le brinda la inferior, al dividir un sistema en capas, cada capa puede tratarse de forma independiente, sin tener que conocer los detalles de las demás. La división de un sistema en capas facilita el diseño modular, en la que cada capa encapsula un aspecto concreto del sistema y permite además la construcción de sistemas débilmente acoplados, lo que significa que si se minimiza las dependencias entre capas, resulta más fácil sustituir la implementación de una capa sin afectar al resto del sistema (29)

A continuación se realiza una breve descripción de las funcionalidades de cada capa.



3.4.2. Descripción de las capas

Procesamiento: Procesa los datos recibidos en el servidor y también los recibidos en el agente, además de los resultados de las pruebas de seguridad que son realizadas.

Recepción: Se encarga de recibir y almacenar los datos enviados a través del middleware, ya sea los datos entre el sistema agente y el servidor o los datos que el servidor recibe del Front End.

Acceso a Datos: Componente encargado de almacenar, buscar, actualizar, o borrar los datos en la base de datos.

Envío: Esta capa es la encargada de enviar los datos a través del middleware.

Dominio: Es una capa que brinda la posibilidad del manejo de los objetos del dominio del sistema.

3.4.3. Patrones de diseño

3.4.3.1. Experto

Este patrón de diseño define asignar la responsabilidad a la clase que cuente con la información necesaria para manejar esta responsabilidad. Es necesario que la asignación de responsabilidades a las clases se lleve a cabo de una forma adecuada, lo que permitirá que el sistema sea más fácil de entender así como que se puedan reutilizar componentes. Con su aplicación se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. (30)

Se requirió el empleo de este patrón para la creación de una clase por cada herramienta en el paquete `ImplementClass`, tanto en el Servidor como en el Agente. De esta forma que existe una clase experto por cada herramienta capacitada para manipular las pruebas de seguridad asociadas a la herramienta.

3.4.3.2. Alta cohesión

Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un gran trabajo. Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. (30)

Con su utilización en el sistema se logró tener bien definidas las responsabilidades de cada clase.

3.4.3.3. Bajo acoplamiento

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Un alto acoplamiento tendría como consecuencia que las clases sean más difíciles de entender cuando estén aisladas, más difíciles de reutilizar porque se requiere la presencia de otras clases de las que dependen. Por lo que un bajo acoplamiento plantea que debe haber pocas dependencias entre las clases. (30)

3.4.3.4. Singleton

Se encarga de garantizar que una clase solo tiene una única instancia, proporcionando un punto de acceso global a la misma. La utilización del patrón *Singleton* se hizo evidente una vez que se requería el acceso a una única instancia del servidor *websocket* desde varios puntos del sistema.

3.4.3.5. Creador

Este patrón como su nombre lo indica es el que crea, el guía la asignación de responsabilidades relacionadas con la creación de objetos, se asigna la responsabilidad de que una clase B cree un Objeto de la clase A. (31)

Para la concepción del paquete Standad se hizo empleo de este patrón debido a que las instancias de las clases `DefClass`, `DefInfo` y `DefAttribute` solo son creadas por la clase `XMLProcessor`.

3.5. Modelo de datos

Un modelo de datos es un lenguaje utilizado para la descripción de una base de datos. Por lo general, un modelo de datos permite describir las estructuras de datos de la base, es decir el tipo de los datos que incluye la base y la forma en que se relacionan.

3.5.1. Modelo lógico de datos

El diagrama de clases persistentes es diseñado a través del diagrama de clases del diseño, definiendo cuáles son las clases que van a persistir para la confección del mismo. Para ver la descripción las tablas dirigirse al [Anexo 2](#).

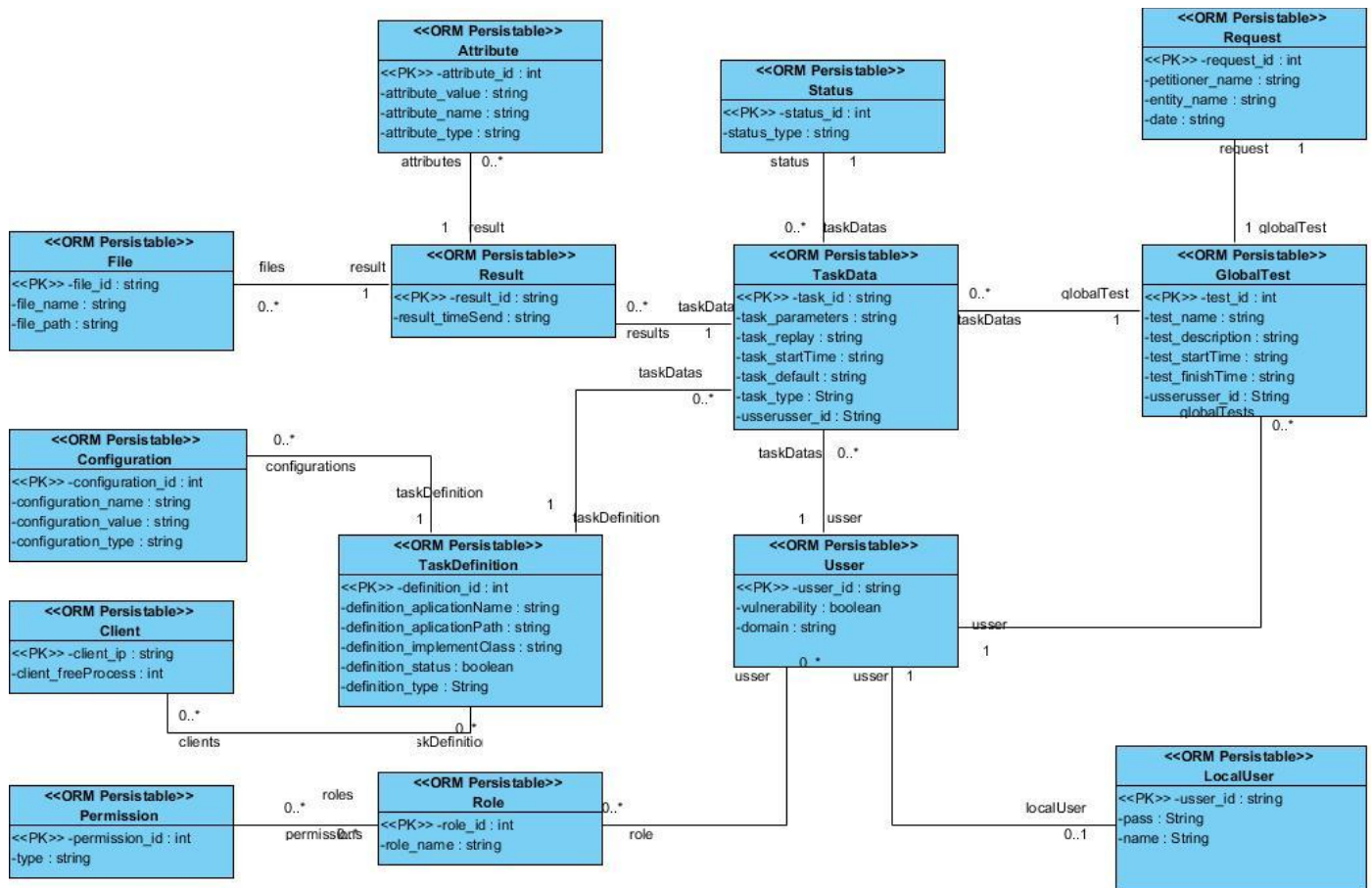
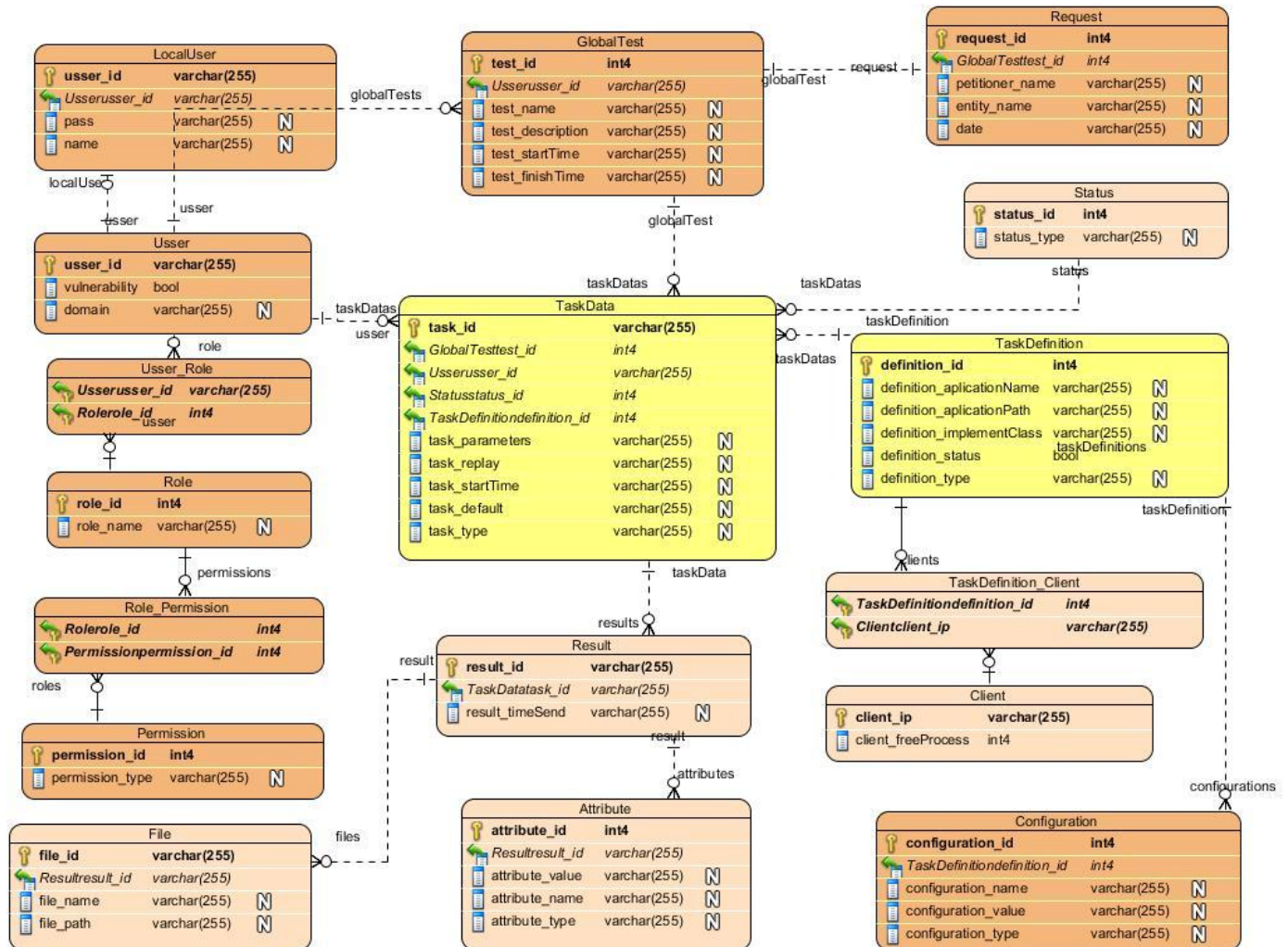


Fig. 7: Diagrama del modelo lógico de datos

3.5.2. Modelo físico de la base de datos

El modelo físico de datos fue generado a partir del diagrama de clases persistentes, para lo cual se hizo uso de la herramienta Visual Paradigm for UML 8.0. Para ver la descripción del modelo dirigirse al [Anexo3](#).



3.6. Conclusiones

La realización de los diagramas de clases del diseño y las clases entidades, así como la definición de los patrones de diseño y de arquitectura expresan la organización del sistema y aumenta la facilidad de desarrollo, necesaria para la implementación del sistema. Además de la definición de la arquitectura descrita anteriormente y la descripción de la base de datos.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

4.1. Introducción

Como resultado del diseño se procede a la implementación del sistema en términos de componentes. En este capítulo se tiene como objetivo desarrollar la arquitectura y el sistema como un todo. Para lograr una correcta implementación, es necesario que la solución propuesta satisfaga las expectativas del usuario final, asegurando que funcione de acuerdo a los requisitos anteriormente definidos. Derivándose entonces la incorporación de pruebas al sistema, actividad para garantizar la calidad del *software*.

4.2. Extracción de información mediante XML y expresiones regulares

A continuación se realiza una explicación detallada del XML mencionado en el capítulo 1 para darle solución a la extracción de la información de los reportes generados por las herramientas de prueba.

El XML: Uno por cada herramienta a incluir en la plataforma. Todo archivo XML que se defina con el objetivo de que el sistema sea capaz de extraer y estandarizar el contenido de reporte de una herramienta debe contener las siguientes etiquetas:

Etiqueta `<main></main>`: Es la raíz del documento y contiene a todas las demás. Sus hijos pueden ser elementos de tipo `<defclass></defclass>` y/o `<defsubclass></defsubclass>`.

Etiqueta `<defclass></defclass>`: Representa una clase en la cual se instanciará la información a extraer y que está definida dentro de dicha etiqueta. Puede contener como hijos inmediatos elementos de tipo `<attribute></attribute>` y `<definfo></definfo>`. Es obligatorio que tenga la propiedad `name` a la cual se le da como valor el nombre de la clase que se va a instanciar. También debe tener la propiedad `implement` que tomará una ruta en la forma `Paquete.Modulo` de donde se encuentra la clase implementada. La etiqueta quedaría de la siguiente forma: `<defclass name = "MyClass" implement = "MyPackage.MyModule">... </defclass>`.

Etiqueta `<defsubclass></defsubclass>`: Tiene la misma estructura de una `<defclass></defclass>` pero se diferencia con este nombre pues representa una definición de instancia de clase que es usada para ser asignada como valor de un atributo de una clase de tipo `<defclass></defclass>`. El extractor no ejecutará las definiciones de una `<defsubclass></defsubclass>` si esta no es asignada al atributo de una `<defclass></defclass>`. Al atributo de una `<defsubclass></defsubclass>` se le puede asignar otra `<defsubclass></defsubclass>`. La existencia de esta etiqueta permite la reutilización de las definiciones de información como instancias de clases.

Etiqueta `<attribute></attribute>`: Representa un atributo de la clase que se va a instanciar. La etiqueta requiere la propiedad *param* el cual representa al nombre del parámetro en el constructor de la clase implementada al que corresponde este atributo. Puede además contener hijos de tipo `<definfo></definfo>` o una propiedad *value* o una propiedad *subclass*, cada uno de estos tres elementos es excluyente de los otros dos. La propiedad *subclass* significa que se le dará el valor obtenido de ejecución de la `<subclass></subclass>` con nombre correspondiente a la propiedad *subclass* del atributo. La propiedad *value* se usa para definir un valor por defecto, o sea no se usará ninguna `<definfo></definfo>` o `<defclass></defclass>` para darle valor al atributo.

Ejemplos:

```
<attribute param="param1" value = "valor por defecto"/>
```

`<attribute param="param2" subclass = "Subclase"/>` El atributo recibirá una instancia de la clase Subclase. Debe existir en el XML la definición `<defsubclass name = "Subclase">... </defsubclass>`

```
<attribute param="param1"/><definfo> ... </definfo> </attribute>
```

Recibirá la información extraída mediante las expresiones regulares definidas dentro de `<definfo></definfo>`.

Etiqueta `<definfo> </definfo>`. Contiene una propiedad *format* la cual se usa para poner el formato del contenido al que representa la etiqueta `<definfo></definfo>`. Además tiene tres hijos `<before/>` `<target/>` y `<after/>`.

Etiquetas: `<before/>`, `<target/>` y `<after/>` Poseen una propiedad *regex* que recibe una expresión regular. La etiqueta `<before/>` se usa para representar el patrón de lo que antecede a la información que se desea extraer, `<after/>` por el contrario representa lo que está después, por consiguiente `<target/>` es el patrón de lo que se desea extraer. La propiedad *regex* de la etiqueta `<target/>` no puede ser vacío ya no se extraerá nada con un patrón así. La existencia de estas tres etiquetas permite simplificar la confección de expresiones regulares permitiendo establecer lo que hay antes, lo que hay después y el objetivo por separado.

A continuación un ejemplo de un XML con la estructura propuesta.

```
<main>
```

```
  <defclass name="Clase" implement="Paquete.Modulo">
```

```
    <definfo format="plain">
```

```
      <before regex = ""/>
```

```

        <target regex= ""/>
        <after regex =""/>
    </definfo>
    <attribute param="param1_en_el_construct" value = "valor por defecto"/>
    <attribute param="param2_en_el_construct" subclass = "Subclass"/>
    <attribute param="param3_en_el_construct">
        <definfo format="plain">
            <before regex = ""/>
            <target regex= ""/>
            <after regex =""/>
        </definfo>
    </attribute>
</defclass>
</main>

```

4.3. Diagrama de despliegue

Un diagrama de despliegue muestra las relaciones físicas entre *hardware* y *software* en el sistema final, así como las conexiones entre ellos. (32)

El diagrama de despliegue de la Plataforma está compuesto por un Servidor *Web*, el cual se conecta a máquinas clientes a través del protocolo HTTP, a un servidor de aplicaciones que este a su vez establece una comunicación con máquinas agentes a través del protocolo de comunicación ICE y se conecta además a la base de datos por TCP/IP al igual que el servidor de aplicaciones.

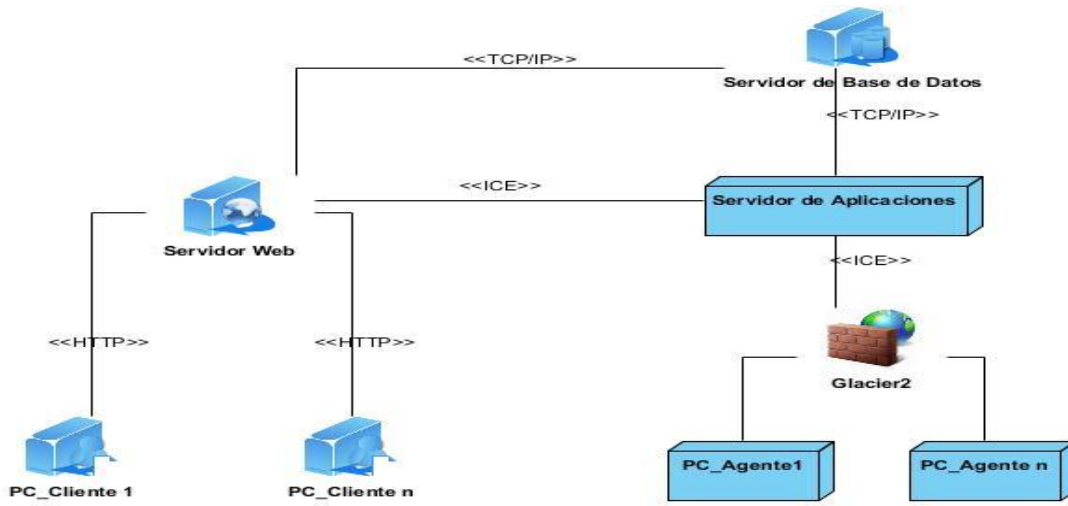


Fig. 8: Diagrama de Despliegue de la Plataforma

4.4. Diagrama de componentes

El diagrama de componentes modela los aspectos físicos de un sistema. Además modela la vista de implementación estática de un sistema y los elementos físicos que residen en un nodo, tales como tablas, librerías, archivos y documentos. (33)

En él se detallan cómo el sistema está desglosado en componentes y las dependencias entre ellos.

4.4.1. Diagrama general del sistema

En el diagrama de componentes del servidor se agregaron los paquetes **ImplementClass** y **Balancer**. Mientras que el agente se agregó el **ImplementClass** y el paquete **Standard**. El paquete Balancer es el que contiene el componente TaskDealer encargado de devolver la lista de los agentes con mejores condiciones para realizar las pruebas. El paquete Standard es el que contiene los componentes encargadas de realizar la extracción de la información de los reportes de prueba, y el ImplementClass es el encargado de crear una relación entre cada una de las herramientas con el sistema.

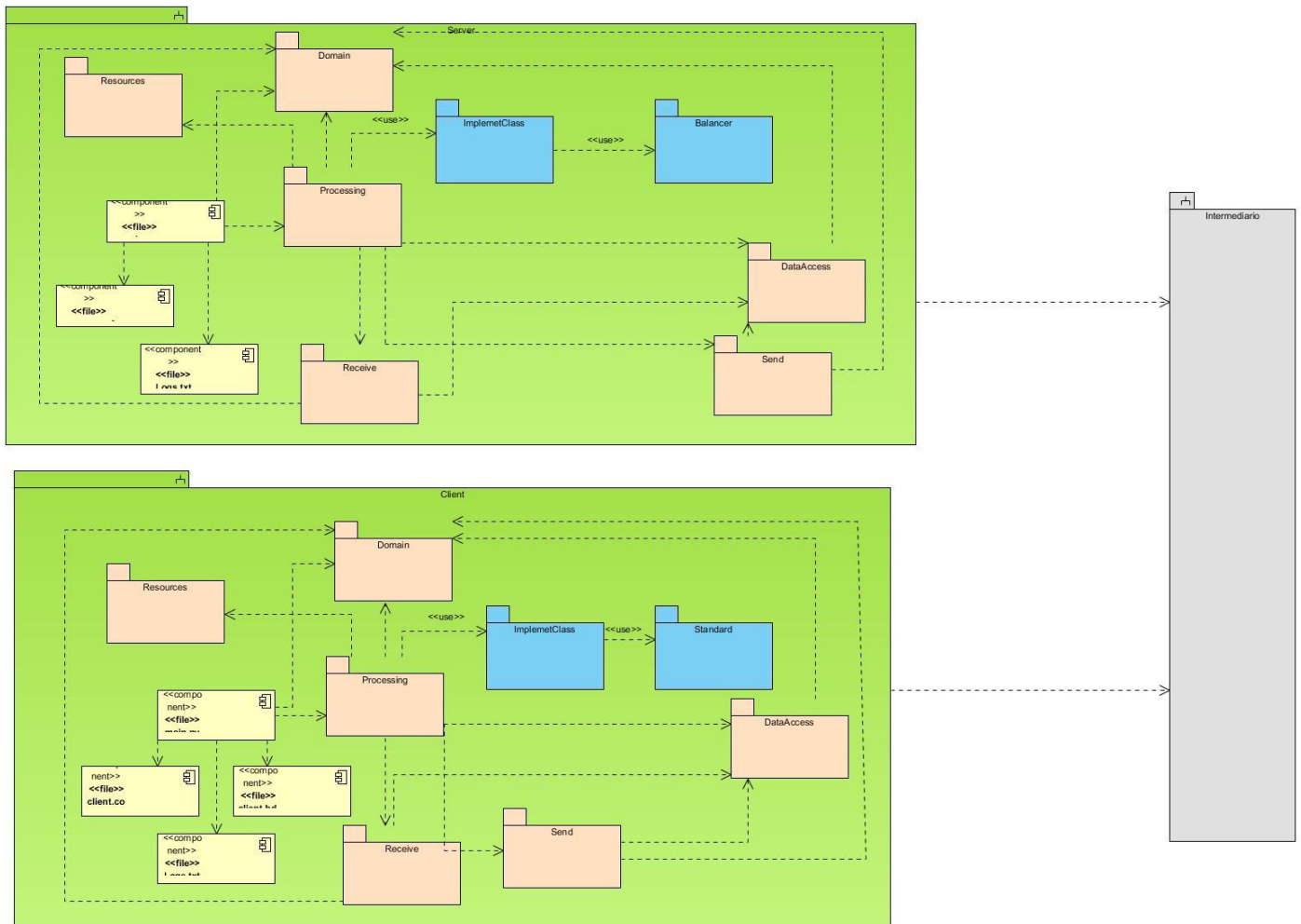


Fig. 9: Diagrama de componentes general del sistema

4.4.2. Diagrama de componentes del servidor paquete Acceso a Datos

En el paquete de Dominio del sistema servidor se agregaron las clases persistentes de la capa para el procesamiento y almacenamiento automatizado de los reportes de prueba como son: **ConfigurationAccesData.py**, **UserAccesData.py**, **RoleAccesData.py**, **GlobalTest.py**, **RequestAccesData.py**, **PermissionAccesData.py**.

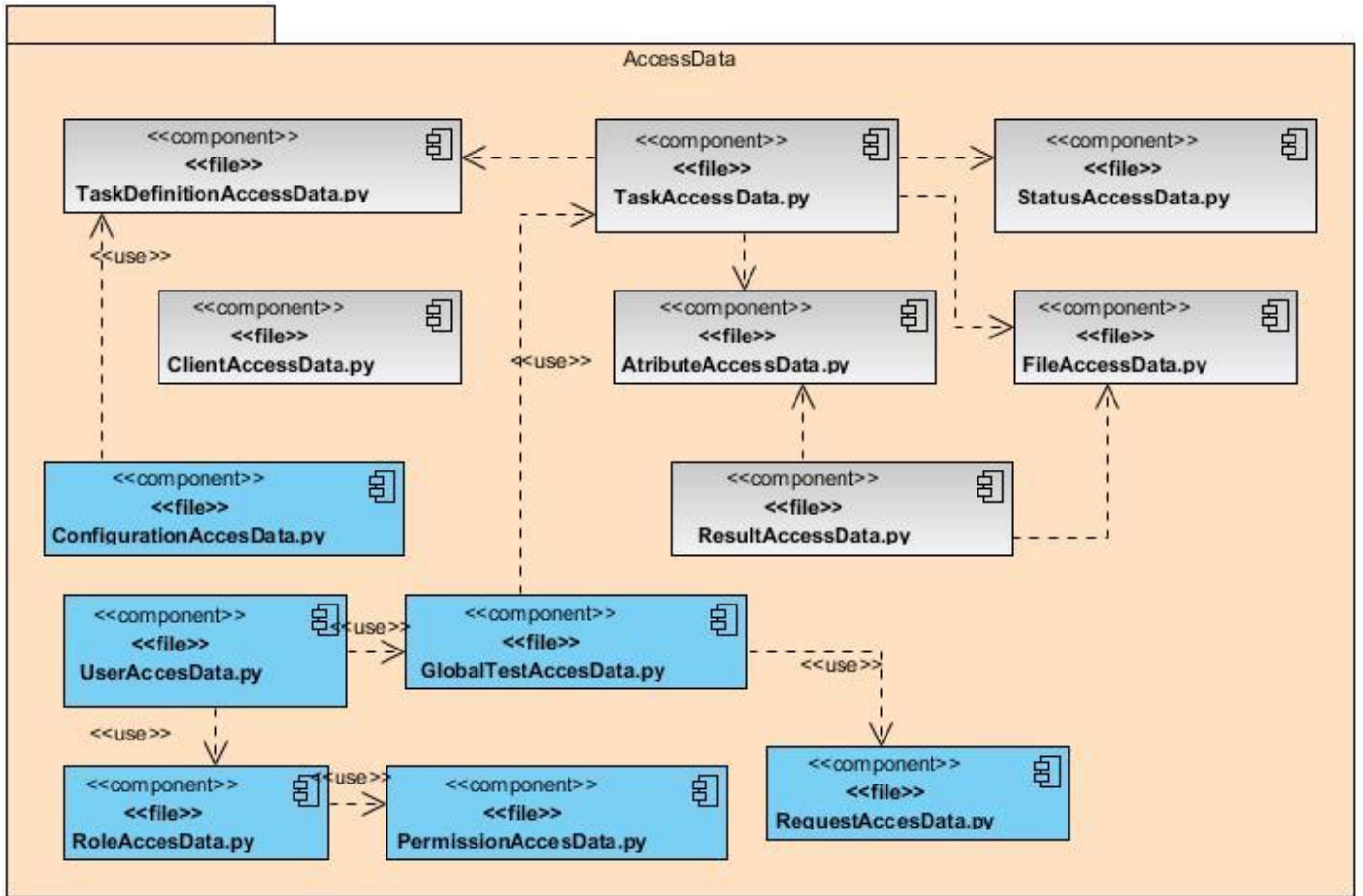


Fig. 10: Diagrama de componentes del servidor paquete Acceso a Datos

4.4.3. Diagrama de componentes del servidor paquete Dominio

En el paquete Dominio del servidor se agregaron los siguientes componentes: **Configuration.py**, **User.py**, **Role.py**, **GlobalTest.py**.

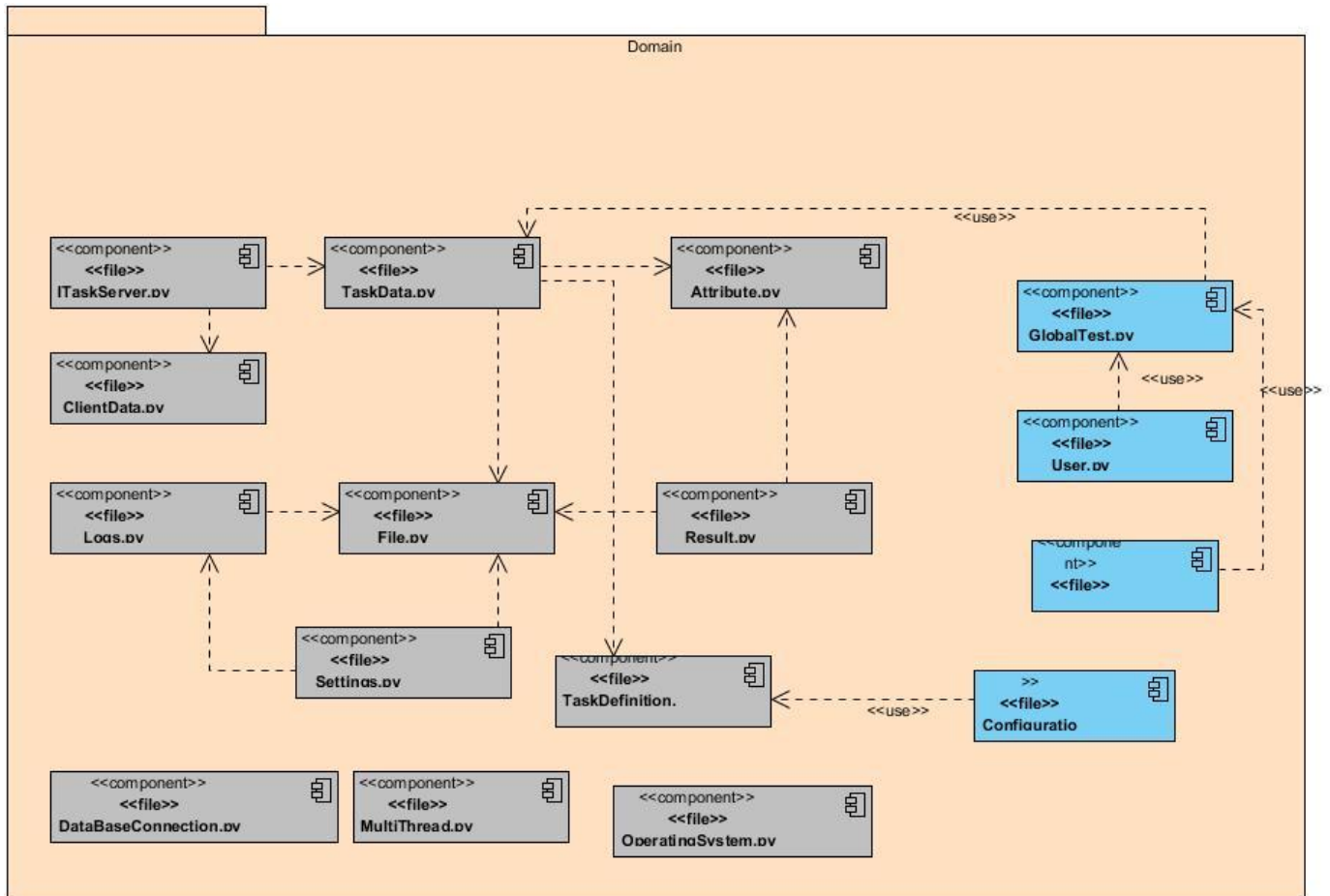


Fig. 11: Diagrama de componentes del servidor paquete Dominio

4.4.4. Diagrama de componentes del servidor paquete ImplementClass

Los componentes del paquete ImplementClass son los encargados de crear una relación entre las herramientas y el sistema.

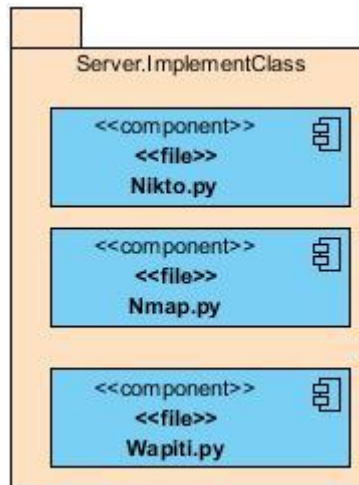


Fig. 12: Diagrama de componentes del servidor paquete ImplementClass

4.4.5. Diagrama de componentes del servidor paquete Balancer

El paquete Balancer es el que contiene el componente el componente TaskDealer encargado de devolver la lista de los agentes con mejores condiciones para realizar las pruebas

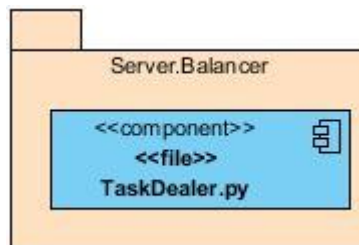


Fig. 13: Diagrama de componentes del servidor paquete Balancer

4.4.6. Diagrama de componentes del cliente paquete ImplementClass

Los componentes del paquete ImplementClass son los encargados de crear una relación entre las herramientas y el sistema.

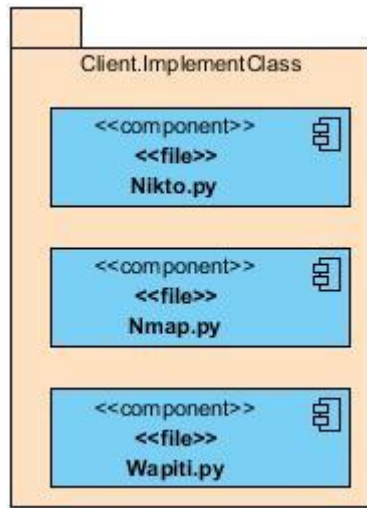


Fig. 14: Diagrama de componentes del cliente paquete ImplementClass

4.4.7. Diagrama de componentes del cliente paquete Standard

En el paquete Standard se encuentran las clases encargadas de realizar la extracción de la información relevante de los reportes de prueba.

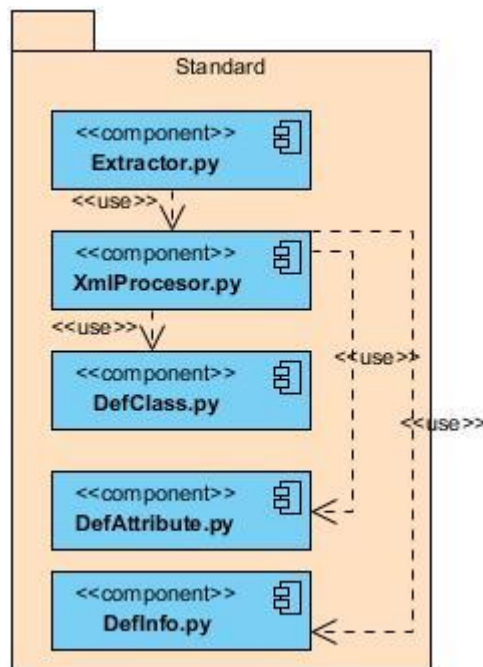


Fig. 15: Diagrama de componentes del cliente paquete Standard

4.5. Pruebas del sistema

El único instrumento adecuado para determinar la calidad de un producto software es el proceso de pruebas. En este proceso se realizan pruebas dirigidas a componentes del software o al mismo en su totalidad, con el objetivo de medir el grado con que el software cumple con los requisitos definidos.

4.5.1. Estrategia de Prueba

Para la realización de las pruebas se creó una estrategia utilizando niveles y métodos de prueba, con el objetivo de encontrar errores que no hayan sido detectados anteriormente.

En los niveles de prueba generales se encuentran:

- Prueba de desarrollador
- Prueba independiente
- Prueba de unidad
- Prueba de integración
- Prueba de sistema
- Prueba de aceptación

La estrategia definida se centra en los niveles, prueba de unidad haciendo uso del método de caja blanca y prueba de integración con el objetivo de verificar que la Plataforma en general funciona correctamente.

4.5.2. Pruebas de Unidad

Las pruebas de unidad es un proceso para probar los subprogramas, los procedimientos individuales o las clases en un programa. Es decir, es mejor probar primero los bloques desarrollados más pequeños del programa, que inicialmente probar el software en su totalidad. Dichas pruebas son en gran parte orientadas a caja blanca, el cual es un método de diseño de casos de prueba que intenta garantizar que se ejecutan al menos una vez todos los caminos independientes de cada módulo, se utilizan las decisiones en su parte verdadera y en su parte falsa, se ejecuten todos los bucles en sus límites y se utilizan todas las estructuras de datos internas. (34)

```

#Pruebas*****
from CPAR.Extractor import Extractor
from CPAR.XmlProcessor import XmlProcessor
from CPAR.Definition import DefClass
xmlFile = "defxmlclass.xml"

#Prueba1 XmlProcessor
listDefClass = XmlProcessor().getObjectClasses(xmlFile)
result = True
for objClass in listDefClass:
    if( not isinstance(objClass, DefClass)):
        result = False
if result:
    print "Prueba1 satisfactoria: Todos los objetos de la lista son de tipo DefClass."

#Prueba2 Extractor
attributes = Extractor().Extract(["file.info"], "plain", xmlFile)
if( not isinstance(attributes[0], DefClass)):
    print
    print "Prueba2 satisfactoria: Objeto de tipo Attribute en posición 0 de la lista."
    print "Estos son sus valores:"
    print "Nombre: "+ attributes[0].getAttributeName()
    print "Valor: "+ attributes[0].getAttributeValue()
    print "Tipo: "+ attributes[0].getAttributeType()

```

Fig. 16: Ejemplo de prueba de unidad

```

Prueba1 satisfactoria: Todos los objetos de la lista son de tipo DefClass.

Prueba2 satisfactoria: Objeto de tipo Attribute en posición 0 de la lista.
Estos son sus valores:
Nombre: informacion_encontrada
Valor: primer valor
Tipo: string

```

Fig. 17: Resultado de las pruebas de unidad

4.5.3. Pruebas de integración

El objetivo de las pruebas de integración es verificar el correcto ensamblaje entre los distintos componentes una vez que han sido probados unitariamente con el fin de comprobar que interactúan correctamente.

Funcionalidad	Condiciones de ejecución	Escenario de prueba	Resultado previsto	Resultado real
Atender solicitud de ejecución de herramientas de pruebas de seguridad de tarea Nikto.	El sistema almacena en la base de datos la prueba recibida.	Envío de petición del Front End al Servidor de Bambú.	Se espera que se muestre en la consola el mensaje "Task has been saved"	Se mostró el mensaje esperado "Task has been saved"
Atender solicitud de ejecución de herramientas de pruebas de seguridad de tarea Nmap.	El sistema almacena en la base de datos la prueba recibida.	Envío de petición del Front End al Servidor de Bambú.	Se espera que se muestre en la consola el mensaje "Task has been saved"	Se mostró el mensaje esperado "Task has been saved"
Atender solicitud de ejecución de herramientas de pruebas de seguridad de tarea Wapiti.	El sistema almacena en la base de datos la prueba recibida.	Envío de petición del Front End al Servidor de Bambú.	Se espera que se muestre en la consola el mensaje "Task has been saved"	Se mostró el mensaje esperado "Task has been saved"

Tabla 6: Tabla de pruebas de integración

4.6. Conclusiones

Una vez finalizada la etapa de implementación se realizaron varias pruebas de unidad y de integración. Para verificar que se cumplió con los requisitos funcionales definidos. Las pruebas realizadas tuvieron resultados satisfactorios quedando así implementada la Plataforma en general.

CONCLUSIONES GENERALES

Una vez concluida la presente investigación queda implementado un sistema capaz de realizar la ejecución de herramientas de prueba de Seguridad Informática de forma distribuida, y que además estandariza y almacena los resultados de las mismas en una base de datos común. Dando cumplimiento a las tareas de la investigación trazadas, así como al objetivo general y los requisitos funcionales y no funcionales definidos al inicio.

RECOMENDACIONES

- ✓ Se sugiere el análisis de esta investigación y su uso como material de consulta.
- ✓ Se recomienda además para futuras versiones la inserción de más herramientas a la Plataforma.
- ✓ Se recomienda la implementación de mejoras en el control de excepciones y respuesta ante fallos.

REFERENCIA BIBLIOGRÁFICA

1. slideshare. *slideshare*. [En línea] [Citado el: 13 de Noviembre de 2011.] <http://www.slideshare.net/lepenago/las-tics-y-el-mundo-actual-presentation>.
2. **Alberto Carlos Carbonell Marcé, Carlos Manuel Morciego González.** *Plataforma de Distribución de Tareas*. La Habana : s.n., 2011.
3. **Rolando Alfredo Hernández León, Sayda Coello.** *EL PROCESO DE INVESTIGACIÓN CIENTÍFICA*. Ciudad de La Habana : Editorial Universitaria cubana, 2011. 978-959-16-.
4. **Aguirre, Jorge Ramió.** *Expectativas de Desarrollo en Seguridad de la Información*. [Presentación] España : s.n., 2009.
5. **Avilés, MSc. Julio Rito Vargas.** Conceptos básicos de auditoría informática. [En línea] 2009. [Citado el: 15 de Noviembre de 2011.] <http://jrvargas.files.wordpress.com/2009/03/conceptos-basicos-de-auditoria-informatica.pdf>.
6. **Centro Empresarial Madrid 92.** ESA Security. [En línea] 2006. [Citado el: 15 de Noviembre de 2011.] <http://www.esa-security.com/web/servicios/hacking.htm>.
7. **NCC Group .** NGSsecure. [En línea] 2010. [Citado el: 18 de Noviembre de 2011.] <http://www.ngssecure.com/>.
8. **NCC Group.** NGSsecure. [En línea] 2010. [Citado el: 18 de Noviembre de 2011.] <http://www.ngssecure.com/>.
9. **BASE4 Security S.A.** BASE4 Security. [En línea] 2010. [Citado el: 18 de Noviembre de 2011.] <http://base4sec.com/>.
10. **Security Solutions.** Security Solutions. [En línea] 2011. [Citado el: 21 de Noviembre de 2011.] <http://www.trustware.com/About-Trustware/>.
11. **softwarelibrevenezuela.** [En línea] 23 de Marzo de 2009. [Citado el: 1 de Diciembre de 2011.] <http://softwarelibrevenezuela.blogspot.es/>.
12. **Desconocido.** [En línea] [Citado el: 2 de Diciembre de 2011.] <http://yaqui.mx1.uabc.mx/~molguin/as/RUP.htm>.
13. **Perovich, Daniel.** [En línea] [Citado el: 2 de Diciembre de 2011.] http://www.google.com.cu/url?sa=t&rct=j&q=por+que+utilizar+rup&source=web&cd=3&ved=0CCsQFjAC&url=https%3A%2F%2Fwww.u-cursos.cl%2Fingenieria%2F2008%2F1%2FCC61H%2F1%2Fmaterial_docente%2Fobjeto%2F161148&ei=eV7aTo37GKn00gHe1aH6DQ&usg=AFQjCNFm26ucFLRdklo8GnN6t.
14. **Object Management Group, Inc.** *OMG Unified Modeling Language Specification*.
15. **Desconocido.** [En línea] 7 de Noviembre de 2011. [Citado el: 2 de Diciembre de 2011.] <http://www.visual-paradigm.com/product/vpuml/>.
16. **Martínez, Jairo Chapela.** Introducción al entorno de desarrollo Eclipse. [En línea] 26 de Septiembre de 2007. [Citado el: 3 de Diciembre de 2011.] http://www-gris.det.uvigo.es/wiki/pub/Main/MiscResources/Manual_Eclipse.pdf.
17. **Martínez, Rafael.** Sobre PostgreSQL. [En línea] 2 de Octubre de 2010. [Citado el: 3 de Diciembre de 2011.] http://www.postgresql.org.es/sobre_postgresql.
18. **Desconocido.** *PgAdmin - Excelente Gestor de Postgresql*. [Documento] 2007.

19. **Alvarez, Miguel Angel.** [En línea] [Citado el: 3 de Diciembre de 2011.] <http://www.desarrolloweb.com/articulos/1325.php>.
20. **Maldonado, Daniel Martín.** [En línea] 1 de Julio de 2008. [Citado el: 4 de Diciembre de 2011.] <http://www.aplicacionesempresariales.com/sqlite-el-motor-de-base-de-datos-agil-y-robusto.html>.
21. **Beltrán, Marta.** Red de Conocimiento en Informática Industrial y Aplicaciones de Gestión en Tiempo Real. [En línea] 27 de Noviembre de 2008. [Citado el: 4 de Diciembre de 2011.] <http://redindustria.blogspot.com/2008/11/qu-es-un-middleware.html>.
22. **Fernández, David Vallejo.** [En línea] Diciembre de 2006. [Citado el: 5 de Diciembre de 2011.] <http://www.esi.uclm.es/www/David.Vallejo/docs/docICE.pdf>.
23. [En línea] Diciembre de 2006. [Citado el: 4 de Diciembre de 2011.] <http://www.esi.uclm.es/www/dvallejo/docs/docICE.pdf>.
24. Mozilla developer network. [En línea] [Citado el: 5 de Diciembre de 2011.] https://developer.mozilla.org/en/WebSockets/Writing_WebSocket_client_applications.
25. [En línea] [Citado el: 7 de Diciembre de 2011.] <http://packages.python.org/pyparsing/>.
26. [En línea] 14 de Diciembre de 2007. [Citado el: 7 de Diciembre de 2011.] http://migueljaque.com/index.php/tecnicas/tecnicasmodnegocio/37-modelado_negocio/46-modelo-de-dominio?tmpl=component&print=1&page=.
27. scribd.com. [En línea] [Citado el: 10 de Diciembre de 2011.] <http://es.scribd.com/doc/58516536/Requerimientos-Funcionales-y-No-Funcionales>.
28. *Estándares de codificación para Python v1.doc*.
29. **Garlan, David y Shaw Mary.** *An Introduction to Software Architecture*.
30. **Dainelvis Martínez Basulto, Miguel Ángel Torres Pérez.** *Portal Comercial de Contenidos y Servicios de la Plataforma de Gestión de Contenidos para Dispositivos Móviles (PGCDM)*. Habana : s.n., 2010.
31. [En línea] [Citado el: 20 de Mayo de 2012.] http://sophia.javeriana.edu.co/~lcdiaz/ADOO2006-3/grasp_cpaternostro-lvargas-jviafara.pdf.
32. [En línea] 2012. [Citado el: 17 de Mayo de 2012.] <http://es.scribd.com/doc/19808824/diagramas-de-despliegue-2222>.
33. **Daniele, Ing. Marcela.** [En línea] 2007. [Citado el: 17 de Mayo de 2012.] <http://fineans.usac.edu.gt:8001/rid=1HV0BP15X-15DBYBZ-FH/UML-diagramaComponentes.pdf>.
34. **Grupo ARQUISOFT.** [En línea] 2007. [Citado el: 25 de Mayo de 2012.] <http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node30.html>.
35. Segurmática consultoría y seguridad informática. [En línea] [Citado el: 4 de Diciembre de 2011.] <http://www.segurmatica.cu/>.

BIBLIOGRAFÍA

1. **Eco, Umberto.** *Como se hace una tesis.* Ciudad México: Sedisa, 1984.
2. **Tesis.UCI.** [Online]. 2011 Disponible en: <http://tesis.uci.cu>
3. **Biblioteca.UCI.**[Online] 2011. Disponible en: <http://biblioteca2.uci.cu>
4. **Diccionario.** [Online] 2011 Disponible en: <http://www.deperu.com/diccionario/>
5. **El proceso de investigación científica** [Online] Disponible en: <http://eva.uci.cu>
6. **Española, Diccionario Manual de la Lengua.** *Diccionario Manual de la Lengua Española.* [Online] 2007.
7. **¿Qué es un Sistema Gestor de Bases de Datos o SGBD.** [Online] Disponible en: <http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/>.
8. Características PostgreSQL. [Online] Disponible en: <http://www.postgresql.org>
9. Vallejo Fernández, David. Documentación de ZeroC ICE. 2006.
10. **Sommerville.** Ingeniería de Software. Bogotá: McGraw-Hil, 2008.
11. **Ivar jacobson,Grady booch, James rumbaugh** [Online] 2011 Disponible en: <ftp://10.0.0.22/documentacion/Ingenieria%20Software/RUP/books/EI%20proceso%20unificado%20de%20desarrollo%20de%20software/>
12. **Visual Paradigm,**[Online] 2011. Disponible en <ftp://10.0.0.22/documentacion/Ingenieria%20Software/Visual%20Paradigm/Introducci%F3n%20a%20VP-UML.ppt>
13. **UML,** [Online] 2011. Disponible en: <ftp://10.0.0.22/documentacion/Ingenieria%20Software/UML/books/Unified%20modeling%20language%20specification/10020>
14. **Daniel Martin Maldonado,** [Online] Disponible en:<http://www.aplicacionesempresariales.com/sqlite-el-motor-de-base-de-datos-ajil-y-robusto.html>
15. **Modelo de dominio.** [Online] Disponible en: <http://lsi.ugr.es>

16. Erich Gamma, R.H., Ralph Johnson, John Vlissides, Patrones de Diseño. Elementos de software orientado a objetos reutilizables. España.
17. Yeni Morgado Sánchez, A.C.A., Guía para la personalización de PostgreSQL 8.4. 2010, Universidad de las Ciencias Informáticas: Cuba.
18. Víctor Theoktisto, Patrones de Diseño.
19. **Pressman**. Estrategia de Pruebas del software capítulo 13

ANEXOS

ANEXO 1 DESCRIPCIÓN DE LOS REQUISITOS FUNCIONALES

Servidor

RF1. Atender solicitud de ejecución de herramientas de pruebas de seguridad.

Este requisito es el encargado de recibir y recepcionar la solicitud de la prueba enviada desde el Front End.

RF2. Distribuir tareas atendiendo a los criterios de hilos de ejecución disponibles.

Este requisito es el encargado de distribuir la ejecución de la prueba hacia las maquinas agentes con la mayor cantidad de hilos de ejecución disponibles.

RF3. Almacenar el contenido del reporte estandarizado en la base de datos.

Este requisito se encarga de almacenar en la base de datos el resultado de la prueba estandarizado.

RF4. Notificar estado de culminación de la prueba.

Este requisito es el encargado de enviar una notificación al Front End para que muestre el reporte de la prueba realizada.

Agente

RF1. Procesar contenido del reporte de la herramienta de pruebas de seguridad.

Este requisito es el encargado de realizar la extracción de la información relevante de los reportes de prueba generados, para lo cual se necesita:

Cargar contenido del archivo de definición de información.

Leer el fichero del reporte de la herramienta de pruebas de seguridad.

Generar resultado estandarizado.

RF2. Enviar resultado estandarizado al servidor.

Este requisito es el encargado de enviar el fichero del resultado de la prueba y el resultado estandarizado al servidor.

ANEXO 2 DESCRIPCIÓN DEL MODELO LÓGICO DE LA BASE DE DATOS.

Nombre: File		Nombre: Result	
Descripción: Contiene los datos de los archivos del resultado.		Descripción: Contiene los datos del resultado de la prueba.	
Atributos:		Atributos:	
Nombre:	Tipo de Dato:	Nombre	Tipo de Dato
file_id	string	result_id	string
file_name	string	result_timeSend	string
file_path	string		
Nombre: Attribute		Nombre: Task_Data	
Descripción: Almacena los atributos del reporte.		Descripción: Contiene los parámetros de la herramienta.	
Atributos:		Atributos:	
Nombre:	Tipo de Dato:	Nombre:	Tipo de Dato:
attribute_id	Int	task_id	string
attribute_value	string	task_parameters	string
attribute_name	string	task_replay	string
attribute_type	string	task_startTime	string
		task_default	string
Nombre: Task_Definition		Nombre: Client	
Descripción: Contiene los datos de configuración de la herramienta.		Descripción: Contiene los datos del cliente encargado de almacenar las herramientas.	
Atributos:		Atributos:	
Nombre:	Tipo de Dato:	Nombre:	Tipo de Dato:
definition_id	int	client_ip	string
definition_aplicationName	string	client_freeProcess	int
definition_aplicationPath	string		
definition_implementClass	string		
Nombre: Status		Nombre: User	
Descripción: Almacena los estados por los que va a transitar cada tarea.		Descripción: Contiene los datos del usuario	
Atributos:		Atributos:	
Nombre:	Tipo de Dato:	Nombre:	Tipo de Dato:
status_id	int	user_id	string
status_type	string	conected	boolean
Nombre: Role		Nombre: Permission	
Descripción: Almacena los nombres de los roles que tendrá la plataforma.		Descripción: Almacena las opciones a las que va a tener acceso cada rol.	

Atributos:		Atributos:	
Nombre:	Tipo de Dato:	Nombre:	Tipo de Dato:
role_id	int	permission_id	int
role_name	string		
Nombre: Request		Nombre: Global Test	
Descripción: Contiene los datos de la solicitud de prueba de un cliente.		Descripción: Contiene todos los datos de las pruebas globales	
Atributos:		Atributos:	
Nombre:	Tipo de Dato:	Nombre:	Tipo de Dato
request_id	int	test_id	int
petitioner_name	varchar	test_name	string
entity_name	varchar	test_description	string
date	varchar	test_startTime	string
		test_finishTime	string
Nombre: Configuration		Nombre: LocalUser	
Descripción: Contiene los datos de configuración de la herramienta.		Descripción: Contiene los datos de los usuarios locales de la plataforma	
Atributos:		Atributos	
Nombre:	Tipo de Dato:	Nombre:	Tipo de Dato:
configuration_id	int	pass	string
configuration_name	string	name	string
configuration_value	string		
configuration_type	string		

Tabla 7: Descripción de del diagrama lógico de la Base de Datos.

ANEXO 3 DESCRIPCIÓN DEL MODELO FÍSICO DE LA BASE DE DATOS

Nombre: File				
Descripción: Contiene los datos del archivo del resultado				
Nombre:	Tipo de Dato:	PK/FK	¿Nulo?	Descripción:
file_id	varchar(255)	PK	No	Identificador de la tabla File.
Resultresult_id	varchar(255)	FK	No	Llave foránea que surge debido a la relación que existe entre la tabla File y la tabla Result.
file_name	varchar(255)		No	Campo que define el nombre del archivo.
file_path	varchar(255)		No	
Nombre: Result				
Descripción: Contiene los datos del resultado de la prueba				
Nombre:	Tipo de Dato:	PK/FK	¿Nulo?	Descripción:
result_id	varchar(255)	PK	No	Identificador de la tabla Result.
TaskDatatask_id	varchar(255)	FK	No	Llave foránea que surge debido a la relación que existe entre la tabla Result y la tabla TaskData.
result_timeSend	varchar(255)		No	
Nombre: TaskData				
Descripción: Contiene los parámetros de la herramienta				
Nombre:	Tipo de Dato:	PK/FK	¿Nulo?	Descripción:
task_id	varchar(255)	PK	No	Identificador de la tabla TaskData.
GlobalTesttest_id	int(4)	FK	No	Llave foránea que surge debido a la relación que existe entre la tabla TaskData y la tabla GlobalTest.

Useruser_id	varchar(255)	FK	No	Llave foránea que surge debido a la relación que existe entre la tabla TaskData y la tabla User.
Statusstatus_id	varchar(255)	FK	No	Llave foránea que surge debido a la relación que existe entre la tabla TaskData y la tabla Status.
TaskDefinitiondefinition_id	int(4)	FK	No	Llave foránea que surge debido a la relación que existe entre la tabla TaskData y la tabla TaskDefinition.
task_parameters	varchar(255)		No	Campo que contiene los parámetros de la tarea.
task_replay	varchar(255)		No	
task_default	varchar(255)		No	
task_type	Varchar(255)		No	Campo que contiene el tipo de la tarea.

Nombre: TaskDefinition

Descripción: Contiene los datos de configuración de la herramienta

Nombre:	Tipo de Dato:	PK/FK	¿Nulo?	Descripción:
definition_id	int(4)	PK	No	Identificador de la tabla TaskDefinition.
definition_applicationName	varchar(255)		No	
definition_applicationPath	varchar(255)		No	
definition_implementClass	varchar(255)		No	

Nombre: Attribute

Descripción: Almacena los atributos del reporte

Nombre:	Tipo de Dato:	PK/FK	¿Nulo?	Descripción:
attribute_id	int(4)	PK	No	Identificador de la tabla

				Attribute
resultresult_id	varchar(255)	FK	No	Llave foránea que surge debido a la relación que existe entre la tabla Attribute y la tabla Result.
attribute_value	varchar(255)		No	Campo que contiene el valor del atributo
attribute_name	varchar(255)		No	Campo que contiene el nombre del atributo
attribute_type	varchar(255)		No	Campo que contiene el tipo del atributo.
Nombre: Status				
Descripción: Almacena los estados por los que va a transitar cada tarea				
Nombre:	Tipo de Dato:	PK/FK	¿Nulo?	Descripción:
status_id	int(4)	PK	No	Identificador de la tabla Status.
status_type	Varchar(255)		No	Contiene el tipo de estado en que se encuentra la prueba.
Nombre: Configuration				
Descripción: Contiene los datos de configuración de la herramienta.				
Nombre:	Tipo de Dato:	PK/FK	¿Nulo?	Descripción:
configuration_id	int(4)	PK	No	Identificador de la tabla Configuration.
TaskDefinitiondefinition_id	int(4)	FK	No	Llave foránea que surge debido a la relación que existe entre la tabla Configuration y la tabla TaskDefinition.
configuration_name	varchar(255)		No	
configuration_value	Varchar(255)		No	
configuration_type	Varchar(255)		No	
Nombre: GlobalTest				

Descripción: Almacena los datos de las pruebas globales.				
Nombre:	Tipo de Dato:	PK/FK	¿Nulo?	Descripción:
test_id	int(4)	PK	No	Identificador de la tabla GlobalTest.
Useruser_id	varchar(255)		No	Llave foránea que surge de la relación que existe entre la tabla GlobalTest y la tabla User.
test_name	varchar(255)			Campo que contiene el nombre de la prueba.
test_description	varchar(255)			Campo que contiene la descripción de la prueba.
test_startTime	varchar(255)			Campo que contiene la hora de inicio de la prueba.
test_finishTime	varchar(255)			Campo que contiene la hora de fin de la prueba.
Nombre: Client				
Descripción: Contiene los datos de la				
Nombre:	Tipo de Dato:	PK/FK	¿Nulo?	Descripción:
client_ip	varchar(255)	PK	No	Identificador de la tabla Client.
client_freeProcess	int(4)		No	
Nombre: User				
Descripción: Contiene los datos del usuario				
Nombre:	Tipo de Dato:	PK/FK	¿Nulo?	Descripción:
user_id	Varchar(255)	PK	No	Identificador de la tabla User
vulnerability	bool		No	Nombre de la vulnerabilidad
domain	Varchar(255)		No	

Nombre: LocalUser				
Descripción: Usuario local de la Plataforma.				
Nombre:	Tipo de Dato	PK/FK	¿Nulo?	Descripción:
Useruser_id	Varchar(255)	FK	No	Llave foránea que surge de la relación que existe entre la tabla User y la tabla LocalUser.
pass	varchar(255)		No	Contraseña del usuario local para acceder a la Plataforma.
name	varchar(255)		No	
Nombre: Role				
Descripción: Contiene los roles que podrán ser asignados.				
Nombre:	Tipo de Dato:	PK/FK	¿Nulo?	Descripción:
role_id	int(4)	PK	No	Identificador de la tabla Role.
role_name	varchar(255)			Nombre del rol
Nombre: Permission				
Descripción: Contiene los permisos que podrá tener el usuario				
Nombre:	Tipo de Dato:	PK/FK	¿Nulo?	Descripción:
permission_id	int(4)	PK	No	Identificador de la tabla Permission.
type	varchar(255)		No	Tipo de permiso
Nombre: Request				
Descripción: Contiene los datos de la solicitud de prueba de un cliente.				
Nombre:	Tipo de Dato:	PK/FK	¿Nulo?	Descripción:
request_id	int(4)	PK	No	Identificador de la tabla Request.
GlobalTesttest_id	int(4)		No	Llave foránea que surge de la relación que existe entre la tabla Request y la tabla GlobalTest.
petitioner_name	varchar(255)		No	Nombre de la petición.

entity_name	varchar(255)		No	Nombre de la entidad.
date	varchar(255)		No	Datos.

GLOSARIO

----A----

Agente: es una aplicación informática o una computadora que accede a un servicio remoto en otra computadora, conocida como servidor, normalmente a través de una red de telecomunicaciones.

Archivo: Unidad o conjunto de información de todo tipo que se almacena en algún medio de escritura que varía según la extensión, es decir, las letras que se escriben después del último punto.

Arquitectura: es el diseño de más alto nivel de la estructura de un sistema.

----F----

Front-End: En diseño de software el front-end es la parte del software que interactúa con el o los usuarios

----H----

Herramienta Case: herramientas utilizadas para el desarrollo de proyectos de Ingeniería de Software.

----I----

IDE: Entorno de desarrollo integrado es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien poder utilizarse para varios.

ICE: es un middleware ligero, orientado a objetos, es decir, ICE proporciona herramientas, APIs y soporte de bibliotecas para construir aplicaciones cliente-servidor orientadas a objetos.

----L----

Librería: es un conjunto de subprogramas utilizados para desarrollar software. Las librerías contienen código y datos, que proporcionan servicios a programas independientes, es decir, pasan a formar parte de estos.

----M----

Middleware: se ha optado por no traducir el término. Puede ser definido como aquel software que funciona como intermediador entre diferentes tipos de software y hardware de modo que éstos puedan trabajar en conjunto dentro de una red.

----N----

Nikto: es una herramienta de escaneo de servidores web que se encarga de efectuar diferentes tipos de actividades tales como, detección de malas configuraciones y vulnerabilidades en el servidor objetivo.

Nmap: es un programa de código abierto que sirve para efectuar rastreo de puertos.

----P----

Plugin: es un módulo de software que añade una característica o un servicio específico a un sistema más grande.

----R----

Rup: Rational Unified Process traducido al español, Proceso Unificado de Desarrollo.

----S----

Servidor: Usualmente se trata de un dispositivo de software que proporciona servicios de aplicación a las computadoras cliente. Un servidor de aplicaciones generalmente gestiona la mayor parte (o la totalidad) de las funciones de lógica de negocio y de acceso a los datos de la aplicación.

----U----

UML: Unified Modeling Language o Lenguaje Unificado de Modelado. Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad.

----V----

Vulnerabilidades: son puntos débiles del software que permiten que un atacante comprometa la integridad, disponibilidad o confidencialidad del mismo. Algunas de las vulnerabilidades más severas permiten que los atacantes ejecuten código arbitrario, denominadas vulnerabilidades de seguridad, en un sistema comprometido.

----W----

Wapiti: es un escáner de vulnerabilidades para aplicaciones web