UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

# Modelo base de conocimiento para auditorías de seguridad en Servicios Web con SQL Injection

**Modalidad: Profundización**

**Autor**
**JOHN EDISON MORENO MARÍN**

**Universidad Distrital Francisco José De Caldas**
**Maestría en Ciencias de la Información y las Comunicaciones**
**Énfasis en Ingeniería de Software**
**Bogotá, Colombia**
**Octubre de 2019**

# Modelo base de conocimiento para auditorías de seguridad en Servicios Web con SQL Injection

**JOHN EDISON MORENO MARIN**

Tesis presentada como requisito para optar al título de:
**Magister en Ciencias de la Información y las Comunicaciones**

Director:
**Ing. MSc. PAULO CESAR CORONADO SANCHEZ**
Grupo de Investigación:
Grupo de Investigación de Telemedicina GITEM++

**Universidad Distrital Francisco José De Caldas**
**Maestría en Ciencias de la Información y las Comunicaciones**
**Énfasis en Ingeniería de Software**
**Bogotá D.C., Colombia**
**Octubre de 2019**

**Agradecimientos**

A la Universidad Distrital, por brindarme la oportunidad de formarme todos estos años y permitirme llevar con éxito éste y otros proyectos más.

A mi familia, por su atención, tiempo y comprensión los cuales me ayudaron emocionalmente para sacar adelante este proyecto.

A mis compañeros de Maestría, por sus observaciones y aportes, que permitieron fortalecer mis conocimientos para culminar el trabajo.

Al profesor Paulo Cesar Coronado Sánchez - Director del proyecto, por su dirección, aportes y colaboración que hicieron posible realizar con éxito este propósito.

## RESUMEN

En este proyecto se presenta un modelo de base de conocimiento para el desarrollo de auditorías de seguridad que verifiquen el estado de vulnerabilidad respecto a vectores de ataque de inyección SQL en servicios web, donde, dicha base está orientada en ontologías de seguridad de la información, organizada lógicamente con una jerarquía o taxonomía no secuencial, es decir se puede acceder a un contenido específico con alguna referencia; como prototipo de objeto de auditoría se utilizó una aplicación llamada "Factura Electrónica". Dicho modelo fue implementado extrayendo partes de metodologías, métodos, estándares, buenas prácticas y herramientas enmarcadas dentro del Consorcio Internacional de Certificación de Seguridad de Sistemas de Información (ISC2), la metodología abierta de comprobación para la seguridad (OSSTMM), el proyecto abierto de seguridad de aplicaciones web (OWASP) y la estrategia de Gobierno en Línea (GEL) del Ministerio de las Tecnologías de información y las comunicaciones (MinTIC) de Colombia, tales como ISO y OpenUP (para la fase de desarrollo), implementadas en un plazo determinado según el cronograma propuesto en el proyecto.

Este proyecto bajo la modalidad de profundización, ejecuta una prueba del modelo en el marco del área de seguridad de la información y específicamente la seguridad informática, para mitigar ataques de tipo SQL Injection (SQLi) como vector de ataque de éste proyecto. La taxonomía del modelo se desarrolló con ideas propias y extrayendo ideas de varias investigaciones previas las cuales se irán referenciando a medida que se va leyendo el documento. (Rodríguez, 2015, p. 10; Contreras, 2017, p. 11; Farías e Infante, 2005, p. 1 & Amirmohammad, Mazdak y Azizah, 2013, p. 1)

## PALABRAS CLAVE

Modelo de información, seguridad de la información, seguridad informática, servicios web, auditoría de aplicaciones web, inyección SQL, base de conocimiento, ontologías, taxonomías.

**ABSTRACT**

This project presents a knowledge base model for the development of security audits that verify the vulnerability status with respect to SQL injection attack vectors in web services, where, said base is oriented in information security ontologies, logically organized with a hierarchy or non-sequential taxonomy, that is, specific content can be accessed with some reference; As an audit object prototype, an application called "Electronic Invoice" was used. This model was implemented by extracting parts of methodologies, methods, standards, good practices and tools framed within the International Information System Security Certification Consortium (ISC2), the Open Source Security Testing Methodology Manual (OSSTMM), the Open Web Application Security Project (OWASP) and the Online Government Strategy (GEL) of the Ministry of Information Technology and Communications (MinTIC) of Colombia, such as ISO and OpenUP (for the development phase), implemented within a period of time determined according to the schedule proposed in the project.

This project under the deepening modality, executes a test of the model within the area of information security and specifically cyber security, to mitigate attacks of type SQL Injection (SQLi) as an attack vector of this project. The taxonomy of the model was developed with own ideas and extracting ideas from several previous investigations which will be referenced as the document is read. (Rodríguez, 2015, p. 10; Contreras, 2017, p. 11; Farías e Infante, 2005, p. 1 & Amirmohammad, Mazdak and Azizah, 2013, p. 1)

**KEYWORDS**

Information model, information security, computer security, web services, web application auditing, SQL Injection, knowledge base, ontologies, taxonomies.

**CONTENIDO**

**1. INTRODUCCIÓN 11**

1.1 Definición del problema 12

1.2 Pregunta de investigación 12

1.3 Alcances 13

1.4 Objetivos 13

1.4.1 Objetivo General 13

1.4.2 Objetivos específicos 14

1.5 Justificación 14

**2. ESTADO DEL ARTE 16**

2.1 Marco teórico 16

2.1.1 Modelo de auditoría de conocimiento para la seguridad de la información 16

2.1.2 Modelo de gestión del conocimiento para mejorar la seguridad de la información 17

2.1.3 Desarrollo de un sistema experto para la auditoría de seguridad de la información 17

2.1.4 Creación de una base de conocimientos para un sistema experto en seguridad de la información 18

2.1.5 Un marco de conocimiento para el modelado de seguridad de la información 18

2.1.6 Un método de construcción de base de conocimiento semántico para la seguridad de la información 19

2.1.7 Un modelo de intercambio de conocimientos sobre seguridad de la información en las organizaciones 19

2.1.8 Seguridad de la información centrada en el conocimiento 20

2.1.9 Taxonomía de las técnicas de detección y prevención de inyecciones SQL 20

2.1.10 Taxonomía de los ataques de inyección SQL 21

2.1.11 La inyección SQL sigue viva: un estudio sobre técnicas de evasión de firma 21

2.1.12 Auditoría de aplicaciones web: metodología y práctica profesional 22

2.1.13 Propuesta de auditoría a las aplicaciones web de la empresa C&M consultores 22

2.1.14 Seguridad informática en web services 23

2.1.15 Pruebas de penetración con Kali linux: inyección SQL, XSS y Wordpress 23

2.1.16 Evaluación de la seguridad de la aplicación web mediante la gestión de errores. 23

2.1.17 Uso del análisis estático para encontrar errores. 24

2.1.18 Sania: Análisis sintáctico y semántico para pruebas automatizadas. 24

2.1.19 Implementación de técnicas y recomendaciones de seguridad OWASP. 24

2.1.20 Asegurar la aplicación web de los ataques de inyección de SQL 25

2.1.21 OwlEye: un sistema avanzado de detección de ataques web basado en HMM 25

2.2 Marco conceptual 26

2.2.1 Modelo y modelo de información 26

2.2.2 Seguridad de software 26

2.2.3 WS-Security 27

2.2.4 Amenazas 27

2.2.5 Análisis de Riesgo 27

2.2.6 Auditoría 27

2.2.7 Ciberseguridad 27

2.2.8 Gestión de incidentes de seguridad de la información 28

2.2.9 Plan de tratamiento de riesgos 28

2.2.10 Privacidad 28

2.2.11 Riesgo 28

2.2.12 Seguridad de la información 28

2.2.13 Sistema de Gestión de Seguridad de la Información SGSI 28

2.2.14 Vulnerabilidad 29

2.2.15 SQL 29

2.2.16 SQL Injection 29

2.2.17 Base de Conocimiento 29

2.2.18 OWASP 30

2.2.19 Falla de inyección SQL 30

2.2.20 Vulnerabilidad inyección SQL 31

2.2.21 LDAP 32

2.2.22 Inyección LDAP 32

2.2.23 Inyección XPath 32

2.2.24 Analizadores XML 33

2.2.25 Agujero de Seguridad 33

2.2.26 Ontología 33

**3. METODOLOGÍA 34**

3.1 Metodología general del modelo base de conocimiento 34

3.1.1 Primera Etapa: Recolección del conocimiento 35

3.1.2 Segunda Etapa: Implementación del prototipo 35

3.1.3 Elementos de evaluación de un modelo de base de conocimiento 36

3.2 Metodología para ejecutar la Auditoría en Seguridad web 39

3.2.1 Introducción, identificación, análisis y detección de ataques sql injection 41

3.2 Metodología de desarrollo de Software 61

**4. MODELO DE ARQUITECTURA DEL MODELO BASE DE CONOCIMIENTO 63**

4.1 Descripción General 63

4.2 Términos y condiciones de uso 63

4.3 Descripción del Modelo 63

4.4 Alcance del Modelo 63

4.5 Representación Arquitectónica del Modelo 64

4.5.1 Arquitectura Base de Conocimiento en OpenKM: cliente/servidor 64

4.5.2 Arquitectura aplicación con Web Services: estilo de capas - subestilo cliente-servidor 64

4.6 Arquitectura del modelo base de conocimiento para auditoría de Web Service 64

4.7 Diagrama de Componentes 65

**5. IMPLEMENTACIÓN DEL ESTILO ARQUITECTÓNICO EN EL PROTOTIPO 66**

5.1 Fase: Inicio 66

5.1.1 Iteración 1 - Caracterización previa de la aplicación objeto de auditoría 66

5.2 Fase: Elaboración 68

5.2.1 Iteración 2 - Implementación del aplicativo Factura Electrónica como prototipo 68

5.3 Fase: Implementación 71

5.3.1 Iteración 3 - Implementación del Web Service a través del IDE Netbeans 71

5.4 Fase: Transición 74

5.4.1 Iteración 4 - Integración del aplicativo y el WS a través del módulo de seguridad 74

**6. PROPUESTA TÉCNICA DE LA AUDITORÍA EN EL MODELO 78**

6.1 Escenario para la auditoría de servicios web con SQL Injection 78

6.2 Inyección de SQL en la aplicación y servicio web propuestos 78

**7. EJECUCIÓN DE LA AUDITORÍA EN EL MODELO 80**

7.1 Jerarquía del modelo base de conocimiento para auditorías de servicios web 80

7.1.1 ACERCA DE LA AUDITORÍA 80

7.1.2 ACTUAL SITUACIÓN Y METODOLOGÍA 82

7.1.3 AUDITORÍA WEB 83

7.1.4 FASE DE CONOCIMIENTO 84

7.1.5 FASE DE MAPEADO 85

7.1.6 FASE DE EXPLOTACIÓN 87

7.1.7 RESULTADOS Y CONCLUSIONES 87

7.2 Implementación del modelo en la herramienta de base de conocimiento OpenKM 88

7.3 Categorización del modelo 90

7.4 Metadata del modelo 91

7.5 Plantillas usando los elementos de evaluación del modelo en OpenKM 92

7.6 Búsquedas en OpenKM como herramienta base de conocimiento 93

7.7 Tablero en OpenKM como herramienta base de conocimiento 94

7.8 Administración en OpenKM como herramienta base de conocimiento 95

7.9 Escenario usuario en OpenKM como herramienta base de conocimiento 96

**8. CRONOGRAMA 97**

**9. CONCLUSIONES 98**

**BIBLIOGRAFÍA 99**

# 1. INTRODUCCIÓN

Desde el momento en que una aplicación web de tipo organizacional es desplegada en un ambiente de producción bajo una determinada infraestructura, es susceptible de recibir ataques malintencionados. Prevenir que esos ataques resulten exitosos, es uno de los principales intereses del área de seguridad de cualquier entidad gubernamental o privada.

El proyecto se enfoca en dos áreas complementarias: (a) Modelos de base de conocimiento del dominio de seguridad de aplicaciones web y (b) auditorías de vulnerabilidad a vectores de ataque SQL Injection en servicios web. Abordados desde los fundamentos, estándares, métodos y conceptos técnicos definidos en: la Estrategia de Gobierno en línea del MinTIC con normas legales y de referencia tales como la NTC-ISO-IEC 27000, NTC-ISO-IEC 27001, NTC-ISO-IEC 27002, NTC-ISO-IEC 27005, NTC-ISO-IEC 31000, las guías de Gestión de Riesgos y Auditorías, los dominios definidos por el consorcio ISC2, la metodología OSSTMM, y las buenas prácticas, herramientas y técnicas del proyecto OWASP.

Este proyecto en modalidad de profundización busca generar un modelo base de conocimiento para guiar procesos de auditoría de seguridad web en un prototipo organizacional, frente a los riesgos informáticos a los que esté expuesto, en concordancia con los recientes ataques cibernéticos a las plataformas de las grandes multinacionales; ofreciendo alternativas de solución y así contribuir en el campo de investigación de seguridad de MinTIC en el sector nacional, e internacionalmente a los ya propuestos por la ISC2 y OWASP en el área de la seguridad de la información y seguridad informática.

A su vez se busca a futuro, evaluar el estado de seguridad en los servicios web expuestos en los aplicativos en producción de una entidad, enmarcado en un modelo de información específico y mediante herramientas y técnicas del modelo, se permitirá la visualización, y definición de un estado de seguridad de su información digital y de los datos expuestos de dicha organización.

Una de las bases de este proyecto se fundamenta en el decreto del reglamento 1078 de 2015 de MinTIC, el cual establece en el **Artículo 2.2.17.6.6. *Seguridad de la información*** que "*Los actores que traten información, en el marco del presente título, deberán adoptar medidas apropiadas, efectivas y verificables de seguridad que le permitan demostrar el correcto cumplimiento de las buenas prácticas consignadas en el modelo de seguridad y privacidad de la información emitido por el Ministerio de Tecnologías de la Información y las Comunicaciones, o un sistema de gestión de seguridad de la información certificable. Esto con el fin de salvaguardar la confidencialidad, integridad y disponibilidad de los activos de información*"; debido a esto en las organizaciones se deben identificar los riesgos y desarrollar los controles necesarios para prevenirlos, identificarlos y mitigarlos, tales

como la penetración vía web por parte de personas, grupos e incluso otras organizaciones con el fin de obtener datos y/o alterarlos en los sistemas de información apoyados en sus aplicativos.

## 1.1 Definición del problema

Debido a la gran cantidad de ataques cibernéticos a nivel internacional y nacional se han venido disparando mecanismos preventivos y procedimientos en las organizaciones, esto con el fin de contrarrestar estas vulnerabilidades en seguridad de la información, sin embargo se debe tener en cuenta que nunca habrá una infraestructura 100% segura, dado que siempre existirán riesgos en las plataformas debido a la naturaleza en la que cambian las formas de realizar los ataques.

Ahora bien, en referencia a éste contexto, se realizó una revisión desde el 2011 a la fecha con una herramienta de la reconocida multinacional IBM, llamada IBM X-Force IRIS, la cual arrojó a SQL Injection como el vector de ataque (conocido) más consistente y extendido a través de estos años.

De acuerdo al Decreto 2183 de diciembre de 2017 del Departamento Administrativo de la Función Pública de Colombia, el cual dicta, entre otras medidas, "*modificar parcialmente la estructura de la DIAN (Dirección de impuestos y Aduanas Nacionales) dando origen a la nueva Oficina de Seguridad de la Información, en concordancia con el Modelo de Seguridad y Privacidad de la Información y la Estrategia de Gobierno Digital, de manera que se proteja la información, independientemente de si ésta se encuentra en medios físicos o electrónicos y en su proceso de acceso, uso, circulación, divulgación, interrupción o destrucción*", se creó la oficina de seguridad de la información para gestionar estos temas.

La problemática de éste proyecto surge de la necesidad de realizar una propuesta a la oficina de seguridad de la información de la DIAN para implementar y hacer seguimiento a la Estrategia de Gobierno en Línea de MinTIC en el componente de Seguridad y Privacidad de la Información, mediante la política de seguridad de la información institucional y a través de éste modelo de base de conocimiento para auditorías en servicios web.

## 1.2 Pregunta de investigación

Para evaluar las vulnerabilidades y valorar las amenazas cibernéticas en las entidades, con fines de alteración, eliminación o robo de información, especialmente en servicios web asumiendo el vector de ataque SQL Injection, se plantea la siguiente pregunta, ¿Cuáles son los elementos de un modelo base de conocimiento organizacional de auditoría en Servicios Web, para prevenir incidentes de seguridad de tipo SQL Injection y así poder asegurar la acción de una organización frente a la integridad de sus datos y confidencialidad de su

información?

Para responder a la pregunta, se tiene que indagar acerca de:

- ¿Qué servicios web se podrán evaluar para tener pruebas de penetración SQL Injection en la entidad, bajo las metodologías, herramientas y técnicas del MinTIC, el ISC2 y OWASP?

- ¿Qué mecanismo de organización podrá implementar jerárquicamente la información resultante de las pruebas de penetración con SQL Injection?

## 1.3 Alcances

Para aplicar el modelo propuesto en un ambiente organizacional en el marco del contexto colombiano, la información que se evalúe será únicamente la que permitan las organizaciones teniendo en cuenta la ley estatutaria No. 1266 de Diciembre del 2008, expedida por el Congreso de la República de Colombia, por la cual se dictan disposiciones generales del Habeas Data; ahora bien, si se quiere implementar en un contexto internacional dependerá de los lineamientos de cada país o determinada organización, para el caso en cuestión se extrajo la información pública del SIE (Sistema de información electrónico) de Factura electrónica como ejemplo.

Las pruebas de inyección SQL se hicieron en un aplicativo con una solución on-premise, como prototipo de un sistema de Factura Electrónica, tecnológicamente similar a la que se usa actualmente en la DIAN.

Para la implementación del modelo base de conocimiento se acudió a recursos monetarios, de experticia y de tiempo necesarios para el desarrollo del proyecto, para éste caso en específico se acudió a la Red de Investigaciones de Tecnología Avanzada de la Universidad Distrital (RITA), la cual me brindó el servidor para instalar la herramienta de base de conocimiento que soporta el modelo.

## 1.4 Objetivos

### 1.4.1 Objetivo General

Modelar una base de conocimiento para la auditoría de vulnerabilidad a vectores de ataque de inyección SQL en Servicios Web de una entidad, basados en tendencias, metodologías y buenas prácticas de seguridad de la información, gestionando los riesgos o incidentes de seguridad en las organizaciones.

### 1.4.2 Objetivos específicos

- Establecer una metodología propia para el modelo, con metodologías definidas, métodos y buenas prácticas en materia de modelos de base de conocimiento integrados con el área de auditorías de seguridad de la información para los Servicios Web de una entidad o un prototipo organizacional.

- Analizar y desarrollar guías, técnicas y herramientas de software de Inyección SQL, sobre uno o varios servicios web, en una entidad o en un prototipo organizacional, para la generación de datos de seguridad de la información.

- Operacionalizar un software de base de conocimiento con datos resultantes de las pruebas de Inyección SQL para tener una base de reporte histórico de auditoría de seguridad de información y así poder tener un control ante posibles ataques o riesgos en la información.

## 1.5 JUSTIFICACIÓN

Organizacionalmente, uno de los Objetivos Tácticos dentro del plan estratégico 2018-2022 es *"desarrollar y mantener la tecnología de la información y las telecomunicaciones para optimizar la gestión institucional y la estrategia de Gobierno en Línea, en función de un mejor servicio al ciudadano: Rediseño estructural de la arquitectura tecnológica; desarrollo de nuevos servicios informáticos, mantenimiento y ajuste de los actuales; y desarrollo de la infraestructura y la gestión de la seguridad informática",* en concordancia con el Decreto 2693 de 2012 del MinTIC el cual promueve la implementación de Sistemas de Gestión de Seguridad de Información.

En ese orden de ideas, una de las iniciativas es implementar y hacer seguimiento a la Estrategia de Gobierno en Línea en el componente de seguridad y privacidad de la Información, con base en esto, surgió la idea para realizar éste proyecto del cual se esperan los siguientes resultados técnicos:

- Auditoría de seguridad en Servicios Web de una entidad o un prototipo organizacional enfatizando en el tema de SQL Injection.

- Con el documento maestro resultado del punto anterior, alimentar la base de conocimiento orientada a la seguridad informática.

A nivel de ingeniería con este proyecto se gestionará los puntos ya descritos específicamente en la seguridad de los Servicios Web de la organización o del prototipo, mediante la prevención de ataques de penetración (SQL Injection) o inyección de código

fuente utilizando buenas prácticas, guías, técnicas y herramientas de software para realizar pruebas al aplicar la auditoría con metodologías del MinTIC avaladas por la ISC2.

En términos económicos y de seguridad de la información, con este proyecto se podrá hacer una propuesta oficial de implementación en determinada entidad la cual podrá contribuir como un referente en el contexto organizacional para garantizar la integridad de los datos y la confidencialidad de la información, generando un ahorro en investigaciones e implementaciones de modelos base de conocimiento para auditorías en servicios web.

# 2. ESTADO DEL ARTE

## 2.1 Marco teórico

El desarrollo de éste modelo base de conocimiento para auditorías de seguridad en servicios web con SQL injection se centró en dos áreas específicas: el área de bases de conocimiento y el área de seguridad de la información, así que se realizó la investigación de los principales estudios a nivel internacional, involucrando una revisión literaria con los siguientes temas clave: ‘'Knowledge base model AND information security', 'Knowledge base AND information security auditing' y 'Knowledge base AND sql injection AND web service'.

El dominio de aplicación de éste modelo para el presente caso se realizó a nivel nacional y se concretó con el prototipo de una aplicación en un ambiente organizacional teniendo en cuenta lineamientos del MinTIC, buenas prácticas de la ISC2, OSSTMM y OWASP.

### 2.1.1 Modelo de auditoría de conocimiento para la seguridad de la información

Los repetidos incidentes de seguridad de la información (InfoSec) muestran un estudio sobre qué tan bien las empresas han mantenido la confidencialidad, integridad y disponibilidad (CIA) de sus sistemas InfoSec, dónde la mayoría de las organizaciones adoptan un marco de control del sistema de información como ISO27001 y COBIT, que requieren auditorías periódicas por canales internos o externos. La mayoría de las auditorías combinan el uso de procedimientos cuantitativos y cualitativos basados en listas de verificación, centrándose en que tan bien el auditado ha cumplido los criterios de la auditoría, incluido el nivel de conocimiento de las partes interesadas. Sin embargo, tales auditorías tradicionales no adoptan la metodología contemporánea de gestión del conocimiento, como las sugeridas por Nonaka, y no exploran cómo el conocimiento puede contribuir a la madurez de InfoSec. (AU y FUNG. 2018, p. 1)

La auditoría de conocimiento (KA) ayuda a las empresas a identificar qué conocimiento e información se necesitan para lograr el objetivo comercial, sin embargo, actualmente existe poca investigación sobre la aplicación de KA en la auditoría de InfoSec. Las empresas importantes utilizan el gobierno de la tecnología de la información (ITG) como un vehículo organizacional para impulsar y aprovechar la entrega adecuada de los sistemas de información y las funciones de seguridad. Esta investigación tiene la intención de desarrollar un Marco de conocimiento impulsado por ITG (ITGKF), esperando que se refuerce la madurez de InfoSec y la auditabilidad de las empresas, además, trata de evaluar si ITG puede adoptar la circulación adecuada de conocimiento dentro de la comunidad InfoSec. Según el marco, se ofrecen algunas recomendaciones sobre cómo llevar a cabo

una auditoría de conocimiento para InfoSec (KA-InfoSec) como suplemento, para ello el marco y las recomendaciones deberían proporcionar a las empresas dimensiones y alternativas adicionales, basadas en la ideología centrada en el conocimiento, sobre cómo el conocimiento puede ser un factor crítico para el mantenimiento de InfoSec. (AU y FUNG. 2018, p. 2)

### 2.1.2 Modelo de gestión del conocimiento para mejorar la seguridad de la información

La seguridad de la información es un tema importante en el mundo de hoy, donde la gestión de esta ya no puede hacerse simplemente con un conjunto de hardware y software, sino que requiere un sistema completo de extremo a extremo, denominado Sistema de Gestión de Seguridad de la Información, en el presente trabajo se ha propuesto un modelo para mejorar la seguridad de la información utilizando técnicas de gestión del conocimiento. (Mittal, Roy y Saxena, 2010, p.1)

El modelo tiene tres módulos de nomenclatura: el módulo de depósito de conocimiento de seguridad de la información, el módulo de intercambio y difusión del conocimiento de seguridad de la información y el módulo de implementación y efectividad del conocimiento de seguridad de la información, el primer módulo se encarga de almacenar el conocimiento de seguridad de la información en un formato sistemático y fácil de usar, el segundo módulo promueve el intercambio y la difusión del conocimiento y el tercer módulo es responsable de monitorear y medir la efectividad del sistema total, respectivamente, donde se han asignado herramientas de gestión del conocimiento a cada módulo para lograr los objetivos de los mismo, posterior a ello se ha analizado la relación entre los módulos. (Mittal, Roy y Saxena, 2010, p.2)

### 2.1.3 Desarrollo de un sistema experto para la auditoría de seguridad de la información

En los días del avance tecnológico, el papel de la seguridad de la información (SI) es muy importante, donde existe una necesidad urgente de implementar y evaluar la seguridad de la información a un excelente nivel. Sin embargo, este proceso va acompañado de costos muy altos, ya que los expertos en esta área de Seguridad de la Información (SI) son especialistas bastante costosos. La automatización de algunas operaciones de seguridad y la evaluación de tareas pueden reducir estos costos y potencialmente aumentar la calidad del desarrollo de las estrategias de SI y la calidad de la auditoría de SI ya que la auditoría de la seguridad de la información constituye una parte clave de las operaciones de seguridad realizadas dentro de las organizaciones, teniendo en cuenta que los problemas de seguridad de la información deben abordarse lo más rápido posible. Para ello debemos tener presente que al hacer que el proceso de auditoría cumpla con los estándares internacionales en

seguridad de la información, p. ISO27000, estamos desarrollando sistemas expertos difusos en auditoría de seguridad de la información (Bennett, 2015, p. 19)

En las preguntas sobre el desarrollo de sistemas expertos, hay muchos problemas relacionados con la creación de una base de conocimiento, generación de recomendaciones, formación de reglas, etc. Aunque la seguridad de la información es un campo muy amplio, que abarca muchos conceptos complejos, estamos tratando de desarrollar un Metodología de formalización del conocimiento de SI para construir una base de conocimiento para un sistema experto que pueda servir como experto en auditoría de SI. Este sistema inteligente se acompaña de un módulo de aplicación basado en la web, que sirve como instrumento para resolver diversos problemas relacionados con la seguridad. La base metodológica y de investigación se basa en la teoría de conjuntos difusos y lógica que pueden simular bien el proceso del pensamiento humano y la toma de decisiones. Este enfoque y el sistema experto propuesto se probaron para definir el nivel de seguridad de algunas empresas. (Bennett, 2015, p. 20-65)

### 2.1.4 Creación de una base de conocimientos para un sistema experto en seguridad de la información

En los días del avance tecnológico, el papel de la seguridad de la información (SI) es muy importante, donde existe una necesidad urgente de implementar y evaluar la seguridad de la información a un buen nivel, sin embargo, se acompaña de costos muy altos: los expertos en SI son especialistas bastante caros. La automatización de algunas tareas de implementación y evaluación de seguridad puede reducir estos costos y potencialmente aumentar la calidad de las estrategias de SI que se están desarrollando y la calidad de la auditoría de SI, donde se cree que el enfoque de sistemas expertos puede ser beneficioso para lograr esta automatización. Aunque la seguridad de la información es un campo muy amplio que abarca muchos conceptos complejos, actualmente se trata de desarrollar una metodología de formalización del conocimiento de SI para construir una base de conocimiento para un sistema experto que pueda servir como experto en auditoría de SI, por lo que en este artículo se discuten métodos para la construcción de bases de conocimiento. (Atymtayeva, Kozhakhmet y Bortsova, 2014, p. 1)

### 2.1.5 Un marco de conocimiento para el modelado de seguridad de la información

El proceso de recopilación de datos para la evaluación de riesgos depende en gran medida de la experiencia en seguridad del personal de seguridad de una organización, donde es difícil contar con el personal de seguridad de la información adecuado, que comprenda tanto los requisitos de seguridad como el estado actual de seguridad de una organización y al mismo tiempo posea la habilidad para realizar una evaluación de riesgos. Sin embargo, un modelo de conocimiento bien definido podría ayudar a describir las categorías de

conocimiento requeridas para guiar el proceso de recopilación de datos. En este documento, se presenta un marco de conocimiento, que incluye un modelo de conocimiento para definir el esqueleto de datos del entorno de riesgo de una organización y los patrones de seguridad sobre las relaciones entre amenaza, entidad y contramedidas; y un mecanismo de integración de datos para integrar datos distribuidos relacionados con la seguridad en un repositorio de datos de seguridad que es específico de una organización para el modelado de seguridad de la información. (Shuangyan, Ching-hang y Lam-for, 2006, p. 1)

### 2.1.6 Un método de construcción de base de conocimiento semántico para la seguridad de la información

La seguridad de la información contiene muchos conceptos y entidades de conocimiento, a medida que se desarrolla la tecnología de la información, la complejidad de aumentar el conocimiento de seguridad de la información necesita una representación general y una organización para el análisis de seguridad y la evaluación de riesgos. La ontología como modelo semántico formal y compartible, que a menudo se usa para definir el esquema de conocimiento de dominio, también se puede aplicar para la construcción de la base de conocimiento de seguridad de la información. En este documento, proponemos un método de construcción de base de conocimiento de ontología para la seguridad de la información, discutimos los procesos de construcción de ontología y diseñamos el esquema de conocimiento. La ontología contiene conceptos principales en seguridad de la información, propiedades y relaciones, conectados entre sí con los conceptos propios de la semántica. Proporciona información relacionada, como activos y debilidades, a las aplicaciones de gestión y análisis de seguridad, en el presente proyecto se introdujo cada paso del método propuesto y se validó utilizando un desarrollo práctico de la base de conocimiento de seguridad de la información. (YAO, LIU, YI, ZHAO y LIU, 2014, p .1)

### 2.1.7 Un modelo de intercambio de conocimientos sobre seguridad de la información en las organizaciones

El intercambio de conocimientos juega un papel importante en el dominio de la seguridad de la información, debido a su efecto positivo en la conciencia de seguridad de la información de los empleados. Se reconoce que la conciencia de seguridad es el factor más importante que mitiga el riesgo de violaciones de seguridad de la información en las organizaciones. En esta investigación, se ha presentado un modelo que muestra cómo se forma el intercambio de conocimientos de seguridad de la información (ISKS) y disminuye el riesgo de incidentes de seguridad de la información. La teoría de la motivación y la teoría del comportamiento planificado además del modelo de Triandis se aplicaron como la columna vertebral teórica del marco conceptual. (Nader y Rossouw, 2016, p. 1)

Los resultados del análisis de datos mostraron que ganarse una reputación y ganarse la promoción como motivación extrínseca y la satisfacción de la curiosidad como motivación intrínseca tienen efectos positivos en la actitud de los empleados hacia ISKS. Sin embargo, la satisfacción de la autoestima no influye en la actitud de ISKS. Además, los hallazgos revelaron que la actitud, el control conductual percibido y las normas subjetivas tienen efectos positivos sobre la intención de ISKS y la intención de ISKS afecta el comportamiento de ISKS. Los resultados también mostraron que el apoyo organizacional influye en el comportamiento de ISKS más que en la confianza, por ende los resultados de esta investigación deberían ser de interés para académicos y profesionales en el ámbito de seguridad de la información. (Nader y Rossouw, 2016, p. 2)

### 2.1.8 Seguridad de la información centrada en el conocimiento

Un número creciente de empresas considera la seguridad de la información (InfoSec) como un factor clave de éxito, mientras que las amenazas pueden originarse en cualquier parte del mundo a través de Internet. Incidentes como el 11 de septiembre de 2001, y las historias de terror de los piratas informáticos, despiertan a las empresas que deberían reformar sus políticas de InfoSec. Tradicionalmente, InfoSec depende en gran medida de la tecnología, invirtiendo una gran cantidad de dinero en software y hardware. En la última década ha surgido un sistema de gestión de seguridad de la información basado en procesos (SGSI) como ISO27001. Muchas organizaciones desde entonces han adoptado dichos SGSI. KM es otra disciplina de gestión que emplean las empresas, con el objetivo de fomentar una gestión más eficaz de la creación de conocimiento para las innovaciones. Este documento propone una iniciativa de investigación para integrar KM e InfoSec en un sistema InfoSec (KCIS) centrado en el conocimiento. Las organizaciones deberían poder mejorar su nivel de madurez de InfoSec, mediante la adopción de KCIS en fases (Fung y Fung, 2008, p .1).

### 2.1.9 Taxonomía de las técnicas de detección y prevención de inyecciones SQL

Si bien el uso de Internet para proponer servicios en línea aumenta cada día, las amenazas a la seguridad en la web también aumentaron dramáticamente, una de las vulnerabilidades de aplicaciones web más graves y peligrosas es la inyección SQL. El ataque de inyección SQL tuvo lugar al insertar una parte de la consulta SQL maliciosa a través de una entrada no validada del usuario en la declaración de consulta legítima. En consecuencia, el sistema de gestión de la base de datos ejecutará estos comandos y conduce a la inyección de SQL. Un ataque exitoso de inyección SQL interfiere en la confidencialidad, integridad y disponibilidad de información en la base de datos. Según las investigaciones estadísticas, este tipo de ataque tuvo un alto impacto en los negocios, por lo cual es necesario encontrar la solución adecuada para detener o mitigar la inyección de SQL. Para abordar este problema, los investigadores de seguridad introducen diferentes técnicas para desarrollar códigos seguros, prevenir ataques de inyección SQL y detectarlos. En este artículo se

presenta una revisión exhaustiva de diferentes tipos de técnicas de detección y prevención de inyección SQL, criticando las fortalezas y debilidades de cada técnica. Dicha clasificación estructural ayudaría aún más a otros investigadores a elegir la técnica correcta para los estudios posteriores. (Amirmohammad, Mazdak y Azizah, 2013, p. 1)

### 2.1.10 Taxonomía de los ataques de inyección SQL

Hoy en día, las aplicaciones web juegan un papel importante en los negocios en línea, incluidas las redes sociales, los servicios en línea, la banca, las compras, las clases, el correo electrónico, etc. La facilidad de uso y el acceso a la aplicación web los hacen más populares al ofrecer servicios en línea en lugar de servicios en persona. Un usuario simple solo necesita una computadora y una conexión a Internet para acceder a la aplicación web y usar los servicios en línea provistos por esa aplicación. Hay un núcleo en común entre todas las aplicaciones web dinámicas y es su necesidad de usar una base de datos para almacenar información dentro de ella y recuperar esa información a solicitud del usuario o agregarla, editarla y eliminarla. Entre todos los tipos de bases de datos, las bases de datos relacionales son muy populares, donde la mayoría de los sistemas de gestión de bases de datos relacionales como MySQL, Oracle, MS SQL Server, MS Access, PostgresSQL utilizan SQL como su lenguaje. (Amirmohammad, Mazdak y Azizah, 2013, p. 1)

La flexibilidad de SQL lo convierte en un lenguaje poderoso, el cual permite al usuario preguntar qué información quiere sin tener ningún conocimiento sobre cómo se obtendrá la información. Sin embargo, el gran uso de bases de datos basadas en SQL lo convierten en el centro de atención de los hackers. El ataque de inyección SQL es una amenaza de seguridad bien conocida para las aplicaciones web basadas en bases de datos. Un ataque exitoso de inyección SQL revela información confidencial crítica al hacker. En este documento, primero proporcionamos información básica sobre esta vulnerabilidad. A continuación, presentamos una revisión exhaustiva de los diferentes tipos de ataques de inyección SQL. Para cada ataque proporcionamos un ejemplo que muestra cómo se inicia el ataque. Finalmente, proponemos la mejor solución en la fase de desarrollo para vencer la inyección y la conclusión de SQL. (Amirmohammad, Mazdak y Azizah, 2013, p. 2)

### 2.1.11 La inyección SQL sigue viva: un estudio sobre técnicas de evasión de firma

La inyección SQL es uno de los mayores desafíos para la seguridad de las aplicaciones web. Según los estudios de OWASP, la inyección SQL tiene el rango más alto en las vulnerabilidades basadas en la web. En caso de un ataque de inyección SQL exitoso, el atacante puede tener acceso a la base de datos de la aplicación web. Con el rápido aumento de los ataques basados en inyección SQL, los investigadores comienzan a proporcionar diferentes soluciones de seguridad para proteger la aplicación web contra ellos. Una de las soluciones más comunes es el uso de firewalls de aplicaciones web. Por lo general, estos

firewalls utilizan una técnica basada en firmas como núcleo principal para la detección. En esta técnica, el firewall verifica cada paquete con una lista de ataques de inyección SQL predefinidos conocidos como firmas. El problema con esta técnica es que, un atacante con un buen conocimiento del lenguaje SQL puede cambiar el aspecto de las consultas SQL de una manera que el firewall no pueda detectarlas, pero aun así conducir a los mismos resultados maliciosos. En este artículo, primero describimos la naturaleza del ataque de inyección SQL, luego analizamos las técnicas actuales de evasión de detección de inyección SQL y cómo pueden evitar los filtros de detección, luego propusimos una combinación de soluciones que ayudan a mitigar el riesgo de ataque de inyección SQL. (Amirmohammad, Mazdak y Azizah, 2013, p. 1)

### 2.1.12 Auditoría de aplicaciones web: metodología y práctica profesional

La finalidad del presente trabajo es consolidar los conocimientos adquiridos durante la carrera, y más en concreto, en el área de seguridad y administración de redes y sistemas operativos. Esta memoria es el documento que recoge todo el trabajo realizado durante el proyecto, en el que se refleja tanto el estudio teórico del tema tratado, como la realización práctica del ejercicio de auditoría propuesto. (Rodríguez, 2015, p. 10)

En el proyecto, se plantea la problemática de una pequeña empresa que busca mejorar el desarrollo de su software, para diseñar páginas webs más seguras. La inseguridad de las páginas webs es un problema actual del mundo real. Poner solución a esto requiere de una labor técnica, que se condensa en una auditoría de la aplicación. El proceso de auditoría web que se describe aquí, nos servirá para detectar y resolver las vulnerabilidades del código, o de la configuración del sistema que aloja la aplicación. (Rodríguez, 2015, p. 11)

### 2.1.13 Propuesta de auditoría a las aplicaciones web de la empresa C&M consultores

En éste proyecto se abordó la propuesta de auditoría a las aplicaciones web desde el enfoque de dos áreas específicas. En primer lugar, la auditoría de aplicaciones consiste en tratar de ayudar, planificar, preparar y realizar auditorías de aplicaciones web en cuanto al grado de cumplimiento de los objetivos para los que las mismas fueron creadas. De esta manera y, en consecuencia, se apoyó el logro de los objetivos organizacionales de la manera más satisfactoria. (Contreras, 2017, p. 11)

En segundo lugar, la auditoría de explotación debe asegurar el funcionamiento adecuado de sistemas informáticos y su actualización. La detección oportuna de las debilidades del sistema permite mejorarlos racionalizando los recursos. (Contreras, 2017, p. 12)

### 2.1.14 Seguridad informática en web services

Los web services marcan un nuevo norte sobre el desarrollo de aplicaciones web, son una nueva tecnología que todavía no ha conocido todo su potencial. Los desarrolladores sin embargo reconocen que las empresas se atemorizan al implementarlos porque pueden representar un gran riesgo de seguridad; es así, que las empresas líderes como Microsoft e IBM proponen un framework llamado WSE (Web Services Enhancements) que garantiza cierto nivel de seguridad para aplicaciones de Web Services. El trabajo de investigación realizado, propone y comprueba un Modelo de Seguridad orientado a Web services que busca complementar el trabajo ya existente. (Farías e Infante, 2005, p. 1)

### 2.1.15 Pruebas de penetración con Kali linux: inyección SQL, XSS y Wordpress

Hoy en día, las computadoras, teléfonos inteligentes, relojes inteligentes, impresoras, proyectores, lavadoras, refrigeradores y otros dispositivos móviles conectados a Internet están expuestos a diversas amenazas y hazañas. De los diversos ataques, la inyección SQL, las secuencias de comandos de sitios cruzados, Wordpress y el ataque WPA2 fueron los ataques de seguridad más populares y se investigarán más a fondo en este documento. Kali Linux proporciona una gran plataforma y un medio para aprender varios tipos de exploits y pruebas de penetración. Todo el ataque simulado se llevará a cabo utilizando Kali Linux instalado en una máquina virtual en un equipo con Intel Core i5 y 8 GB de RAM, mientras que la máquina de la víctima es la computadora host que ejecuta Windows 10 versión 1709. Los resultados mostraron que los ataques se lanzaron tanto en la web y firewall se llevaron a cabo con éxito. (Gunawan, Kartiwi, Malik e Ismail, 2018, p.1)

### 2.1.16 Evaluación de la seguridad de la aplicación web mediante la gestión de errores.

Como una plataforma de aplicaciones grande y compleja, la World Wide Web es capaz de ofrecer una amplia gama de sofisticadas aplicaciones. Sin embargo, muchas aplicaciones Web pasan por fases de desarrollo ágil con un tiempo de respuesta extremadamente corto, lo que dificulta la eliminación de vulnerabilidades. Acá se analiza el diseño de mecanismos de evaluación de la seguridad de las aplicaciones Web con el fin de identificar prácticas de codificación deficientes que hacen que las aplicaciones Web sean vulnerables a ataques como la inyección de SQL (SQL Injection) y la codificación de sitios cruzados (Cross-Site Scripting - XSS). Se describe el uso de varias técnicas de prueba de software (incluyendo análisis dinámico, pruebas de caja negra, inyección de fallas y monitoreo de comportamiento), y se sugiere mecanismos para aplicar estas técnicas a aplicaciones Web. Las situaciones del mundo real se utilizan para probar una herramienta que se denomina Vulnerabilidad de Aplicaciones Web y Escáner de Errores (WAVES, un proyecto de código abierto disponible en http://waves.sourceforge.net) y compararlo con otras

herramientas. Los resultados muestran que WAVES es una plataforma factible para evaluar la seguridad de las aplicaciones web. (Huang, Huang, Lin y Tsai, 2003, p. 1)

### 2.1.17 Uso del análisis estático para encontrar errores.

El análisis estático examina el código en ausencia de datos de entrada y sin ejecutar el código. Puede detectar posibles violaciones de seguridad (SQL Injection), errores de ejecución (extracción de un puntero nulo) e inconsistencias lógicas (una prueba condicional que posiblemente no sea verdadera). Aunque existe una amplia literatura sobre algoritmos y frameworks analíticos usados por tales herramientas, los informes que describen experiencias en la industria son mucho más difíciles de conseguir. Los autores describen FindBugs, como una herramienta de análisis estático de código abierto para Java, y las experiencias de uso en la producción. FindBugs evalúa qué tipos de defectos pueden ser detectados eficazmente con técnicas relativamente simples y ayuda a los desarrolladores a comprender cómo incorporar tales herramientas en el desarrollo de software. (Ayewah, Hovemeyer, Morgenthaler, Penix y Pugh. 2008, p.1)

### 2.1.18 Sania: Análisis sintáctico y semántico para pruebas automatizadas.

Con el reciente incremento rápido de aplicaciones Web interactivas que emplean servicios de bases de datos back-end, los ataques de SQL Injection se han convertido en una de las amenazas de seguridad más graves. El ataque SQL Injection permite a un atacante acceder a la base de datos subyacente, ejecutar comandos arbitrarios intencionados y recibir una salida generada dinámicamente, como páginas Web HTML. En este artículo, presentamos nuestra técnica, Sania, para detectar vulnerabilidades de SQL Injection en aplicaciones Web durante las fases de desarrollo y depuración. Sania intercepta las consultas SQL entre una aplicación web y una base de datos y genera automáticamente ataques elaborados de acuerdo con la sintaxis y la semántica de los puntos potencialmente vulnerables en las consultas SQL. Además, Sania compara los árboles de análisis de la consulta SQL deseada y los que reflejan después de un ataque para evaluar la seguridad de estos puntos. Evaluamos nuestra técnica utilizando aplicaciones web del mundo real y encontramos que nuestra solución es eficiente en comparación con un popular escáner de vulnerabilidades de aplicaciones web. También encontramos vulnerabilidad en un producto que estaba a punto de ser lanzado. (Kosuga, Kono, Hanaoka, Hishiyama y Takahama, 2007, p. 1)

### 2.1.19 Implementación de técnicas y recomendaciones de seguridad OWASP.

En la Universidad Técnica Particular de Loja de Ecuador se realizó un trabajo que contiene la implementación de técnicas y recomendaciones OWASP sobre un prototipo SOA desarrollado con J2EE. Para su diseño y codificación utilizamos algunas especificaciones

WS-Security, framework Metro, MVC como patrón arquitectónico, Fachada y DAO como patrones de diseño. El prototipo fue validado en términos de diseño, codificación y seguridad a través de algunas herramientas como: Análisis Estructural para Java, SonarQube, OWASP ZAP, Vega y Wireshark. Con el desarrollo de este prototipo probamos que el uso de estándares, recomendaciones y técnicas para escribir código seguro en aplicaciones de software son necesarios para prevenir vulnerabilidades; además, el análisis estático ayuda a identificar brechas de seguridad y aspectos de calidad que muchas veces no son considerados por los desarrolladores. (Guaman, Jaramillo y Sucunuta, 2017, p. 1)

### 2.1.20 Asegurar la aplicación web de los ataques de inyección de SQL

La inyección SQL es un proceso de ataque generalizado para aplicaciones web que apunta al objetivo a través del firewall que lo protege. La escasa validación de entrada en el código y la administración del sitio web hace que la inyección de SQL ataque al recurso web. Dicho sistema propuesto presenta un método de autenticación para detectar y prevenir la inyección de SQL del tipo de tautología de usuarios web malintencionados que desean acceder a servicios web en aplicaciones relacionadas con la web. Para detectar y evitar que usuarios malintencionados accedan a servicios web, dicho sistema proporciona una validación basada en XML efectiva. La investigación también concluye la efectividad y el rendimiento del sistema utilizando los datos resultantes del sistema propuesto. (Krishnaveni, 2016, p. 1-3)

### 2.1.21 OwlEye: un sistema avanzado de detección de ataques web basado en HMM

Los servicios web de hoy en día desempeñan un papel importante en nuestra vida diaria, la seguridad de los servidores y aplicaciones web se han convertido en uno de los temas clave de la seguridad cibernética. Las cadenas de consulta de solicitud web (consultas) son responsables de pasar los parámetros a los recursos a los que se hace referencia, por lo que son vulnerables a que los atacantes los manipulen para recuperar datos confidenciales e incluso perder el control total de las víctimas. En dicho documento, proponen OwlEye, un sensor de detección de ataque híbrido basado en el modelo oculto de Markov (HMM), diseñado para ataques de inyección de código de texto, como la inyección de SQL y las secuencias de comandos cruzadas. Su innovador diseño de arquitectura de puntuación de directorios dobles ofrece la ventaja de utilizar tanto tráfico benigno como malintencionado en la capacitación de modelos, por lo que ha logrado una tasa de detección satisfactoria a una tasa aceptable de falsos positivos (Liu, 2018, p. 1-6).

## 2.2 Marco conceptual

### 2.2.1 Modelo y modelo de información

Existen varias definiciones y bastante ambiguas de modelo, sin embargo vamos a remitirnos a dos investigaciones de revistas indexadas, la primera nos dice que un modelo es una representación ejemplar o general de una lengua (Moreno, 2001, p. 5); la otra dice que un modelo es un instrumento o esquema conceptual por el cual los seres humanos intentan articular de manera sistemática el conocimiento que se obtiene de la experiencia mediante el proceso de investigación; se entiende como una entidad intermedia entre la teoría y la realidad. (Carvajal, 2002, p. 2)

Ahora bien, extendiendo su definición como modelo de información podemos expresarlo como la concretización de un enfoque abstracto construido a partir de la representación de elementos relacionados y definidos por unos interesados. (Carvajal, 2002, p. 29)

### 2.2.2 Seguridad de software

La seguridad contempla un alto grado de criticidad; pues, desde el punto de vista de software, los datos son un bien clave a nivel organizacional que deben ser asegurados desde todos los aspectos (Serme, De Oliveira, Massiera y Roudier, 2012, p. 114-121; McGraw, 2004, p. 80-83). Además, se consideran como un conjunto de recursos destinados a lograr que los activos de una organización sean confidenciales, íntegros, consistentes y disponibles a sus usuarios, autentificados por mecanismos de control de acceso y sujetos a auditoría.

Existen tres aspectos a tomar en cuenta para el aseguramiento de la información:

1. Confidencialidad: Asegura que la información no pueda estar disponible, ni descubierta por terceros.
2. Disponibilidad: Garantiza la seguridad de la información, hardware y software, para ser accedido en cualquier momento.
3. Integridad: Condición de seguridad que certifica que la información ha sido insertada, actualizada y eliminada por usuarios autorizados.

Existen implicaciones considerables dentro de una organización como producto de la falta de estandarización y uso de normativas de seguridad durante un proceso de desarrollo de software. Específicamente el aseguramiento en la actualidad no se planea, sino se ejecuta al ultimar detalles de implementación de software. (Yag, 2015, p. 1)

### 2.2.3 WS-Security

En los servicios web (WS) la autenticación y el nivel de seguridad del mensaje (message-level security) se pueden lograr a través del uso de WS-Security. La especificación WS-Security (Nadalin, Kaler, Hallam-Baker, Monzillo, 2004, p. 56) define un conjunto de extensiones para servicios SOAP con el fin de proporcionar integridad, confidencialidad y disponibilidad de la información, mediante el cifrado, firmado y soportando varios formatos de tokens de seguridad. WS-Security es flexible y está diseñado para ser utilizado como base en la construcción de modelos de seguridad entre los que se incluye PKI, Kerberos, y SSL (Moser, Melliar-Smith, Zhao, 2007, p. 14-26), con lo cual provee soporte para múltiples tokens de seguridad, dominios de confianza, formatos de firmado, y tecnologías de encriptación (Makino, Tamura, Imamura, Nakamura, 2004, p. 58–72). El objetivo de WSSecurity es la seguridad a nivel de intercambio de mensajes SOAP messages y está sometido a la estandarización dada por OASIS Web Services Security Technical Committee. La especificación WS-Security define nuevas extensiones SOAP (message headers) para proporcionar autenticación mensaje a mensaje y prevenir la repetición de los mismos a través de marcas de tiempo (Naedele, 2003, p. 96-98), así como confidencialidad e integridad de extremo a extremo implementando XML Encryption y XMLDSig respectivamente en un ambiente de Web Services.

### 2.2.4 Amenazas

Causa potencial de un incidente no deseado, que puede provocar daños a un sistema o a la organización. (ISO/IEC 27000, 2018, p. 10)

### 2.2.5 Análisis de Riesgo

Proceso para comprender la naturaleza del riesgo y determinar el nivel de riesgo. (ISO/IEC 27000, 2018, p. 8)

### 2.2.6 Auditoría

Proceso sistemático, independiente y documentado para obtener evidencias de auditoría y obviamente para determinar el grado en el que se cumplen los criterios de auditoría. (ISO/IEC 27000, 2018, p. 1)

### 2.2.7 Ciberseguridad

Capacidad del Estado para minimizar el nivel de riesgo al que están expuestos los ciudadanos, ante amenazas o incidentes de naturaleza cibernética. (Conpes, 2011, p. 2)

### 2.2.8 Gestión de incidentes de seguridad de la información

Procesos para detectar, reportar, evaluar, responder, tratar y aprender de los incidentes de seguridad de la información. (ISO/IEC 27000, 2018, p. 4)

### 2.2.9 Plan de tratamiento de riesgos

Documento que define las acciones para gestionar los riesgos de seguridad de la información inaceptables e implantar los controles necesarios para proteger la misma. (ISO/IEC 27000, 2018, p. 10)

### 2.2.10 Privacidad

En el contexto de este documento, por privacidad se entiende el derecho que tienen todos los titulares de la información en relación con la información que involucre datos personales y la información clasificada que estos hayan entregado o esté en poder de la entidad en el marco de las funciones que a ella le compete realizar y que generan en las entidades destinatarias del Manual del GEL la correlativa obligación de proteger dicha información en observancia del marco legal vigente. (MinTIC, 2018, p.15)

### 2.2.11 Riesgo

Posibilidad de que una amenaza concreta pueda explotar una vulnerabilidad para causar una pérdida o daño en un activo de información. Suele considerarse como una combinación de la probabilidad de un evento y sus consecuencias. (ISO/IEC 27000, 2018, p.8)

### 2.2.12 Seguridad de la información

Preservación de la confidencialidad, integridad y disponibilidad de la información, además, otras propiedades tales como autenticidad, responsabilidad, no-repudio y confiabilidad pueden estar involucradas. (ISO/IEC 27000, 2018, p.4)

### 2.2.13 Sistema de Gestión de Seguridad de la Información SGSI

Conjunto de elementos interrelacionados o interactuantes (estructura organizativa, políticas, planificación de actividades, responsabilidades, procesos, procedimientos y recursos) que utiliza una organización para establecer una política y unos objetivos de seguridad de la

información y alcanzar dichos objetivos, basándose en un enfoque de gestión y de mejora continua. (ISO/IEC 27000, 2018, p.11).

### 2.2.14 Vulnerabilidad

Debilidad de un activo o control que puede ser explotada por una o más amenazas. (ISO/IEC 27000, 2018, p.11)

### 2.2.15 SQL

Es un lenguaje declarativo de dominio específico, que se emplea para manipular y definir la estructura de los datos, especificar los permisos de acceso y describir las transacciones de un sistema de gestión de bases de datos. (ISO/IEC 9075, 2003, p.5 y 6)

Structured Query Language, lenguaje declarativo para acceso en Bases de Datos Relacionales. Su característica principal es uso de cálculo y álgebra relacional para realizar consultas y así obtener información necesaria. (Martín, 2002, p. 5)

### 2.2.16 SQL Injection

El ataque de inyección de SQL permite que un atacante acceda a la base de datos subyacente, ejecute comandos arbitrarios en el intento y reciba un resultado generado dinámicamente, como páginas web HTML. (Kosuga, Kono, Hanaoka, Hishiyama y Takahama, 2007, p. 1)

### 2.2.17 Base de Conocimiento

La base de conocimiento contiene los hechos y la heurística (métodos para aumentar el conocimiento), para inferir soluciones a problemas, hacer análisis, formar hipótesis, etc. (Feigenbaum, 1980, p. 14). El concepto de base de conocimiento proviene de la integración de la tecnología de bases de datos con inteligencia artificial. El conocimiento en la base de conocimiento necesita un modo de estructura unificada. La estructura lógica del conocimiento de tres clases, a saber, "Hecho-Concepto-Regla", se adopta en el sistema de base de conocimiento. (Chenjian, 2011, p. 2)

Una base de conocimiento, se basa en la inteligencia que tiene un cierto grado de comportamiento que nadie lo puede definir de forma exitosa, hace difícil atribuir un concepto claro de lo que es la inteligencia, sin embargo, se han desarrollado diversos modos de representar los conocimientos humanos y la estructura de la información que

asociamos con la inteligencia. Actualmente encontramos herramientas tecnológicas que incorporan dichos representaciones cognitivas y técnicas estructurales denominados sistemas expertos. (RAUCH-HINDIN, 1989, p. 1)

Existen dos tipos de bases de conocimiento:

- Bases de conocimiento legibles por máquinas: Son diseñadas para almacenar conocimientos en forma legible para el computador, con el fin de obtener razonamiento deductivo. Este tipo de bases de conocimiento son utilizadas por la Web semántica. (RAUCH-HINDIN, 1989, p. 2)
- Base de conocimientos legibles por humanos: Permiten acceder al conocimiento con los propósitos de aprendizaje, para obtener y manejar conocimiento explícito de las organizaciones. Proporcionando medios que permiten descubrir soluciones a problemas ya resueltos, los cuales podrían ser aplicados como base a otros problemas dentro o fuera de la misma área de conocimiento. (RAUCH-HINDIN, 1989, p. 1)

Una base de conocimiento tiene como objetivo principal modelar y almacenar bajo forma digital un conjunto de conocimiento, ideas, conceptos o datos que permitan ser consultados o utilizados en la base de conocimientos. Existen varios métodos y programas para crear bases de conocimientos. (Varela, Estrada y Acosta, 2017, p. 7)

### 2.2.18 OWASP

Es un proyecto de seguridad de aplicaciones web abiertas (OWASP) y una organización benéfica sin fines de lucro en todo el mundo centrada en mejorar la seguridad del software. Su misión es hacer visible la seguridad del software , para que las personas y las organizaciones puedan tomar decisiones informadas. OWASP se encuentra en una posición única para proporcionar información imparcial y práctica sobre AppSec a individuos, corporaciones, universidades, agencias gubernamentales y otras organizaciones en todo el mundo. Operando como una comunidad de profesionales con ideas afines, OWASP emite herramientas de software y documentación basada en el conocimiento sobre la seguridad de las aplicaciones. (OWASP Foundation, 2004, p. 1)

### 2.2.19 Falla de inyección SQL

Es una vulnerabilidad en la que los atacantes intentan explotar la inyección de SQL principalmente para obtener ganancias financieras (por ejemplo, el robo de información de identificación personal que luego se usa para el robo de identidad). (Universidad de Berkeley, 2019, p. 1) Las fallas de inyección permiten a un atacante crear, modificar o borrar cualquier información arbitraria disponible en la aplicación. En el peor de los

escenarios, estas fallas permiten a un atacante comprometer totalmente la aplicación y los sistemas subyacentes, incluso sobrepasar entornos con cortafuegos. (Owasp Foundation, 2008, p. 1) Las fallas de inyección SQL se presentan en tecnologías y/o aplicaciones producidas por desarrolladores sin experiencia que codifican tecnologías y/o plataformas emergentes. En caso de que se descubra una falla de inyección SQL, dicha configuración permite una forma fácil y conveniente de extraer información del servidor de la base de datos: si el atacante tiene suficientes privilegios para escribir en el sistema de archivos, puede redirigir los resultados de una consulta a un archivo dentro de la raíz del servidor web, y luego normalmente acceda al archivo con el navegador. (Clarke, 2012, p. 30 y 301)

### 2.2.20 Vulnerabilidad inyección SQL

Es una ocurrencia del desarrollador de una aplicación web, en la cual no se garantiza que los valores recibidos de un formulario web, cookie, parámetro de entrada, etc. se validen antes de pasarlos a las consultas SQL que se ejecutarán en un servidor de base de datos. Si un atacante puede controlar la entrada que se envía a una consulta SQL y manipular esa entrada para que los datos se interpreten como un código en lugar de como datos, el atacante puede ejecutar el código en la base de datos de back-end. Las vulnerabilidades de inyección de SQL ocurren por dos razones: (Clarke, 2012, p. 30)

- Falta de saneamiento en la entrada del usuario
- Estructuras de datos y control mezcladas en el mismo canal de transporte.

Estos problemas juntos han sido la causa de algunos de los tipos más importantes de vulnerabilidades explotadas hasta ahora en la historia de las computadoras, como los desbordamientos de memoria y de pila, y los problemas de cadenas de formato. La falta de saneamiento de entrada del usuario permite que un atacante salte de la parte de datos (por ejemplo, una cadena encerrada entre comillas simples o un número) para inyectar comandos de control. (como SELECT, UNION, AND, OR, etc.) Para defenderse contra este tipo de vulnerabilidad, la primera medida que se debe adoptar es realizar un saneamiento estricto de la entrada del usuario y / o una codificación de salida. Por ejemplo, puede adoptar un enfoque de lista blanca, por lo que si espera un número como valor de parámetro, puede configurar su aplicación web para rechazar todos los caracteres de la entrada suministrada por el usuario que no es un dígito. Si está esperando una cadena, solo acepta los caracteres que determinó anteriormente que no son peligrosos. Cuando esto no sea posible, debe asegurarse de que todas las entradas estén correctamente citadas/codificadas antes de usarse para evitar la inyección SQL. (Clarke, 2012, p. 59-60)

### 2.2.21 LDAP

Protocolo ligero de acceso a directorios. El Directorio es "una colección de sistemas abiertos que cooperan para proporcionar servicios de directorio". Un usuario del directorio, que puede ser una entidad humana u otra, accede al Directorio a través de un cliente (o Agente de Usuario del Directorio (DUA)). El cliente, en nombre del usuario del directorio, interactúa con uno o más servidores (o Agentes del sistema de directorio (DSA)). Los clientes interactúan con los servidores utilizando un protocolo de acceso a directorios. (RFC 4511, 2006, p. 2)

### 2.2.22 Inyección LDAP

Es un ataque utilizado para explotar aplicaciones basadas en web que construyen sentencias LDAP basadas en la entrada del usuario. Cuando una aplicación no puede desinfectar correctamente la entrada del usuario, es posible modificar las declaraciones LDAP usando un proxy local. Esto podría resultar en la ejecución de comandos arbitrarios, como la concesión de permisos a consultas no autorizadas y la modificación de contenido dentro del árbol LDAP. Las mismas técnicas de explotación avanzadas disponibles en la inyección SQL se pueden aplicar de manera similar en la inyección LDAP. (OWASP, 2016, p.1) Por ejemplo el directorio de Oracle para LDAP es Oracle Internet Directory (OID), el cual viene con muchas contraseñas con hash en varias tablas. Se puede acceder a los hashes de contraseña de OID si tiene acceso normal a todos los usuarios de la empresa. Por razones de compatibilidad, OID guarda la misma contraseña de usuario con diferentes algoritmos de hash (MD4, MD5 y SHA1). (Clarke, 2012, p. 293)

### 2.2.23 Inyección XPath

Al igual que en la inyección SQL, los ataques de inyección XPath ocurren cuando un sitio web utiliza información proporcionada por el usuario para construir una consulta XPath para datos XML. Al enviar información mal formada intencionalmente al sitio web, un atacante puede averiguar cómo están estructurados los datos XML o acceder a datos a los que normalmente no tiene acceso. Incluso puede ser capaz de elevar sus privilegios en el sitio web si los datos XML se utilizan para la autenticación (como un archivo de usuario basado en XML). La consulta de XML se realiza con XPath, un tipo de declaración descriptiva simple que permite que la consulta XML ubique una parte de la información. Al igual que SQL, puede especificar ciertos atributos para encontrar y patrones para hacer coincidir. Cuando se utiliza XML para un sitio web, es común aceptar algún tipo de entrada en la cadena de consulta para identificar el contenido para ubicar y mostrar en la página. Esta entrada debe limpiarse para verificar que no estropea la consulta XPath y devuelve los datos incorrectos. XPath es un lenguaje estándar; su notación / sintaxis es siempre independiente de la implementación, lo que significa que el ataque puede ser automatizado.

No hay dialectos diferentes, ya que tiene lugar en las solicitudes a las bases de datos SQL. (OWASP Foundation, 2015, p. 1)

### 2.2.24 Analizadores XML

Permiten leer documentos XML y comprobar el correcto uso del marcado, comprobar su validez mediante un análisis léxico y sintáctico, y ofrecer una representación de los documentos procesados junto con una interfaz para manipular y consultar los elementos de información contenidos en los documentos. (Lam, 2008, p. 33 & Birbeck, 2001, p. 37)

### 2.2.25 Agujero de Seguridad

Es una vulnerabilidad implícita o explícita de un sistema, que potencialmente puede ser objeto de vectores de ataque para obtener accesos no autorizados a los datos o las funcionalidades.

### 2.2.26 Ontología

Una ontología es una especificación explícita de una conceptualización. El término está tomado de la filosofía, donde una ontología es una cuenta sistemática de la existencia.. (Gruber, 1993, p. 5). Las ontologías, para representar el conocimiento, precisan los siguientes componentes: conceptos, relaciones, funciones, instancias y axiomas. Retomando la definición de Gruber, si se logra especificar los componentes y sus relaciones de un ámbito del conocimiento siguiendo un formalismo estricto codificado en un lenguaje informático (no de programación, sino de descripción), entonces es una ontología.

# 3. METODOLOGÍA

Para la elaboración de este modelo, se utilizaron varias metodologías integradas una dentro de otra, las cuales se ejemplifican en el siguiente diagrama:

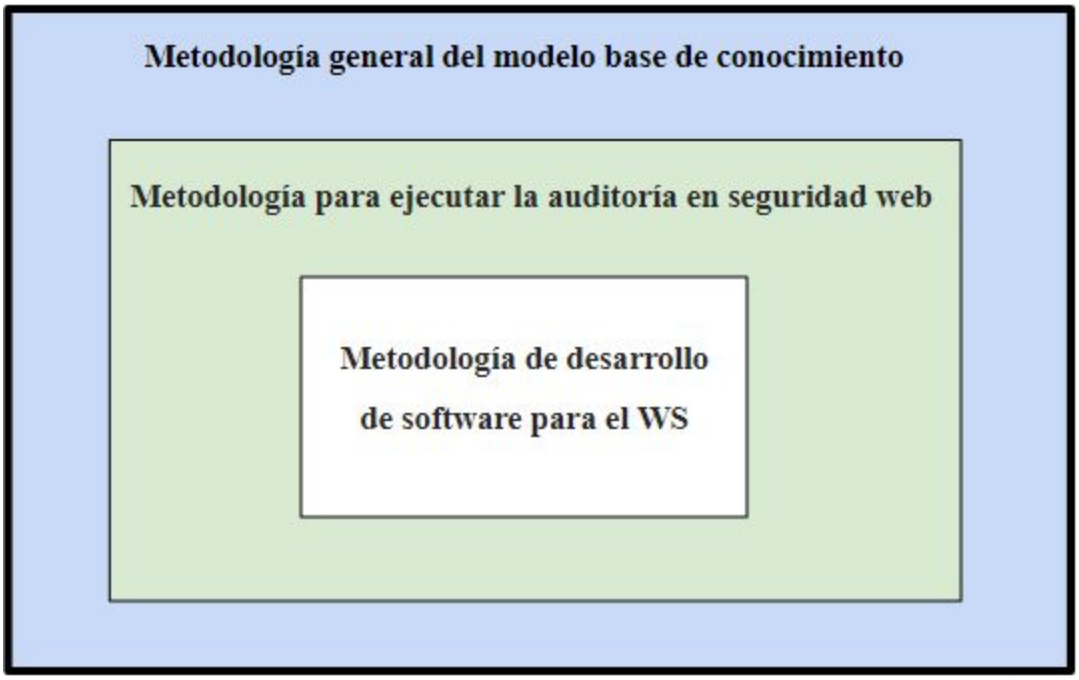



Figura 1. Ejemplificación de metodologías utilizadas en el proyecto

## 3.1 Metodología general del modelo base de conocimiento

Esta sección constituye la definición metodológica general para el desarrollo del modelo en sus diferentes elementos; por lo tanto, se contextualizan a continuación varias investigaciones, artículos, trabajos y definiciones de los componentes esenciales al momento de realizar un análisis a los modelos de base de conocimiento.

Éste proyecto incluye dos tipos de metodologías de trabajo, la primera que corresponde al diseño del prototipo del sistema recolección, procesamiento y depuración de la base de conocimiento - para preparar el modelo prototipo; y la segunda, que corresponde al proceso de sistematización del modelo de diagnóstico. Ambas metodologías tienen como soporte la investigación y aunque se dan de manera simultánea, para efectos de presentación se analizan por separado. (Méndez y Álvarez, 2003, p. 15-17)

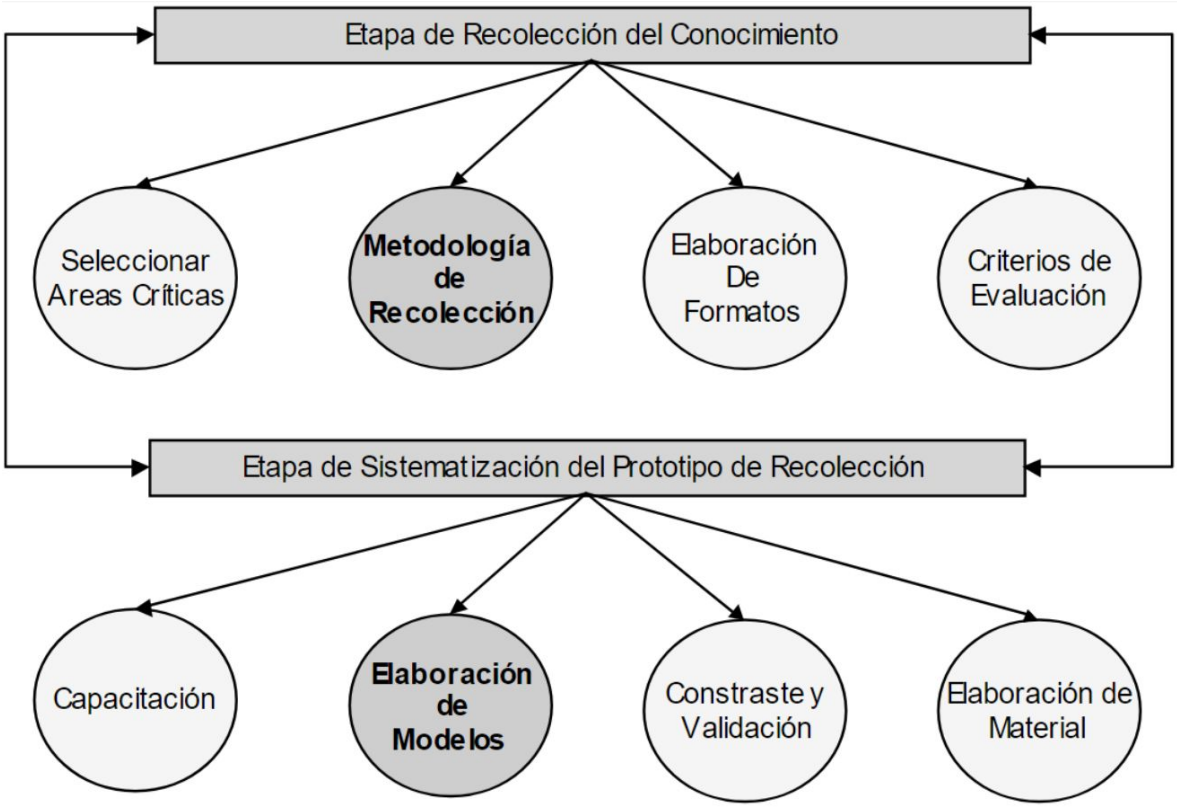



Figura 2. Metodología de desarrollo para la Base de Conocimiento (Méndez y Álvarez, 2003, p. 15-17)

### 3.1.1 Primera Etapa: Recolección del conocimiento

Esta fase corresponde a la adquisición del conocimiento del sistema, este puede ser adquirido por medios indirectos como libros, estudios realizados por sectores oficiales, tesis, diagnósticos sectoriales, etc., incluye (Méndez y Álvarez, 2003, p. 18):

a) Determinar las áreas críticas para el diagnóstico del sistema
b) La metodología de adquisición de conocimiento
c) La elaboración de formatos para recolección de datos y de conocimiento
d) La determinación de criterios de evaluación tecnológicos

### 3.1.2 Segunda Etapa: Implementación del prototipo

En esta etapa se elabora el modelo de dinámica del sistema que permita entender el funcionamiento del sistema, de tal suerte que pueda posteriormente aplicarse al proceso de

elaboración del sistema experto. Esta fase incluye las siguientes actividades: (Méndez, Álvarez, 2003, p. 18)

a) Inducción del grupo de investigadores en la temática de la dinámica del sistema, la teoría de la causalidad y los modelos mentales y se deben dar clases de soporte

b) Incluye el desarrollo de los modelos mentales por área de interés seleccionada

c) Una vez se realice la integración de los diferentes modelos mentales, matemáticos y de simulación, se debe abordar la evaluación y contraste del modelo integral

### 3.1.3 Elementos de evaluación de un modelo de base de conocimiento

Para determinar los componentes del modelo se consideró la lógica general del proceso de construcción de sistemas basados en ontologías. Para esto se ha tenido en cuenta el contexto de la gestión de proyectos, desde una perspectiva que responda a las particularidades, objetivos y exigencias de este proceso. La estructura del modelo está integrada en cinco secciones que permiten de manera lógica expresar su contenido, o sea, las subestructuras o pasos que componen cada una de estas secciones. (Fernández, López y Prevot, 2015, p. 4)

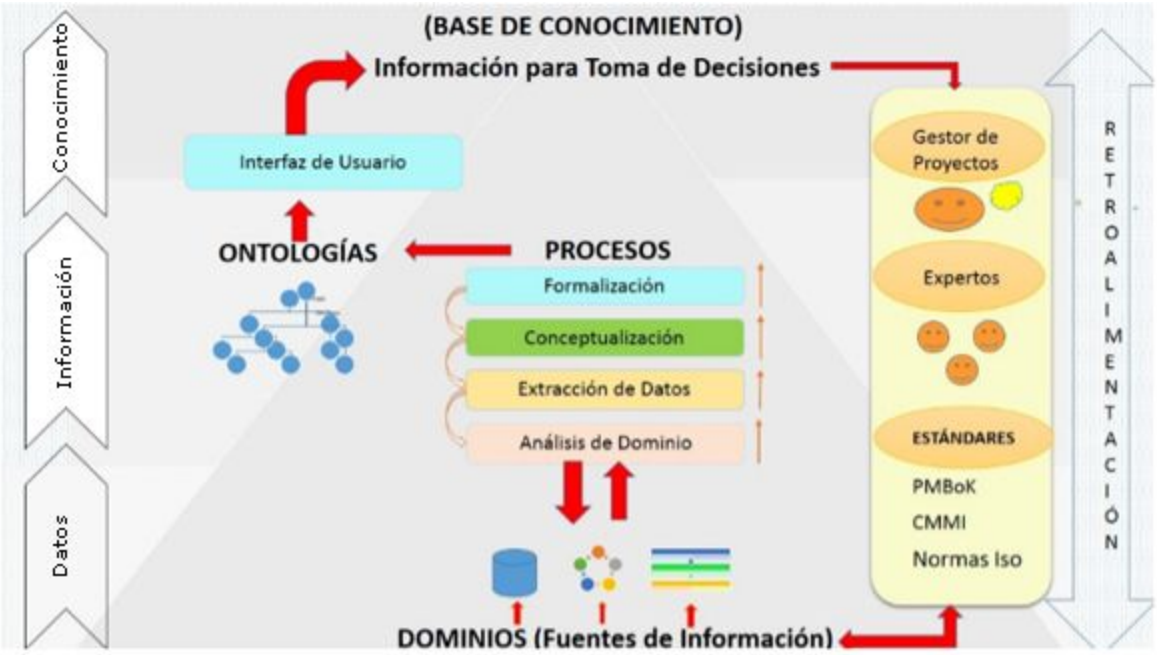



Figura 3. Elementos evaluación del modelo (Fernández, López y Prevot, 2015, p. 5)

**Análisis de dominio (Dominio aplicación objeto de auditoría)**

Preparación del escenario o detección de necesidades. En este paso se debe organizar un encuentro con los participantes que se tendrán en cuenta para desarrollar el proceso de construcción del sistema. Se debe presentar el proyecto, se explican los resultados que se obtendrán a partir de la propuesta del modelo y se identifican los requerimientos necesarios para la construcción del sistema basado en ontologías. Las actividades en las que intervienen los participantes tendrán que ver con su visión clara en la aplicabilidad del modelo. Se recogen sus criterios y consideraciones al respecto, su nivel de compromiso para lograr el diseño y la construcción del sistema basado en ontologías y la importancia que requiere la revisión de toda la información existente que pueda resultar útil para comprender el dominio de Gestión de Proyectos, lo que constituye el objetivo fundamental de la actividad a realizar. Los participantes podrán expresar sus opiniones sobre cómo consideran que debe ser el sistema que será desarrollado y ante qué interrogantes debe responder, entre otros elementos que consideren importantes. (Fernández et al, 2015, p. 6)

**Extracción de datos (Datos Fuentes recopilados como resultados de la auditoría)**

La extracción de datos recopilados se refiere a la depuración o refinamiento de los resultados que se obtienen a partir de las técnicas aplicadas, mediante mecanismos de retroalimentación. En este caso se revisarán los resultados con los involucrados a partir de actividades interactivas para recoger sus opiniones definitivas sobre la concepción del dominio de gestión de proyectos: (Fernández et al, 2015, p. 7)

a) Extracción del conocimiento explícito e implícito de las fuentes de información. La extracción de los datos obtenidos es donde se organizan todos los resultados a partir de las técnicas aplicadas, se obtienen mapas conceptuales o modelos abstractos del pensamiento o representación que tienen los involucrados del dominio.

b) Refinamiento de los elementos seleccionados como dominio representativo. Una vez alcanzados los resultados de los encuentros se procederá al refinamiento del dominio a partir de la aplicación de otras técnicas como es el caso de las encuestas, como proceso de refinamiento de los resultados alcanzados.

**Conceptualización (Modelo conceptual Sql Injection y definición terminológica del dominio)**

A medida que el desarrollo de la ontología avanza, es relevante categorizar todos y cada uno de los términos definidos como áreas o conceptos representativos del dominio. En la conceptualización deberán quedar definidas las clases o conceptos que se tendrán en cuenta para el diseño de la ontología; por eso se le llama a este paso definición terminográfica del dominio. Es uno de los pasos de mayor complejidad, ya que a partir de aquí se realizará todo el diseño posterior de la ontología. (Fernández et al, 2015, p. 8)

**Formalización (Documento maestro y Scripts)**

Uno de los mayores inconvenientes a la hora de desarrollar una ontología es la selección del lenguaje de representación, el cual debe ser el más adecuado para el usuario final de la ontología. Para esto han sido desarrolladas herramientas como editores de ontologías en un determinado lenguaje. Estas herramientas proporcionan un núcleo de servicios relacionados con ontologías y se pueden extender fácilmente con otros módulos. En la etapa de formalización se selecciona la herramienta que contendrá o modelará cada uno de los elementos que se tuvieron en cuenta en el diseño de la estructura ontológica. (Fernández et al, 2015, p. 9)

**Interfaz de usuario (Modelo gráfico de aplicación)**

Un sistema de recuperación de información, en su forma más simple, puede verse como una "caja negra" que acepta inputs y produce outputs. Durante este proceso realiza actividades que incluyen: el reconocimiento de la estrategia de búsqueda planteada, la aplicación de diferentes algoritmos de recuperación y de ordenación de los resultados según su relevancia o utilidad probable para el usuario, la selección de ítems de información (documentos o su representación), etc. Intentan localizarlos y recuperarlos tan veloz y económicamente como sea posible, por lo que su valor depende de su capacidad para identificar rápida y correctamente la información útil, de su facilidad para rechazar los ítems extraños o irrelevantes y de la versatilidad de los métodos propuestos. De esto trata la interfaz de usuario, la cual es la que permite realizar las consultas en tiempo real al sistema de información basado en ontologías. (Fernández et al, 2015, p. 10)

Figura 4. Representación de interfaz de usuario en el modelo (Fernández et al, 2015, p. 11)

### 3.2 Metodología para ejecutar la Auditoría en Seguridad web

Para la preparación de la auditoría, se presentará la metodología, los tiempos y recursos que se utilizarán. Se recolecta y analiza la información evidenciando los hallazgos, las oportunidades de mejora y las fortalezas encontradas durante la auditoría. Una vez se culmine, se presentarán las conclusiones. Con base en el informe de la auditoría, se establecerán las acciones de mejora pertinentes (MinTIC, 2016, p. 14).

La metodología inicia con un proceso de planeación, en esta se fijan los objetivos y las herramientas a usar, esto implica que hacer, cómo hacerlo y cuándo hacerlo. Esta etapa incluye una investigación previa con el fin de conocer la operación de lo que se va a evaluar. (MinTIC, 2016, p. 15)

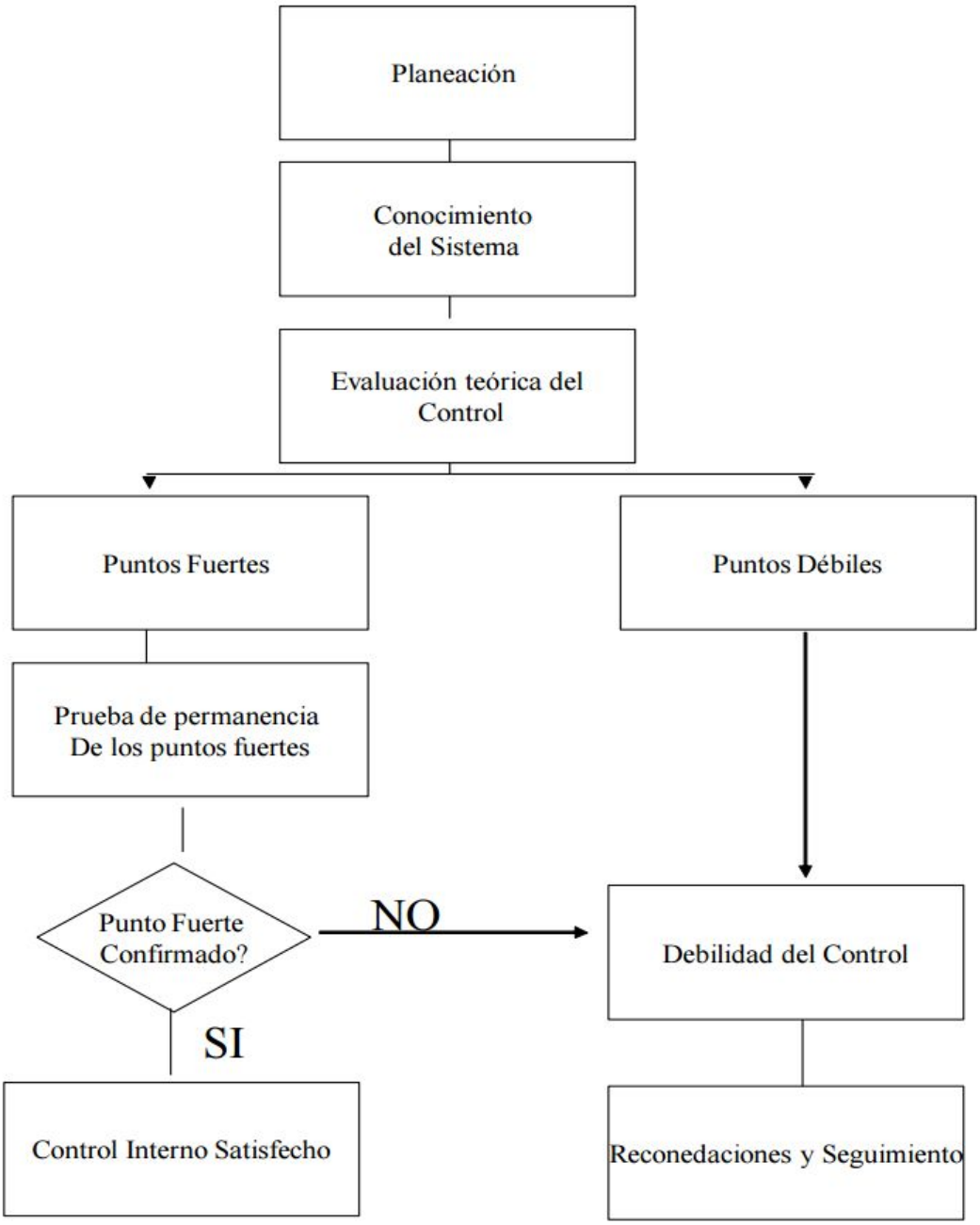



Figura 5. Metodología para la Auditoría. (MinTIC, 2016, p. 16)

Para el desarrollo de la auditoría es importante tener presente que dentro de la metodología general se incluyeron una serie de guías en cada una de las fases del modelo, el proyecto hará uso de estándares, buenas prácticas, herramientas y consejos profesionales tales como: NTC-ISO-IEC 27000, 27001, 27002, 27005, 31000, OSSTMM , OWASP, JUnit y las guías de Gestión de Riesgos y Auditoría emitidas por el MinTIC.

### 3.2.1 Introducción, identificación, análisis y detección de ataques sql injection

Esta parte constituye el objeto de estudio del proyecto; por lo tanto, se hizo una aproximación al objeto en cuestión desde el dominio técnico, de evolución histórica del concepto, de su impacto social (en la industria, en la calidad de los datos, etc). Esta búsqueda recopila varias definiciones, trabajos e investigaciones del término bajo diferentes revisiones:

La inyección de SQL es una de las vulnerabilidades más devastadoras que afectan a una empresa, ya que puede generar la exposición de toda la información confidencial almacenada en la base de datos de una aplicación, incluida información útil como nombres de usuario, contraseñas, nombres, direcciones, números de teléfono y detalles de tarjetas de crédito. Es la vulnerabilidad que se produce cuando le da a un atacante la capacidad de influir en las consultas del lenguaje de consulta estructurado (SQL) que una aplicación pasa a una base de datos de back-end. Al ser capaz de influir en lo que se pasa a la base de datos, el atacante puede aprovechar la sintaxis y las capacidades del propio SQL, así como la potencia y la flexibilidad de la funcionalidad de la base de datos de apoyo y la funcionalidad del sistema operativo disponible para la base de datos. La inyección de SQL no es una vulnerabilidad que afecta exclusivamente a las aplicaciones web; cualquier código que acepte la entrada de una fuente no confiable y luego la use para formar sentencias de SQL dinámico podría ser vulnerable. (Clarke, 2012, p. 16)

Una vulnerabilidad de inyección de SQL es un tipo de agujero de seguridad que se encuentra en una aplicación de múltiples niveles; es donde un atacante puede engañar a un servidor de base de datos para que ejecute una consulta de SQL arbitraria, no autorizada y no intencionada, agregando elementos de SQL adicionales sobre una consulta predefinida y existente que estaba destinada a ser ejecutada por la aplicación. La aplicación, que generalmente es, pero no necesariamente, una aplicación web, acepta la entrada del usuario e incrusta esta entrada dentro de una consulta SQL. Esta consulta se envía al servidor de base de datos de la aplicación donde se ejecuta. Al proporcionar cierta entrada mal formada, un atacante puede manipular la consulta SQL de tal manera que su ejecución tenga consecuencias no deseadas. (Litchfield, 2005, p. 2)

Sin embargo, cabe aclarar que dichas consultas de SQL no son necesariamente cláusulas no intencionadas sino cláusulas cuya estructura está siendo definidas de manera deliberada por el atacante. Las cláusulas SQL manipuladas pueden acarrear consecuencias no deseadas para el propietario y los usuarios del producto mientras cumplen con los objetivos del vector de ataque.

En este trabajo se aborda uno de los ataques más frecuentes, severos y dinámicos a nivel de la capa de aplicación en sistemas distribuidos y se propone un modelo de solución. En concreto se estudian los ataques de inyección SQL (Litchfield, et al., 2005 & Halfond, et al., 2006), ejecutados contra las aplicaciones dinámicas que gestionan una base de datos,

ejecutados contra entornos de servicios Web. Los ataques de inyección SQL, consisten en la inyección de código SQL sobre las consultas SQL enviadas a la base de datos. Aunque este tipo de ataque es conocido y ha sido ampliamente estudiado en la actualidad sigue siendo uno de los ataques más frecuentes y resulta difícil de detectar debido a la amplia variedad de técnicas utilizadas y continua evolución. Lo que desean los ataques de inyección SQL es la extracción de información y el escalado de privilegios; en otros casos es "*afectar la disponibilidad de las aplicaciones, los datos y los servicios disponibles, inhabilitando el acceso a los usuarios autorizados.*" (Pinzón, 2010, p. 3)

La inyección de SQL es ahora una técnica bien conocida en la comunidad de seguridad, pero una gran cantidad de aplicaciones de las organizaciones aún son vulnerables. En mi experiencia, alrededor de un tercio de las aplicaciones web tienen algún tipo de inyección SQL. Normalmente, esto se puede usar para comprometer completamente el servidor de base de datos back-end y, en los casos más graves, puede actuar como un conducto directamente a la red interna de la organización. (Litchfield, et al., 2005, p. 200)

Los ataques de inyección de SQL (SQLIAs) son especialmente dañinos. Los SQLIA pueden dar a los atacantes acceso directo a la base de datos que subyace a una aplicación y les permite filtrar información confidencial o incluso confidencial. Hay muchos ejemplos de SQLIA con graves consecuencias, y la lista de víctimas de tales ataques incluye compañías y asociaciones de alto perfil, como Travelocity, FTD.com, Creditcards.com, Tower Records y RIAA. Aún más alarmante es un estudio realizado por el Grupo Gartner en más de 300 sitios web, que encontró que el 97% de los sitios auditados eran vulnerables a este tipo de ataque web. De hecho, los SQLIA se han descrito como una de las amenazas de seguridad más graves para las aplicaciones web. (Halfond y Orso, 2005a, p. 1)

La inyección SQL se refiere a una clase de ataques de inyección de código en los que los datos proporcionados por el usuario se incluyen en una consulta SQL de tal manera que parte de la entrada del usuario se trata como un código SQL. Los SQLIA son un tipo de vulnerabilidad que, en última instancia, está causada por una validación de entrada insuficiente: se produce cuando los datos proporcionados por el usuario no se validan correctamente y se incluyen directamente en una consulta SQL. Al aprovechar estas vulnerabilidades, un atacante puede enviar comandos SQL directamente a la base de datos. Este tipo de vulnerabilidad representa una seria amenaza para cualquier aplicación web que lee los comentarios de los usuarios (por ejemplo, a través de formularios web o API web) y la usa para realizar consultas SQL a una base de datos subyacente. La mayoría de las aplicaciones web utilizadas en Internet o dentro de las empresas funcionan de esta manera y, por lo tanto, podrían ser vulnerables a la inyección de SQL. Un ataque de inyección SQL (SQLIA) ocurre cuando un atacante intenta cambiar la lógica, la semántica o la sintaxis de una declaración SQL legítima al insertar nuevas palabras clave u operadores de SQL en la declaración. (Halfond y Orso, 2005a, p. 2-3)

La inyección SQL es una clase de ataques de inyección de código en los que la entrada del usuario se incluye en una consulta SQL de tal manera que parte de la entrada se trata como código. Al usar la inyección SQL, los atacantes pueden filtrar información confidencial, como números de tarjetas de crédito, de las bases de datos de las aplicaciones web e incluso corromper la base de datos. Existen técnicas novedosas para contrarrestar la inyección de SQL. (Halfond y Orso, 2005b, p. 1)

Los ataques de inyección SQL representan una seria amenaza para la seguridad de las aplicaciones web: permiten a los atacantes obtener acceso sin restricciones a las bases de datos subyacentes a las aplicaciones y a la información potencialmente sensible que contienen estas bases de datos. Aunque los investigadores y los profesionales han propuesto varios métodos para abordar el problema de la inyección de SQL, los enfoques actuales no abordan el alcance completo del problema o tienen limitaciones que impiden su uso y adopción. Muchos investigadores y profesionales están familiarizados con solo un subconjunto de la amplia gama de técnicas disponibles para los atacantes que intentan aprovechar las vulnerabilidades de inyección de SQL. Muchas soluciones propuestas en la literatura abordan sólo algunos de los problemas relacionados con la inyección de SQL, sin embargo, hay revisiones exhaustivas de los diferentes tipos de ataques de inyección SQL conocidos hasta la fecha. Para cada tipo de ataque, se proporcionan descripciones y ejemplos de cómo se podrían realizar ataques de ese tipo. Existen análisis de técnicas de detección y prevención existentes contra los ataques de inyección de SQL. Para cada técnica, hay fortalezas y debilidades que abordan el rango completo de ataques de inyección SQL. (Halfond, et al., 2006, p. 1)

Las fallas de inyección ocurren cuando datos no confiables son enviados a un intérprete como parte de un comando o consulta. Los datos del atacante pueden engañar al intérprete al ejecutar comandos intencionados o acceder datos no autorizados. Características de este tipo de ataque:

- Los atacantes pueden ser atacantes externos, internos y administradores.
- Cualquier fuente de datos puede ser un vector de inyección, incluyendo fuentes internas.
- Es muy común que aparezca este tipo de amenaza dentro de las aplicaciones.
- Su detección no es complicada.
- Una falla de inyección puede resultar en la pérdida o corrupción de datos, falta de integridad, o negación de acceso, en algunos casos puede incluso tomar posesión del servidor.
- Dentro del impacto del negocio, se debe considerar el valor de los datos que pueden ser robados o modificados.

La inyección puede presentarse de diferentes formas de las que destacan LDAP (Lightweight Directory Access Protocol), XPath, inyección XML, inyección de comandos

de sistema operativo e Inyección SQL como el más común y peligroso según los estándares OWASP y CWE, el cual se detalla a continuación. (Minero, 2011, p. 13)

El rápido crecimiento de las aplicaciones para trabajar en internet ha generado diversas problemáticas entre las que se encuentran los ataques de inyección SQL. La mayoría de las aplicaciones hacen uso de algún producto de base de datos para gestionar y proporcionar información actualizada y ofrecer nuevos servicios a los usuarios. La gestión de la información en las bases de datos se realiza a través del lenguaje SQL, que constituye la columna vertebral de la mayoría de los sistemas de bases de datos, especialmente de las bases de datos relacionales. Una de las amenazas potenciales a nivel de las aplicaciones desarrolladas para trabajar en Internet que pone en riesgo la confidencialidad, integridad y disponibilidad de la información almacenada en las bases de datos, es el ataque de inyección de código o ataque de inyección SQL.

En un principio este tipo de amenaza a nivel de capa de aplicación se conocía como vulnerabilidad de código SQL, fue hasta el 2005 cuando fue acuñado el término "inyección" después de varias publicaciones sobre demostraciones de éste ataque. Este tipo de ataque es uno de los más frecuentemente utilizados contra las aplicaciones en Internet. (Meléndez, 2014, p. 36)

En las siguientes figuras extraídas del reporte de seguridad de IBM con la herramienta X-Force IRIS (Incident Response and Intelligence Services) entre 2011 y 2019, se muestran cómo los ataques de inyección de SQL han afectado diferentes tipos de organizaciones/industrias de diferentes países en diferentes momentos. También se determina que en el 2012 hubo la mayor cantidad de ataques de éste tipo, 79 en total; y a partir de ahí fue disminuyendo, sin embargo, no han desaparecido totalmente, incluso en enero de este año ocurrió un ataque en Dinamarca.

**X-Force IRIS Data Breach Report 2011**

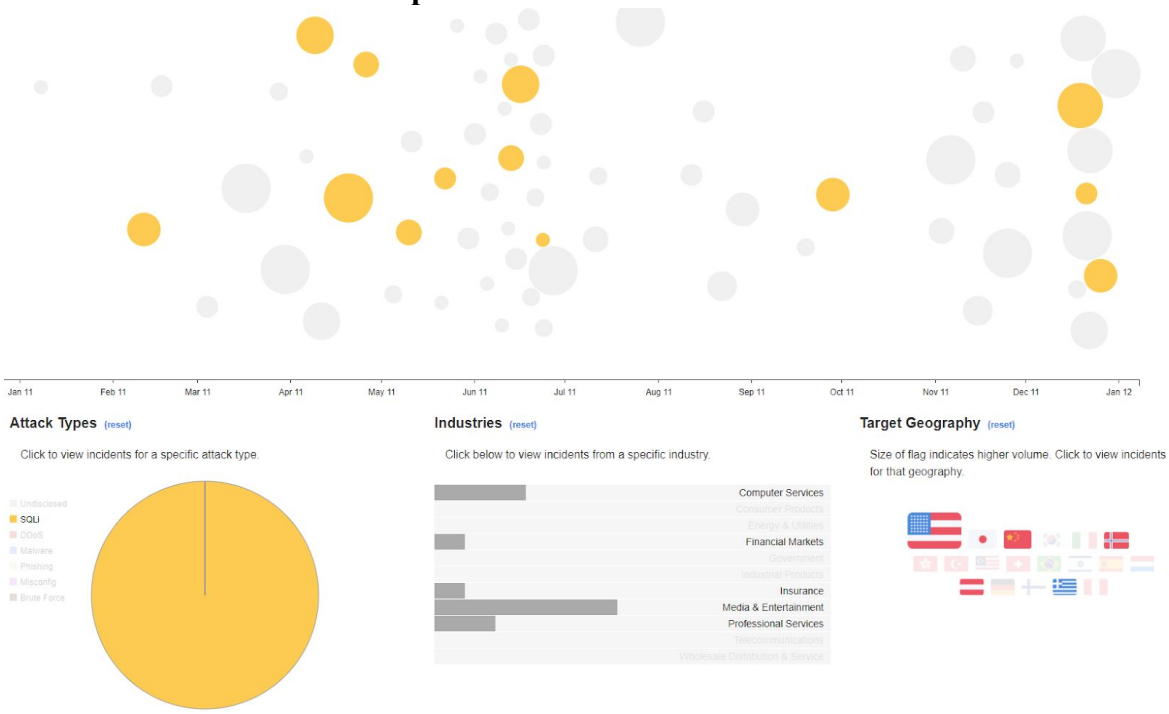



Figura 6. Evolución de ataques de inyección SQL durante 2011. (IBM Corporation, 2019)

**X-Force IRIS Data Breach Report 2012**

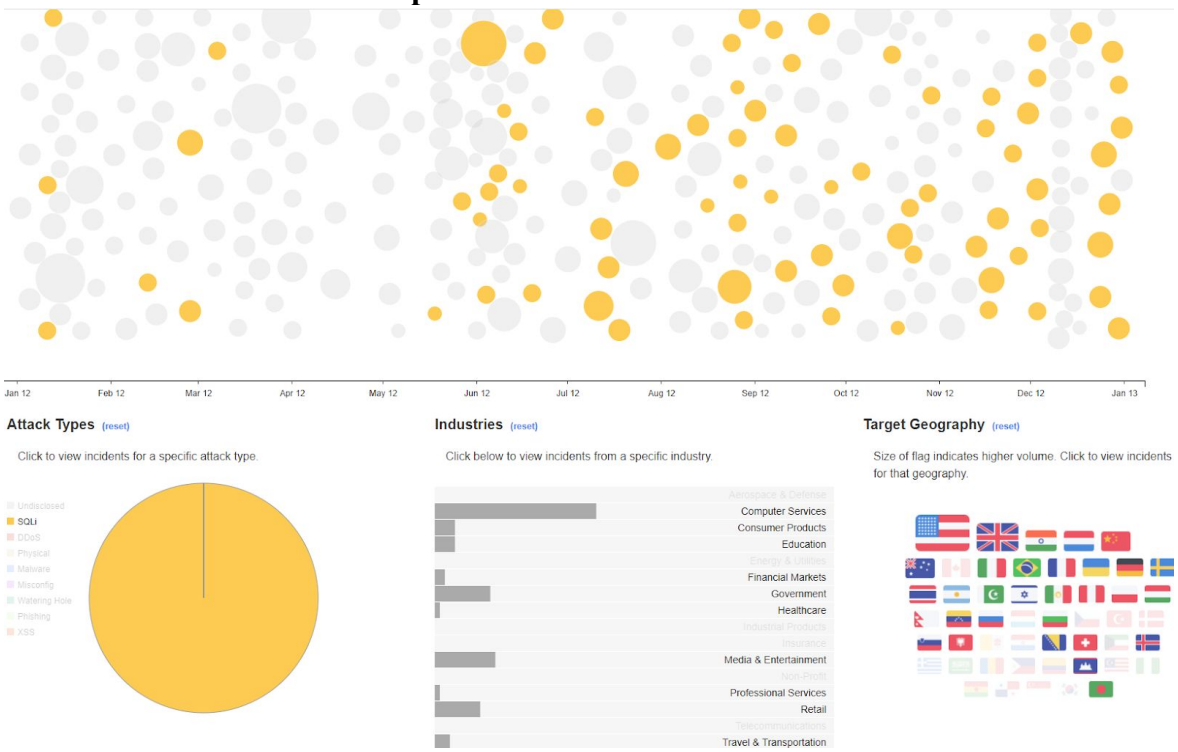



Figura 7. Evolución de ataques de inyección SQL durante 2012. (IBM Corporation, 2019)

**X-Force IRIS Data Breach Report 2013**

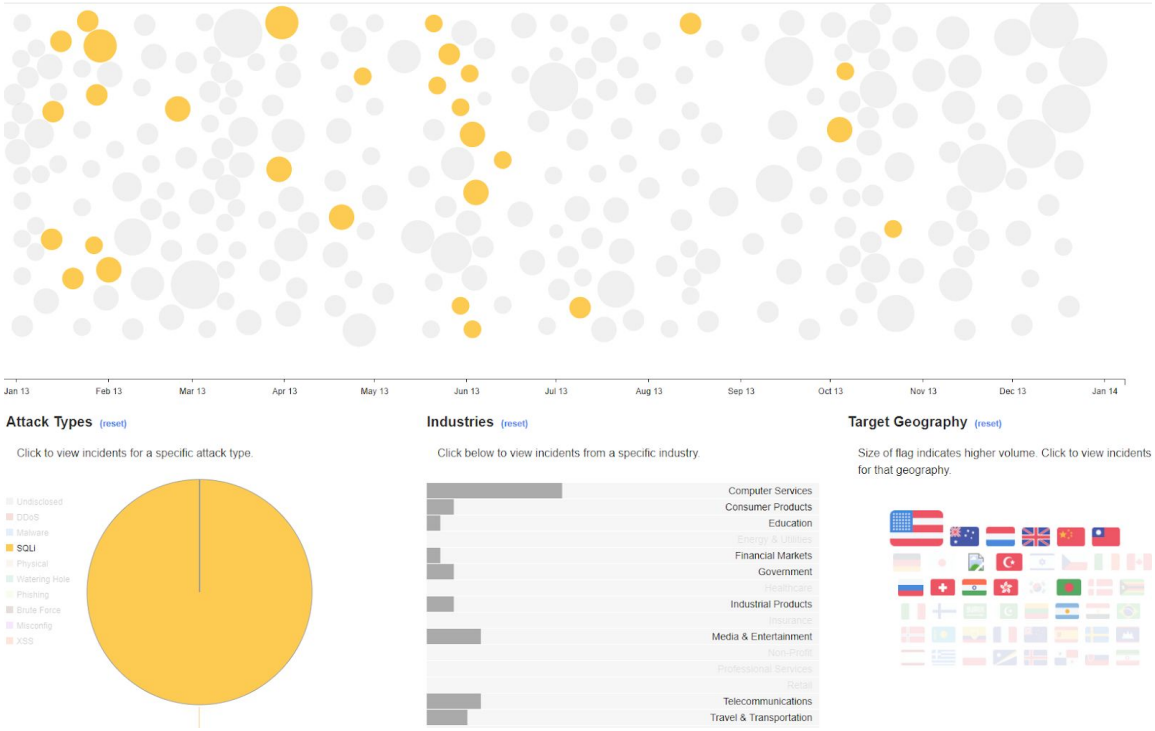



Figura 8. Evolución de ataques de inyección SQL durante 2013. (IBM Corporation, 2019)

**X-Force IRIS Data Breach Report 2014**

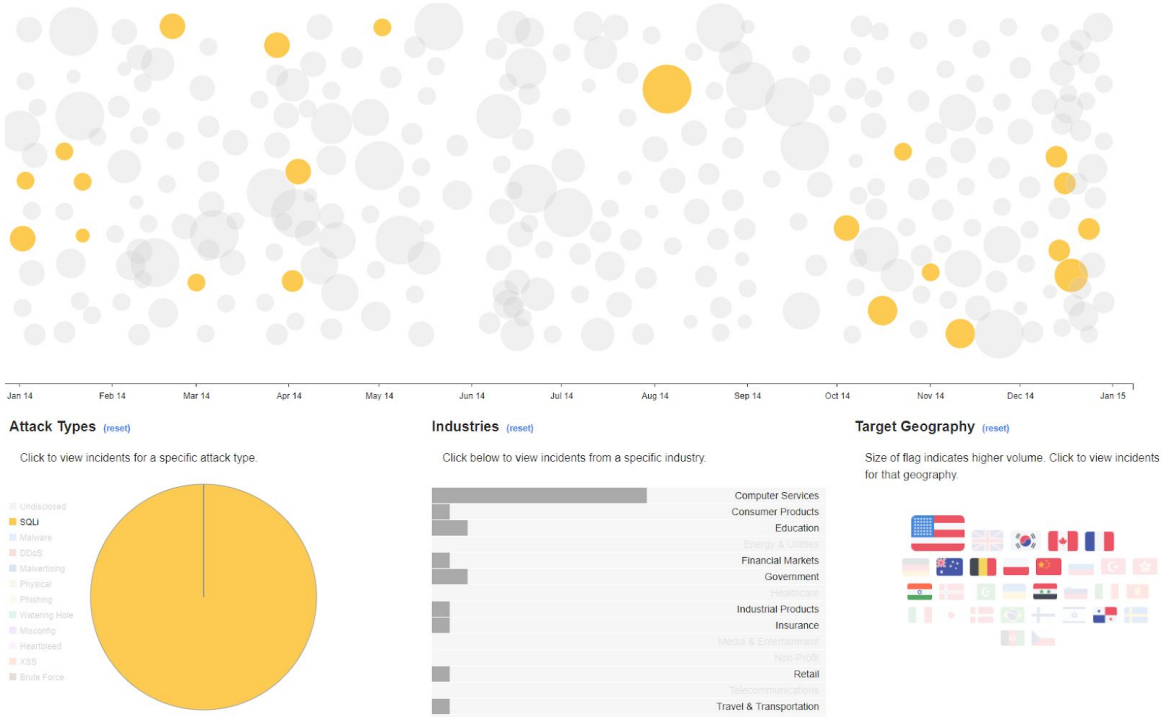



Figura 9. Evolución de ataques de inyección SQL durante 2014. (IBM Corporation, 2019)

**X-Force IRIS Data Breach Report 2015**

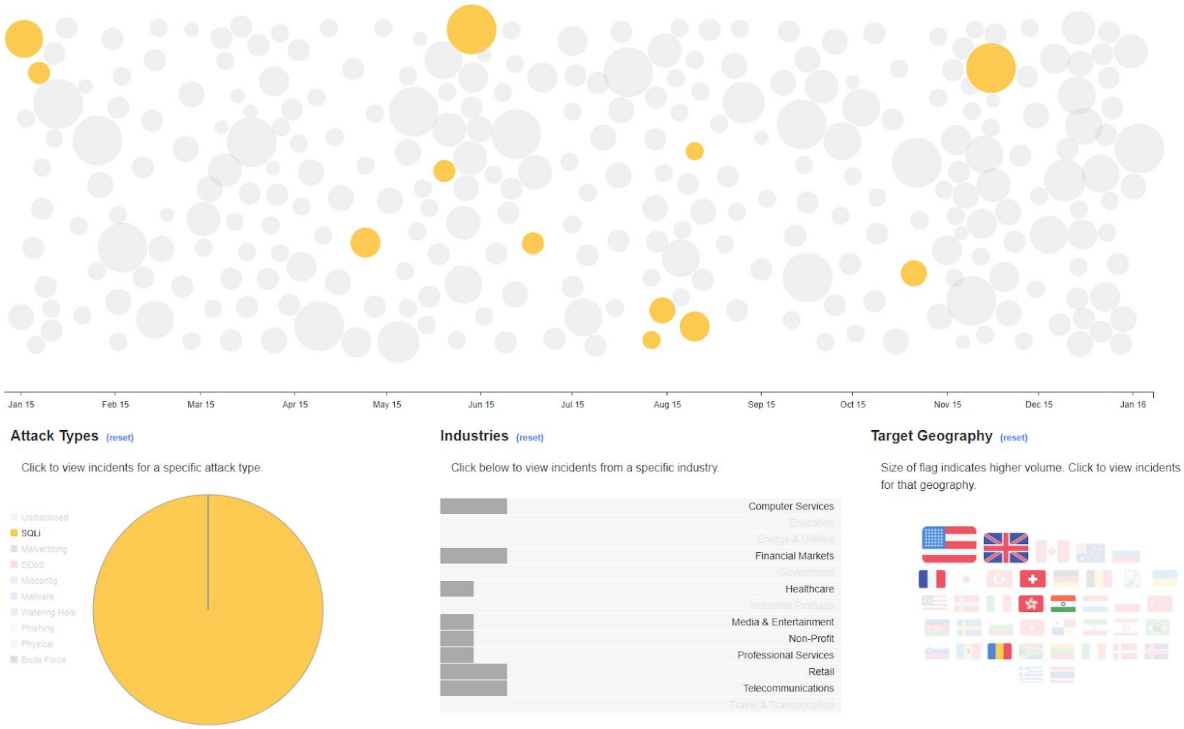



Figura 10. Evolución de ataques de inyección SQL durante 2015. (IBM Corporation, 2019)

**X-Force IRIS Data Breach Report 2016**

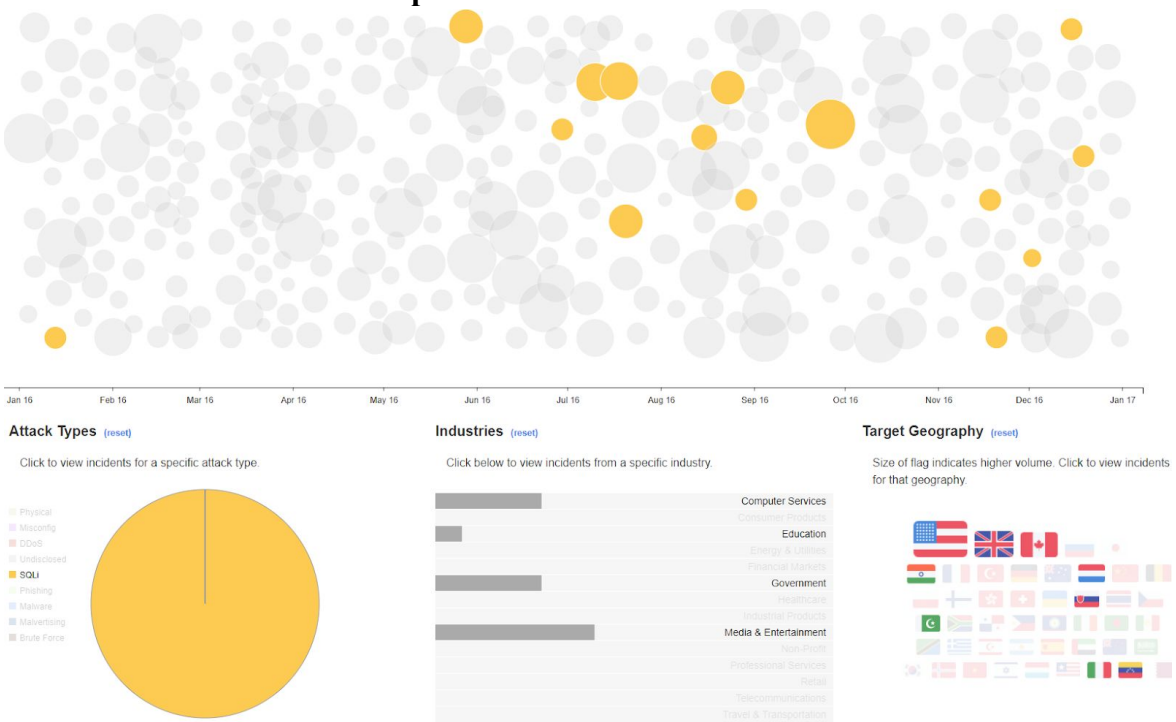



Figura 11. Evolución de ataques de inyección SQL durante 2016. (IBM Corporation, 2019)

**X-Force IRIS Data Breach Report 2017**

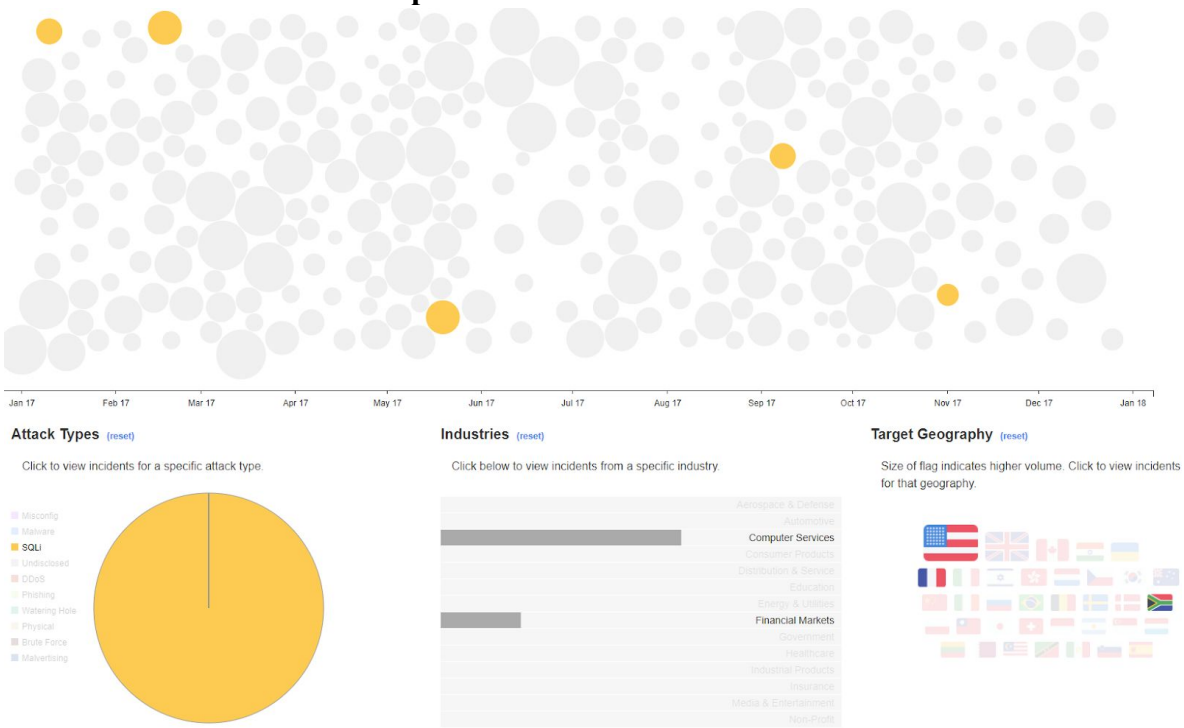



Figura 12. Evolución de ataques de inyección SQL durante 2017. (IBM Corporation, 2019)

**X-Force IRIS Data Breach Report 2018**

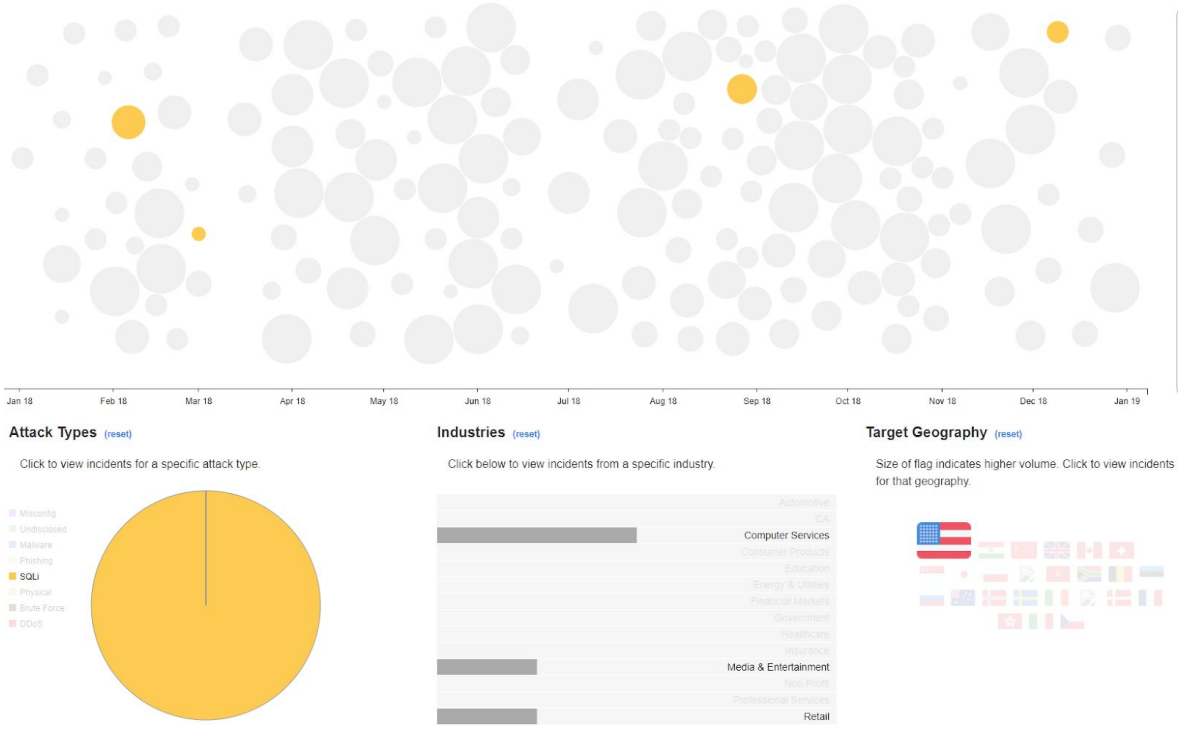



Figura 13. Evolución de ataques de inyección SQL durante 2018. (IBM Corporation, 2019)

**X-Force IRIS Data Breach Report 2019**

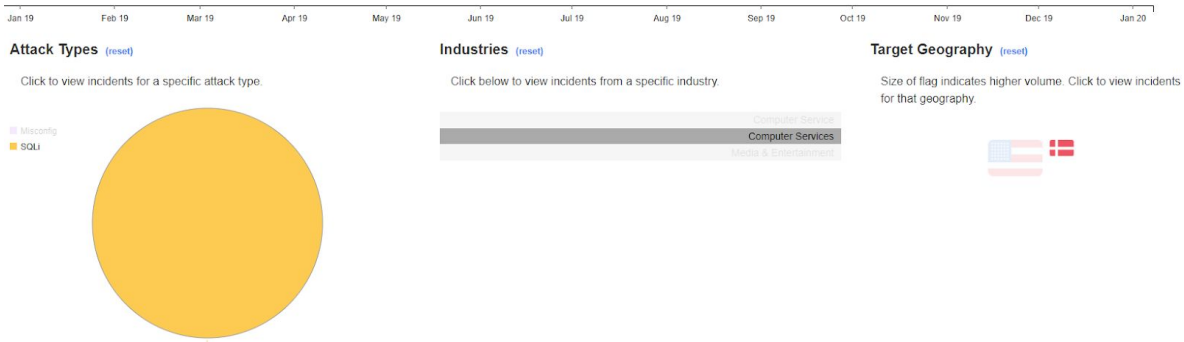



Figura 14. Evolución de ataques de inyección SQL durante 2019. (IBM Corporation, 2019)

Existe una plataforma de IBM llamada X-Force Exchange basada en nube para compartir inteligencia sobre amenazas dentro de las cuales está SQL Injection, que permite a los usuarios investigar rápidamente las amenazas más recientes para la seguridad, agregar inteligencia procesable y colaborar con colegas. IBM X-Force Exchange está soportado por la inteligencia generada por humanos y por máquinas que utilizan la escala de IBM X-Force, más adelante se hará un ejemplo de su uso.

Figura 15. IBM X-Force Exchange. (IBM Corporation, 2019)

El ataque SQL Injection consiste en la inserción o inyección de código SQL (Query) del lado del cliente de un aplicativo, si se realiza una inyección de código SQL exitosa se la puede explotar de las maneras ya habladas anteriormente, aplicativos web desarrollados con PHP o ASP son los más vulnerables a un ataque de este tipo, dependiendo de lo que desee realizar el atacante y de qué tan bien configurada este la seguridad a este medio de ataque, será la severidad de lo que pueda pasarle al aplicativo. SQL Injection en el 2013 se encontraba en el top 10 de los ataques más comunes y peligrosos a las Bases de Datos desde aplicativos web según el documento OWASP Top10 – 2013. Mediante este método de ataque puede realizar varias cosas dentro de una Base de Datos como son: (Coronado, 2017, p. 25)

- Descubrimiento de información que se encuentra dentro de la Base de Datos y qué es de acceso restringido, esto lo puede hacer por medio de modificación de consultas que se realizan a la base de datos y de esta manera acceder a registros u objetos a los que inicialmente no se tienen acceso por ser restringidos a un usuario normal.

- Elevación de Privilegios: Si el sistema de autenticación dentro de un aplicativo que use credenciales y los tenga almacenados dentro de un motor de Base de Datos hace que un atacante por medio de SQL Injection acceda a estos registros y cambie su credencial a un usuario con más privilegios dentro del sistema.

- Denegación de servicio: un atacante por medio de comandos puede causar daños directamente a la base de datos ya sea borrando registros, objetos o realizando un alto al servicio de la base de datos causando que ésta no responda en tiempos óptimos para los usuarios reales.

- Suplantación de usuarios: Al momento en que un atacante accede a los registros de credenciales donde se encuentran almacenados los datos de los usuarios, éste puede robarlos y hacerse pasar por otro usuario al momento de realizar acciones en el aplicativo. (Coronado, 2017, p. 25)

Para prevención de este tipo de ataques OWASP ha desarrollado 3 documentos para poder evitar ataques así, estos documentos son:

- OWASP SQL Injection Prevention Cheat Sheet.
- OWASP Query Parametrization Cheat Sheet.
- OWASP Guide article how to avoid SQL Injection Vulnerabilities

Por medio de estos documentos se puede tener una mejor idea de cómo configurar y desarrollar nuestros aplicativos web para no sufrir de un ataque de este tipo y en caso de hacerlo poder actuar inmediatamente para eliminar la vulnerabilidad. SQL Injection se ha convertido es uno de los problemas más comunes para aplicativos web al ser fácil de

detectar la vulnerabilidad y también al ser de fácil explotación y es por esto que todo aplicativo web que use base de datos para su almacenamiento será objeto de intento de este tipo de ataque, este ataque se da por medio de una persona al ingresar una serie de caracteres en el campo que irá a autenticación dentro de la Base de Datos a saber si es dicho usuario el que intenta acceder al aplicativo y en este momento es que se puede modificar la consulta que está escrita en código al juntarse con el campo de ingreso del atacante que lo realiza desde el lado cliente del aplicativo. (Coronado, 2017, p. 26)

La inyección de SQL abstracta es una de las vulnerabilidades en la Lista de las Diez Principales de OWASP para la Explotación de Aplicaciones Basadas en la Web. Este tipo de ataques tienen lugar en aplicaciones web dinámicas, ya que interactúan con bases de datos para diversas operaciones. El sistema de gestión de contenido actual como Drupal, Joomla o Wordpress tiene toda la información almacenada en sus bases de datos. Una sola intrusión en este tipo de sitios web puede llevar a un control general de los sitios web por parte de un atacante. Los investigadores son conscientes de los ataques básicos de inyección de SQL, pero hay numerosos ataques de inyección de SQL que aún no se han prevenido ni detectado. Aquí, presentamos la extensa revisión para el ataque de Inyección SQL avanzada, como la Inyección de SQL de flujo rápido, la Inyección de SQL compuesto y la Inyección de SQL ciego profundo. También analizamos la detección y prevención utilizando los métodos clásicos y los enfoques modernos. Vamos a discutir la evaluación comparativa para la prevención de la inyección SQL. (Singh, 2017, p. 1)

El Ataque de Inyección SQL ocurre cuando el atacante intenta insertar un código malicioso en la base de datos de la Aplicación Web que está destinada a la recuperación o corrupción de datos. Además, estos ataques se utilizan en los sitios web de comercio electrónico para la extracción de números de tarjetas de crédito o se utilizan ampliamente para omitir la autenticación. Su y Wassermann describen la Inyección de SQL de manera exhaustiva y formal con una explicación sobre la Inyección de Código, así como la validación utilizando la Comprobación de SQL. Para los campos del sitio web sin la validación de entrada adecuada, un atacante podría obtener acceso directo a la base de datos de la aplicación subyacente. (Singh, 2017, p. 2)

Los ataques de inyección permiten a un usuario malintencionado agregar o inyectar contenido y comandos en una aplicación para modificar su comportamiento. Estos tipos de ataques son comunes, generalizados, y es fácil para un pirata informático probar si un sitio web es vulnerable y fácil y rápido para que el atacante se aproveche. Hoy en día son muy comunes en aplicaciones heredadas que no se han actualizado. La vulnerabilidad de inyección más común es la inyección SQL. La vulnerabilidad de la inyección también es fácil de remediar y proteger. Esta vulnerabilidad cubre SQL, LDAP, Xpath, comandos del sistema operativo, analizadores XML.

La vulnerabilidad de la inyección puede llevar a:

- Divulgación / filtración de información sensible.
- Problemas de integridad de los datos. La inyección SQL puede modificar datos, agregar nuevos datos o eliminar datos.
- Elevación de privilegios.
- Obtener acceso a la red de back-end.

Los comandos SQL no están protegidos de entradas no confiables. El analizador de SQL no puede distinguir entre el código y los datos. (Chapman, 2017, p. 45)

Figura 16. Consulta SQL personalizada. (OWASP Foundation, 2018)

El uso de concatenación de cadenas para generar una declaración SQL es muy común en aplicaciones heredadas donde los desarrolladores no estaban considerando la seguridad. El problema es que esta técnica de codificación no le dice al analizador qué parte de la declaración es el código y qué parte son los datos. En situaciones donde la entrada del usuario se concatena en la declaración SQL, un atacante puede modificar la declaración SQL agregando un código SQL a los datos de entrada. (Chapman, 2017, p. 46)

De acuerdo a lo anterior se muestra a continuación una clasificación que distingue una inyección SQL de otros tipos de ataques, esto con el fin de delimitar el alcance del objeto de negocio del presente trabajo:

**Ataques de inyección SQL**

Utilizan las vulnerabilidades en las aplicaciones y aprovechan funcionalidades extras en los sistemas de base de datos. Un ataque de inyección SQL se produce cuando la consulta original es modificada mediante la inyección de elementos propios del lenguaje SQL de forma arbitraria y no autorizada. El medio tradicional para ejecutar un ataque de inyección SQL es a través de las interfaces de las aplicaciones, donde los usuarios proporcionan los parámetros de entrada. Generalmente, los ataques de inyecciones SQL aprovechan la ausencia de mecanismos de validación sobre los parámetros de entrada, los cuales son concatenados de forma directa a la cadena de texto de la consulta SQL. Una vez construida la consulta SQL, ésta es enviada a la base de datos para su procesamiento, materializándose el ataque. (Pinzón, 2010, p. 32). Los ataques de inyección SQL han ido evolucionando, dando origen a varios tipos de ataque. (Halfond, et al., 2006 & D. B. Networks, 2009).

En la siguiente tabla se presentan los tipos más comunes de ataques de inyección SQL.

| **Tipo de inyección** | **Descripción** | **Ejemplo** |
|---|---|---|
| Tautologías | Se inyecta código en la cadena SQL, de la consulta, en una o en todas las declaraciones condicionales, de forma que siempre sean evaluadas como verdaderas. En este caso se aprovecha un parámetro de entrada vulnerable, usado para construir la condición. El ataque resulta efectivo, cuando de la tabla objetivo, se logra recuperar cuando menos un registro. | SELECT * from TblUser where Username = ‘’ or 1=1--and pass= ‘’ |
| Errores Lógicos / Consultas Ilegales | Se aplica reingeniería sobre los mensajes de errores, enviados desde la base de datos como respuesta, para obtener información acerca del esquema de la base de datos. Un ejemplo es la adición de la comilla dentro de la cadena de texto suministrada desde el parámetro de entrada. | SELECT * FROM TblSupplier WHERE NameSupplier = ‘O’Reill’ |
| Basados en el Operador Union | Es posible modificar el conjunto de resultados de la consulta original, cambiando su lógica. A través de un parámetro de entrada vulnerable, se adiciona una segunda consulta mediante el operador union. Es posible recuperar resultados de diferentes tablas producto de la unión de las tablas de la consulta original y de la segunda consulta adicionada. | SELECT * FROM TblSupplier WHERE NameSupplier = ‘’ UNION ALL SELECT * From TblConsumer WHERE 1 = 1 |
| Piggy-backed Queries | Este tipo de ataque no modifica la lógica de la consulta original. A su vez, adiciona una nueva consulta totalmente diferente a la primera, con lo que la base de datos recibe más de una consulta SQL. Este ataque depende de la configuración de la base de datos, cuando permite múltiples consultas en una única cadena de texto SQL. | SELECT * FROM TblUser WHERE UserName = “; DROP TABLE TblUser —‘AND pass =” |

| | | |
|---|---|---|
| Inyección basada en Inferencia | La inyección a ciegas permite inferir tomando en cuenta el comportamiento de la página cuando se envía una pregunta (cierto/falso) al servidor. Si la respuesta es verdadera, la aplicación continúa con su funcionamiento normal. En caso contrario, el funcionamiento de la página difiere del habitual. El timing attacks es similar a la inyección a ciegas, pero utiliza un método de inferencia diferente. La estructura de la consulta a ejecutar es un enunciado de la forma SELECT IF (expression, true, false) donde en unas de las ramas se incluye una función de tiempo. Por ejemplo BENCHMARK()o WAITFOR DELAY. Midiendo el incremento o decremento del tiempo de respuesta, es posible determinar la rama seguida por la consulta e inferior en la respuesta. | Declare @s varchar(8000); select @s = db_name(); if (ascii(substring(@s, 1, 1)) & ( power(2, 0))) > 0 waitfor delay '0:0:5' |
| Inyección basada en Store Procedures | Lo primero es conocer cuál producto de base de datos está dando soporte. El problema se deriva de la extensión en la funcionalidad que la mayoría de los productos de bases de datos ofrecen mediante la incorporación de procedimientos almacenados. Estos procedimientos almacenados pueden incluso interactuar con el sistema operativo. Es posible elaborar consultas maliciosas que aprovechen la funcionalidad de un procedimiento almacenado para tomar el control del servidor o bloquearlo. | Simplequoted.asp?city=seat tl e';EXEC master.dbo.xp_cmdshell 'cmd.exe dir c: |
| Codificación Variable | Más que un tipo de ataque, esto permite evadir los mecanismos de detección y prevención de intrusiones. Mediante la codificación, es posible ocultar a los mecanismos de seguridad patrones conocidos de ataque. Para codificar las cadenas de texto, se utilizan varios métodos de codificación tales como hexadecimal, ASCII o UNICODE). Esta técnica de codificación hace difícil plantear algún | Ej.: declare @q varchar(8000); select @q = 0x73656c65637420404076 6 57273696f6e; exec(@q) El resultado de la codificación es 'select @@version' |

| | | |
|---|---|---|
| | mecanismo de seguridad basado en la revisión de código, sencillamente porque resulta difícil manejar todas las posibles variaciones. | |
| Combinación de Ataques | Las técnicas descritas arribas se pueden combinar para alcanzar más de un objetivo a la vez. Por ejemplo, es posible recuperar información del esquema de la base de datos mediante consultas ilegales, al mismo tiempo que se apunta a identificar el producto de la base de datos para aprovechar el uso de procedimientos almacenados que den acceso al sistema operativo. Todo ello, oculto bajo un método de codificación que no sea detectado por los mecanismos de seguridad. | |

Tabla 1. Tipos de ataques de inyección SQL (Pinzón, 2010, p. 34)

**Ataques XDoS**

En el caso concreto de los servicios Web, una extensión de los ataques DoS se ha convertido en la nueva amenaza para este tipo de aplicaciones. Conocido como ataque de denegación de servicio basado en XML (XML Denial of Service Attack, XDoS) (Im y Song, 2005; Moradian y Håkansson, 2006; Vorobiev y Han, 2006 & Gruschka y Luttenberger, 2006), este ataque toma ventaja de la incapacidad de las propuestas de seguridad existentes para direccionar esta amenaza. La probabilidad de un ataque XDoS aumenta considerablemente si tenemos en cuenta que los servicios Web están cimentados sobre una serie de estándares conocidos (Pulier y Taylor, 2005 & Cerami, 2002), incluyendo HTTP como el medio más habitual para el transporte de los mensajes y el estándar XML para la codificación de los mensajes. Los servicios Web reúnen las condiciones. idóneas para un ataque de este tipo debido a la flexibilidad nativa de los estándares, y a su naturaleza abierta. Las técnicas de ataque XDoS a nivel de los servicios Web se aprovechan, en la mayoría de los casos, del procesamiento costoso que puede requerir ciertos tipos de peticiones de usuario. Una revisión de las diferentes técnicas de ataques XDoS contra los servicios web fue llevado a cabo por (Moradian y Håkansson, 2006). La revisión incluye mecanismos de ataques que afectan la disponibilidad de los servicios Web (Moradian y Håkansson, 2006).

**Antecedentes y estado del arte Sql Injection en Web Services**

Una arquitectura de servicios Web está compuesta por una pila de protocolos. Esta pila ha ido evolucionando y actualmente está conformada por 4 capas importantes (Cerami, 2002):

- Servicio de Transporte: Es la responsable de transportar los mensajes entre las aplicaciones de red y los protocolos. Actualmente esta capa incluye el Protocolo de Transferencia de Hiper-Texto (HTTP, HyperText Transfer Protocol), Protocolo Simple de Transferencia de Correo (SMTP, Simple Mail Transfer Protocol), Protocolo para la Transferencia de Ficheros (FTP, File Transfer Protocol), y nuevos protocolos como el Protocolo de Intercambio de Bloques Extensible (Blocks Extensible Exchange Protocol, BEEP).

- Mensajería XML: Es la capa responsable de codificar los mensajes en un formato XML común, de forma que pueda ser entendido en ambos extremos. Actualmente, esta capa incluye XML-RPC y SOAP y REST.

- Descripción de Servicio: Es la capa responsable de describir la interfaz pública de un servicio Web específico. Actualmente la descripción de los servicios se maneja mediante un lenguaje de descripción de servicios Web (WSDL, Web Services Description Language).

- Descubrimiento de Servicios: Es la responsable de centralizar los servicios en un registro común y provee las funcionalidades para publicar y localizar servicios. El proceso de descubrimiento de servicio actualmente es llevado mediante la API Universal Description, Discovery, and Integration (UDDI). (Pinzón, 2010, p. 42)

En la Figura se muestra un resumen de la pila de protocolos de los servicios Web:

Figura 17. Pila de protocolos de los Servicios Web (Pinzón, 2010, p. 43)

Dentro de los componentes de la tecnología de servicios Web, la transmisión de los mensajes seguramente es uno de los aspectos más importantes. El protocolo estándar SOAP (The Simple Object Access Protocol), es la pieza clave dentro de la arquitectura de servicios Web (Newcomer y Lomow, 2004). Está basado en XML para el intercambio de información entre computadoras; es el protocolo de comunicación que define como los mensajes deben ser construidos, transmitidos de un nodo a otro y procesado por cada nodo en su ruta hasta alcanzar el nodo destino. SOAP asume que cada mensaje tiene un remitente y un receptor, y un número arbitrario de intermediarios (nodos) que procesan el mensaje y redirigen éste al nodo destino o receptor. Aunque SOAP puede funcionar en una variedad de protocolos de transportes, HTTP es el protocolo de transporte preferido. SOAP vía HTTP realiza un intercambio de mensajes basado en el modelo Solicitud/Respuesta semejante a una Llamada a Procedimiento Remoto (RPC). (Pinzón, 2010, p. 44)

**Revisando código de aplicaciones**

Realizando revisiones de código de SQL injection involucrando servicios web se encontró: Nils, una persona que trabaja en MWR InfoSecurity, creó una herramienta llamada "WebContentResolver" (http://labs.mwrinfosecurity.com/tools/android_webcontentresolver) que puede ejecutarse en un dispositivo Android (o emulador) y expone una interfaz de servicio web para todos los proveedores de contenido instalados. Esto nos permite usar un navegador web para probar vulnerabilidades y aprovechar el poder de las herramientas, como sqlmap (http://sqlmap.sourceforge.net), para encontrar y explotar vulnerabilidades en los proveedores de contenido. Es recomendable probarlo en caso de evaluar aplicaciones de Android. Para probar la explotación de la vulnerabilidad, la utilidad WebContentResolver debe instalarse junto con la aplicación vulnerable. La utilidad expone una interfaz de servicio web a todos los proveedores de contenido instalados. Podemos usar la utilidad WebContentResolver para enumerar el proveedor de contenido accesible como se ilustra a continuación: (Clarke, 2012, p. 163)

*$ curl http://127.0.0.1:8080/list*
*package: org.sapmentors.sapnoteview*
*authority: org.sapmentors.sapnoteview.noteprovider*
*exported: true*
*readPerm: null*
*writePerm: null*

Entonces podemos consultar al proveedor de contenido así:

> *http://127.0.0.1:8080/query?a=org.sapmentors.sapnoteview.noteprovider?&selNa me=_id&selId=1*
> *1223*

Query successful:

*Column count: 3*
*Row count: 1*
*| _id | suggest_text_1 | suggest_intent_data*
*| 11223 | secret text | 11223*

La instrucción SQL que se ejecuta realmente se ilustra a continuación:

*SELECT _id, title AS suggest_text_1, _id AS suggest_intent_data FROM notes WHERE (_id=11223)*

Entonces podemos probar la inyección de SQL dentro de la selección así:

*$ curl*
*http://127.0.0.1:8080/query?a=org.sapmentors.sapnoteview.noteprovider?&selNa*
*me=_id&selId=1*
*1223%20or%201=1*

Query successful:

*Column count: 3*
*Row count: 4*
*| _id | suggest_text_1 |suggest_intent_data*
*| 11223 | secret text | 11223*
*| 12345 | secret text | 12345*
*| 54321 | super secret text | 54321*
*| 98765 | shhhh secret | 98765*

La instrucción SQL que se ejecuta se presenta a continuación:

*SELECT _id, title AS suggest_text_1, _id AS suggest_intent_data FROM notes WHERE (_id=11223 or 1=1)*

Se debe tener en cuenta que los parámetros selName y selId son vulnerables. La explotación puede entonces automatizarse utilizando sqlmap:

*$ ./sqlmap.py -u*
*"http://127.0.0.1:8080/query?a=org.sapmentors.sapnoteview.noteprovider?&selNa*
*me=_id&selId=*
*11223'-b--dbms=sqlite*
*sqlmap/1.0-dev (r4409) - automatic SQL injection and database takeover tool*
*http://www.sqlmap.org*

**Filtrando mensajes de Servicios web**

El patrón de filtro de intercepción también es fácil de aplicar a los servicios web XML con filtros de entrada y salida personalizados. Un filtro de entrada podría realizar la validación de los parámetros del método y registrar los intentos de inyección de SQL. También se puede usar un filtro de salida para suprimir los detalles del error, como los que a menudo se filtran en la cadena de fallas de un mensaje de falla de SOAP. Las plataformas .NET Web Services y Apache Axis, por ejemplo, proporcionan mecanismos para filtrar los mensajes entrantes y salientes. (Clarke, 2012, p. 563)

ModSecurity también puede manejar mensajes XML entrantes para realizar la validación y el registro con XML TARGET. La validación se puede realizar con consultas XPATH, o contra un esquema o archivo de definición de tipo de documento (DTD). Los firewalls XML comerciales también se pueden considerar, aunque normalmente son dispositivos de red y es probable que sean excesivos si solo está buscando la protección de inyección SQL. (Clarke, 2012, p. 563)

**Deshabilitar información del lenguaje de descripción de servicios web (WSDL)**

Los servicios web a menudo son tan vulnerables a la inyección de SQL como las aplicaciones web. Para encontrar vulnerabilidades en los servicios web, los atacantes deben saber cómo comunicarse con el servicio web, a saber, los protocolos de comunicación compatibles (por ejemplo, SOAP, HTTP GET, etc.), los nombres de los métodos y los parámetros esperados. Toda esta información se puede extraer del archivo de Lenguaje de descripción de servicios web (WSDL) del servicio web. Por lo general, se puede invocar agregando un WSDL al final de la URL del servicio web. Siempre que sea posible, es una buena idea suprimir esta información de intrusos no deseados. (Clarke, 2012, p. 582)

El siguiente código muestra cómo configurar un servicio web .NET para que no muestre el WSDL. Puede aplicar este cambio de configuración al archivo web.config o machine.config de la aplicación.

```
<webServices>
  <protocols>
    <remove name="Documentation"/>
  </protocols>
</webServices>
```

Apache Axis, una plataforma de servicios web SOAP (Simple Object Access Protocol) de uso común para aplicaciones Java, admite la configuración personalizada del archivo WSDL, que se puede usar para suprimir la generación automática. Se puede configurar los

ajustes de wsdlFile en el archivo .wsdd del servicio para que apunte a un archivo que devuelva una etiqueta <wsdl /> vacía.

En general, no se recomienda el acceso remoto a la información WSDL en los servidores web orientados a Internet. Se puede usar un canal de comunicación seguro alternativo, como el correo electrónico cifrado, para proporcionar este archivo a los socios de confianza que pueden necesitar esta información para comunicarse con el servicio web. (Clarke, 2012, p. 583)

**Técnica de ataque XDoS**

Las técnicas de ataque XDoS a nivel de los servicios Web se aprovechan, en la mayoría de los casos, del procesamiento costoso que puede requerir ciertos tipos de peticiones de usuario. Una revisión de las diferentes técnicas de ataques XDoS contra los servicios web fue llevado a cabo por (Moradian y Håkansson, 2006). La revisión incluye mecanismos de ataques que afectan la disponibilidad de los servicios Web. En la siguiente tabla se presenta la inyección SQL como un tipo de estos ataques XDoS evaluado dentro del trabajo previo de Moradian y Håkansson (Moradian y Håkansson, 2006).

| | **Tipo de ataque** | **Descripción** | **Componente Objetivo** | **Nivel de daño** |
|---|---|---|---|---|
| **Evaluando estructura y contenido de los mensajes SOAP** | SQL Injection | Un atacante modifica la consulta SQL dentro del mensaje SOAP, adicionando comandos SQL de forma arbitraria para ejecutarlos en la base de datos. | Base de Datos | Alto |

Tabla 2. Técnica de ataque XDoS dentro de los servicios Web (Pinzón, 2010, p. 68)

**Agentes con capacidad de Análisis**

La carga útil de las peticiones de los usuarios enviadas a las aplicaciones y los servicios vienen codificadas, por ejemplo, en el estándar XML como es el caso de las solicitudes enviadas a los servicios Web, o una cadena SQL en el caso de las consultas a las base de datos. La detección de intrusiones en este tipo de peticiones requiere examinar además de la información de las cabeceras de las peticiones, el propio contenido de la petición. Para examinar el contenido y la estructura sintáctica generalmente se utiliza algún tipo de analizador sintáctico/semántico. Un analizador sintáctico convierte el texto de entrada en otras estructuras (comúnmente árboles), que son útiles para el posterior análisis. Para llevar

a cabo esta tarea se ha diseñado un tipo de agente analizador, diseñado para emplear las herramientas adecuadas dependiendo de la carga útil a procesar. Este tipo de agente incorpora el analizador SAX (Simple API for XML) (Brownell, 2002, p. 1) para el análisis de contenido XML. SAX fue la primera API para XML ampliamente adoptada en Java y más tarde implementada en muchos otros entornos de programación. (Pinzón, 2010, p. 104)

A diferencia de otros analizadores, SAX funciona por eventos y métodos asociados, haciéndolo más eficiente desde el punto de vista computacional. A medida que el analizador va leyendo el documento XML y encuentra (los eventos) los componentes del documento (elementos, atributos, valores, etc.) o detecta errores, va invocando a las funciones que ha sido configuradas. En el caso de las peticiones a las bases de datos basadas en cadenas SQL, se implementa un analizador construido utilizando las herramientas JFlex y Cup. Ambas herramientas están desarrolladas en Java y se caracterizan por facilitar el análisis léxico y la escritura de la gramática. Debido a los distintos estándares SQL disponibles, se ha adoptado el gestor de base de datos MySQL 5.0–SQL Sin embargo, se puede extender fácilmente la funcionalidad del analizador a otros gestores de bases de datos. Para este tipo de agente se ha utilizado el modelo BDI (creencias, deseos e intenciones), considerando la posibilidad de extender sus funcionalidades, como es el caso de la detección de errores. Sin embargo, de momento sólo se ha desarrollado sus funciones básicas. (Pinzón, 2010, p. 105)

### 3.2 Metodología de desarrollo de Software

Para crear la metodología de desarrollo de éste proyecto se usó el método OpenUP; éste se define como un proceso unificado ligero que aplica enfoques iterativos e incrementales dentro de un ciclo de vida estructurado. OpenUP adopta una filosofía pragmática y ágil que se centra en la naturaleza colaborativa del desarrollo de software. Es un proceso agnóstico de herramientas y de baja ceremonia que se puede extender para abordar una amplia variedad de tipos de proyectos. (OpenUP, 2007, p. 2)

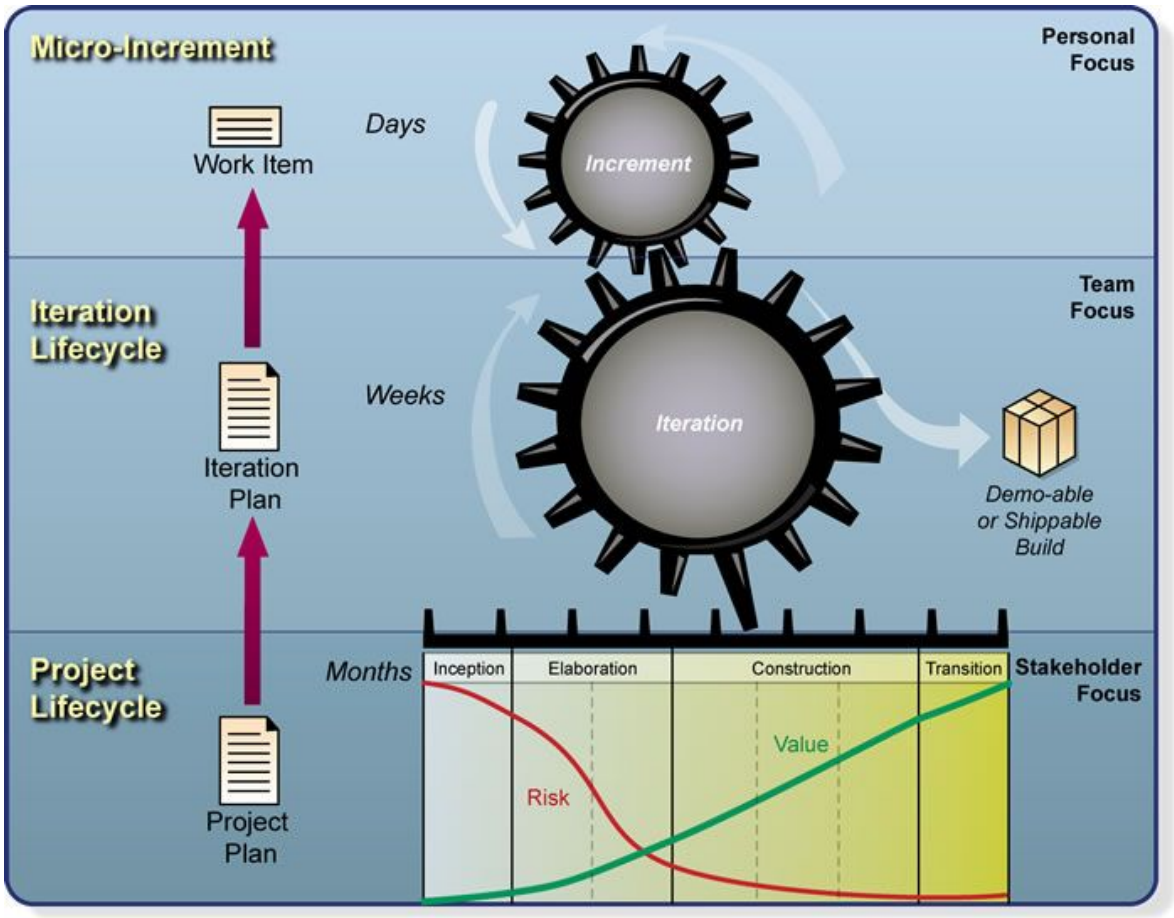



Figura 18. Microincrementos, ciclo de vida de iteración y del proyecto (OpenUP, 2007)

**JUnit**

Para las pruebas a la aplicación desarrollada se utilizó JUnit, el cual es un marco de código abierto diseñado con el propósito de escribir y ejecutar pruebas en el lenguaje de programación Java. JUnit, originalmente escrito por Erich Gamma y Kent Beck, ha sido importante en la evolución del desarrollo basado en pruebas, que forma parte de un paradigma de diseño de software más amplio conocido como Extreme Programming(XP). (JUnit, 2019, p. 5)

Figura 19. JUnit para ejecutar pruebas en Java (JUnit, 2019, p. 5)

# 4. MODELO DE ARQUITECTURA DEL MODELO BASE DE CONOCIMIENTO

## 4.1 Descripción General

Modelo de Arquitectura del proyecto "Base de conocimiento para Auditorías de WS con SQL Injection".

## 4.2 Términos y condiciones de uso

Este documento es de carácter confidencial aplicable para la construcción del proyecto "Base de conocimiento en Auditorías de WS con SQL Injection. Se realiza con carácter educativo y formativo en el marco del espacio académico de tesis de grado de la Maestría de Ciencias de Información y las Comunicaciones de la Universidad Distrital.

## 4.3 Descripción del Modelo

Este documento proporciona los elementos necesarios de diseño para la implementación del Proyecto "Base de conocimiento en Auditorías de WS con SQL Injection" que tiene como finalidad la implementación de un sistema encargado de almacenar lógica y estructuralmente las auditorías que se realizarán en los servicios web con el uso de SQL Injection.

Este documento sirve como medio de comunicación entre el arquitecto de software y otros miembros del equipo respecto a las decisiones significativas para la arquitectura que se llevan a cabo en el proyecto; se debe tener en cuenta que el presente documento es susceptible de futuras modificaciones o ampliaciones.

## 4.4 Alcance del Modelo

De acuerdo al documento de especificación detallada del requerimiento de la "Base de conocimiento en Auditorías de WS con SQL Injection", se busca un sistema que permita "almacenar y organizar" los documentos y/o artefactos resultados de las auditorías realizadas en los sistemas de web services de acuerdo a los prototipos propuestos.

## 4.5 Representación Arquitectónica del Modelo

La arquitectura de la solución se desarrollará bajo la arquitectura cliente/servidor del lado de la base de conocimiento y de componentes independientes del lado de la auditoría de los web services.

### 4.5.1 Arquitectura Base de Conocimiento en OpenKM: cliente/servidor

| | |
|---|---|
| Servidor | Apache Tomcat: https://ritaportal.udistrital.edu.co:10229 |
| Cliente | Cualquier cliente web |

### 4.5.2 Arquitectura aplicación con Web Services: estilo de capas - subestilo cliente-servidor

| | |
|---|---|
| Cliente | Glassfish: https://ipservidor.:8080/Factura Electrónica/test-resbeans.html |
| Servidor | Cualquier cliente dentro del segmento de red privado |

## 4.6 Arquitectura del modelo base de conocimiento para auditoría de Web Service

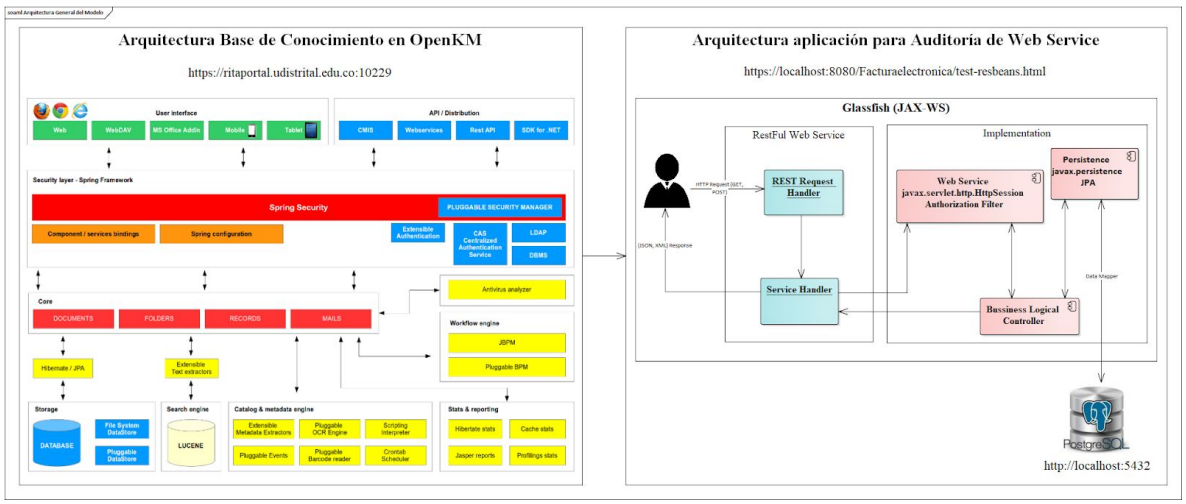



Figura 20. Arquitectura del modelo general (www.openkm.com/en/architecture.html)

## 4.7 Diagrama de Componentes

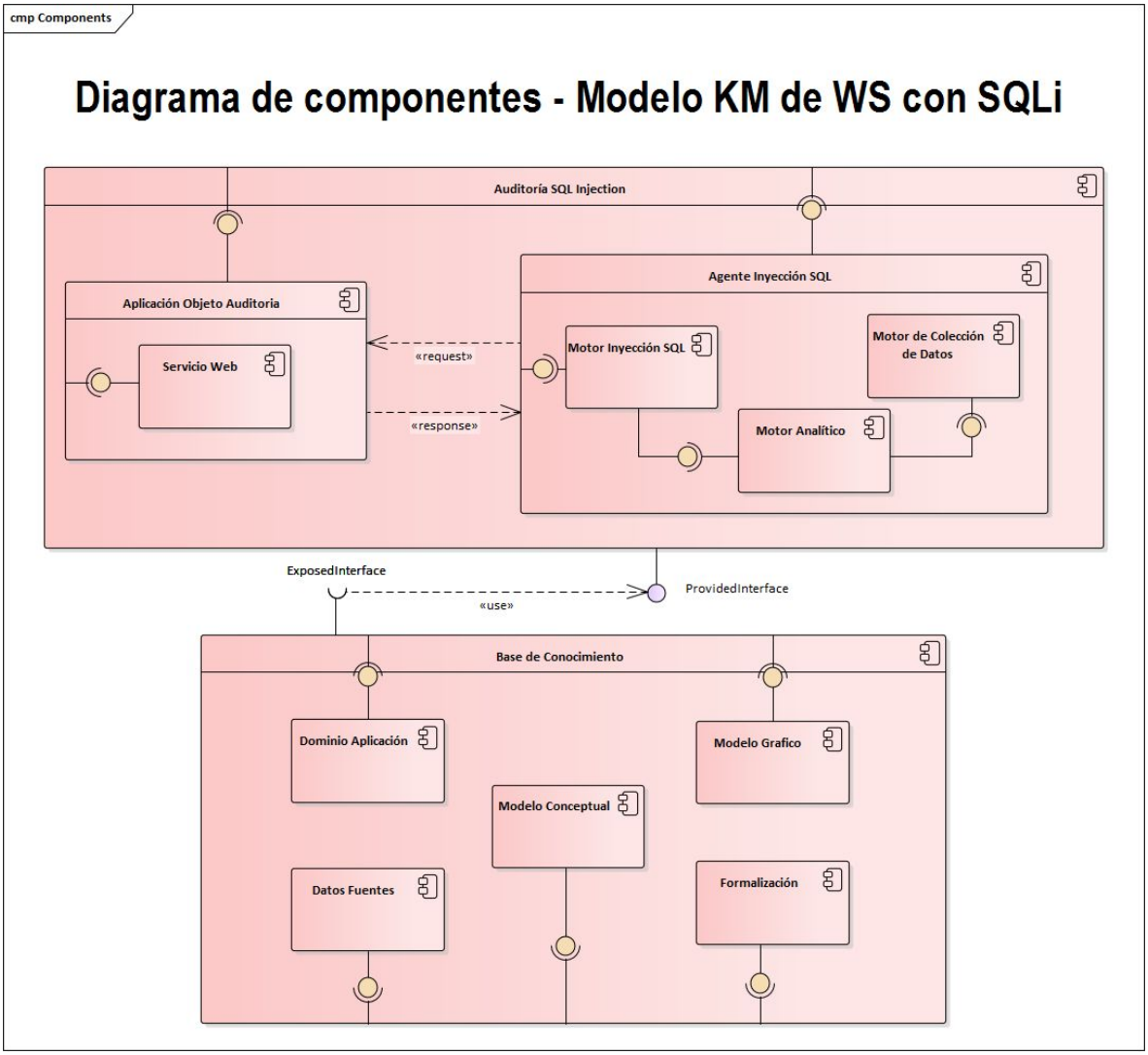



Figura 21. Diagrama de componentes Modelo Base de Conocimiento para la Auditoría

# 5. IMPLEMENTACIÓN DEL ESTILO ARQUITECTÓNICO EN EL PROTOTIPO

Para implementar el prototipo se utilizó el estilo arquitectónico por capas ejemplificado en un modelo vista-controlador (MVC); estos sistemas constituyen uno de los estilos que aparecen con mayor frecuencia mencionados como categorías mayores del catálogo, o, por el contrario, como una de las posibles encarnaciones de algún estilo más envolvente. En Garlan y Shaw definen el estilo en capas como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Instrumentan así una vieja idea de organización estratigráfica que se remonta a las concepciones formuladas por el patriarca Edsger Dijkstra en la década de 1960, largamente explotada en los años subsiguientes.

Se definió el modelo de datos e identificadores que representan los recursos a exponer. La construcción del WS se limitó a la implementación de la tecnología de servicio web RESTful (REpresentational State Transfer, based on REST architecture), con dos métodos HTTP: GET y POST para especificar las acciones de consulta y registro de información respectivamente. Cabe aclarar que se estableció así de acuerdo al masivo uso que se tiene en las entidades y organizaciones del estado colombiano (probablemente por su potencial escalable, así como el acceso con escaso consumo de recursos); sin embargo si se prefiere implementar SOAP por su robustez, la seguridad, el control y validación que ofrecen los esquemas debido a su alto acoplamiento, se podría usar por ejemplo en integraciones de cores bancarios o pasarelas de pago. Para las fases de inicio, elaboración, construcción y transición de la aplicación y del servicio web de la aplicación objeto de auditoría se implementó el método OpenUP en cuatro iteraciones, una por cada fase.

## 5.1 Fase: Inicio

En primera instancia se realizaron reuniones de trabajo entre el director del proyecto y el estudiante, en las que se explicó la visión de la aplicación, la metodología, y el entorno en el que se desenvolverá el proyecto y finalmente se seleccionó la tecnología y técnica a utilizar en el desarrollo del prototipo de aplicativo del proyecto.

### 5.1.1 Iteración 1 - Caracterización previa de la aplicación objeto de auditoría

Prototipo de aplicación basada en un software para gestionar las facturas electrónicas de la DIAN a través de una aplicación web llamada “Factura Electrónica” la cual se enmarca en un modelo de información de facturas tradicional, que sirve como soporte de costos, deducciones y descontables de las compras y adquisiciones realizadas. Al ser digital, se convierte en un medio eficiente para el registro de las compras en los sistemas contables, a

través de procesos automáticos o semiautomáticos ejemplificados en los módulos de gestión de factura electrónica, asociación de proveedores o software, administración de software, proveedores tecnológicos, facturadores electrónicos, asignaciones, clientes y pagos. Se entrega los siguientes productos:

1. Modelos de arquitectura, y diagramas de componentes.
2. Aplicación Factura Electrónica, código fuente y scripts de base de datos.

Para el desarrollo de la aplicación se definen los siguientes aspectos tecnológicos para su implementación.

- Estilo arquitectónico de la aplicación es por capas (Presentación, negocio y datos).
- La arquitectura de la aplicación de tipo cliente servidor, con formularios en Java.

Así mismo de definen la utilización de las siguientes librerías por cada capa:

**Capa Datos**

- JPA Versión 2.1
- Paquete javax.persistence

**Capa Negocio**

- EJB Versión 1.1
- Paquetes import java.beans y import javax.ejb

**Capa Presentación**

- Framework JavaServer Faces.
- Componente Primefaces 5.0
- Paquetes import javax.faces y import org.primefaces

**Ambiente de Desarrollo**

- Producto: NetBeans IDE 8.2
- Java: 1.8.0_191; Java HotSpot(TM) 64-Bit Server VM 25.191-b12
- Runtime: Java(TM) SE Runtime Environment 1.8.0_191-b12
- System: Windows 10 versión 10.0 running on amd64; Cp1252; es_ES

**Servidor**

- Server Name: GlassFish Server 4.1.1
- Versión: 4.1.1

**Base de Datos:**

- Producto: PostgreSQL
- Versión: 11.1

**Navegación:**

Dadas las diferencias de funcionalidades que debe proveer el sistema para cada actor, se decidió dividir el mapa de acuerdo con el rol del usuario que ingresó a la aplicación:

Facturador electrónico: tendrá acceso a ver y modificar toda la información relacionada con el sistema también tendrá acceso a crear, editar y asignar las facturas recibidas a un Proveedor tecnológico para que ejecute las acciones respectivas.
Proveedor Tecnológico: podrá recibir y modifica el estado de las asignaciones.
Cliente: podrá ingresar al sistema para consultar el estado de sus facturas y ver las tareas ofrecidas.

Cada mapa de navegación permite determinar el módulo web a desarrollar a través de las páginas web que contiene y de las relaciones entre ellas.

## 5.2 Fase: Elaboración

### 5.2.1 Iteración 2 - Implementación del aplicativo Factura Electrónica como prototipo

Tomando en cuenta los requisitos funcionales y no funcionales se implementó la aplicación, la cual soporta la solución de Factura Electrónica Web, a continuación, se muestra el inicio de sesión general de dicho aplicativo con un tipo de usuario:

Figura 22. Inicio de sesión

La interfaz del sistema se basó en la navegación mencionada previamente, con un inicio y una ventana principal compuesta por un menú general y una serie de opciones de menús para cada una de los módulos de gestión del sistema, como se ve en la figura a continuación:

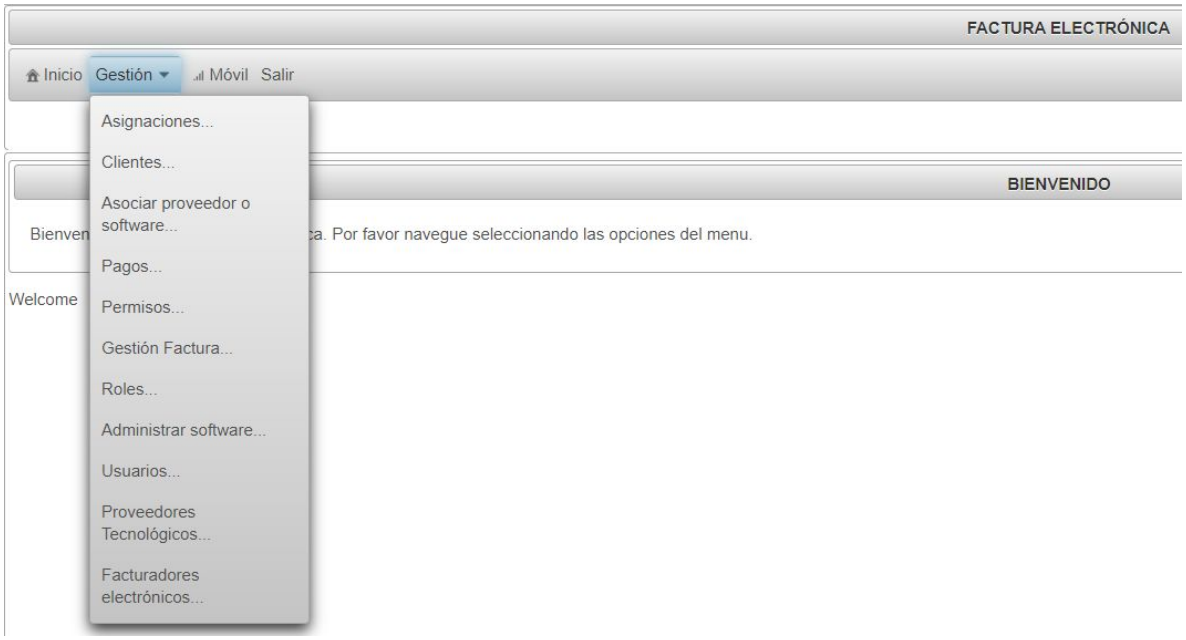



Figura 23. Menú general con los módulos de gestión del aplicativo

Si entramos a cada módulo podremos ver las diferentes opciones que podremos elegir, como crear, ver, editar o eliminar correspondiente al caso de uso que se esté trabajando:

Figura 24. Ingreso a módulo de gestión de facturas electrónicas vía web

Una vez realizado el desarrollo, se realizaron pruebas unitarias con la herramienta JUnit de las clases relevantes en el modelo, involucradas en los casos de uso descritos anteriormente:

```
Asignaciones.java | AsignacionesTest.java | AuthorizationFilter.java | AuthorizationFilterTest.java
Source | History

248         System.out.println("equals");
249         Object object = null;
250         Asignaciones instance = new Asignaciones();
251         boolean expResult = false;
252         boolean result = instance.equals(object);
253         assertEquals(expResult, result);
254         // TODO review the generated test code and remove the default call to fail.
255         if (result!=expResult){
256             fail("The test case is a prototype.");
257         }
258     }
259
260     /**
261      * Test of toString method, of class Asignaciones.
262      */
263     @Test
264     public void testToString() {
265         System.out.println("toString");
266         Asignaciones instance = new Asignaciones();
267         String expResult = "entity.Asignaciones[ idAsignacion=null ]";
268         String result = instance.toString();
269         assertEquals(expResult, result);
270         // TODO review the generated test code and remove the default call to fail.
            if (result == null ? expResult != null : !result.equals(expResult)){
272             fail("The test case is a prototype.");
273         }
274     }

Output | HTTP Server Monitor | Test Results
entity.AsignacionesTest | filter.AuthorizationFilterTest

Tests passed: 100.00 %
All 15 tests passed. (0.128 s)
entity.AsignacionesTest passed
  testGetIdReparacion passed (0.002 s)
  testGetAsignadoPor passed (0.001 s)
  testToString passed (0.001 s)
  testSetAsignadoA passed (0.001 s)
  testSetIdReparacion passed (0.001 s)
  testGetIdAsignacion passed (0.001 s)
  testSetFechaAsignado passed (0.0 s)
  testSetFechaRecibo passed (0.0 s)
  testSetIdAsignacion passed (0.0 s)
  testHashCode passed (0.0 s)
  testEquals passed (0.0 s)
  testSetAsignadoPor passed (0.0 s)
  testGetFechaRecibo passed (0.0 s)
  testGetFechaAsignado passed (0.001 s)
  testGetAsignadoA passed (0.0 s)

getIdReparacion
getAsignadoPor
toString
setAsignadoA
setIdReparacion
getIdAsignacion
setFechaAsignado
setFechaRecibo
setIdAsignacion
hashCode
equals
setAsignadoPor
getFechaRecibo
getFechaAsignado
getAsignadoA
```

Figura 25. Pruebas con JUnit para Clase Asignaciones

El aplicativo también tiene la opción “mobile” para entrar a los módulos vía web, emulando como se ingresaría desde un dispositivo móvil.

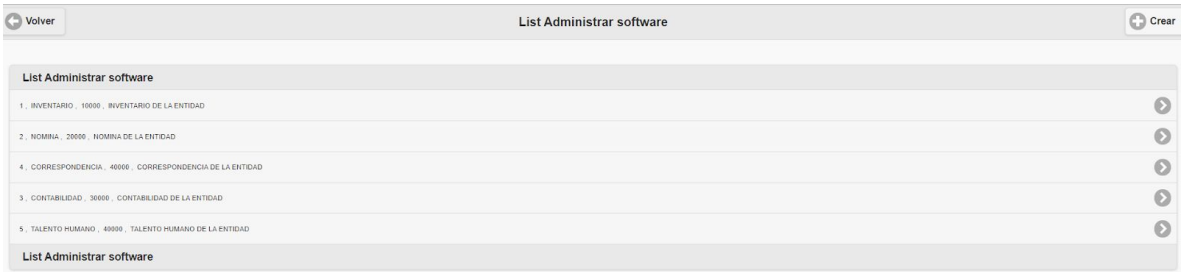



Figura 26. Ingreso a módulo de gestión de software vía móvil

## 5.3 Fase: Implementación

### 5.3.1 Iteración 3 - Implementación del Web Service a través del IDE Netbeans

Considerando los requisitos funcionales, se establecen los comportamientos deseados del aplicativo Factura Electrónica Web, los cuales fueron traducidos en especificaciones técnicas utilizando el lenguaje de descripción de aplicaciones web (WADL) utilizado por la herramienta Netbeans que permitió desarrollar la construcción del servicio web RESTFul; el cual nos permite escoger el método (GET utilizado únicamente para consultar información al servidor o POST utilizado para solicitar la creación de un nuevo registro) y el formato (XML un lenguaje de marcado extensible o JSON un formato ligero de intercambio de datos) para la consulta o ingreso de datos de la aplicación, como se ve en la siguiente figura:

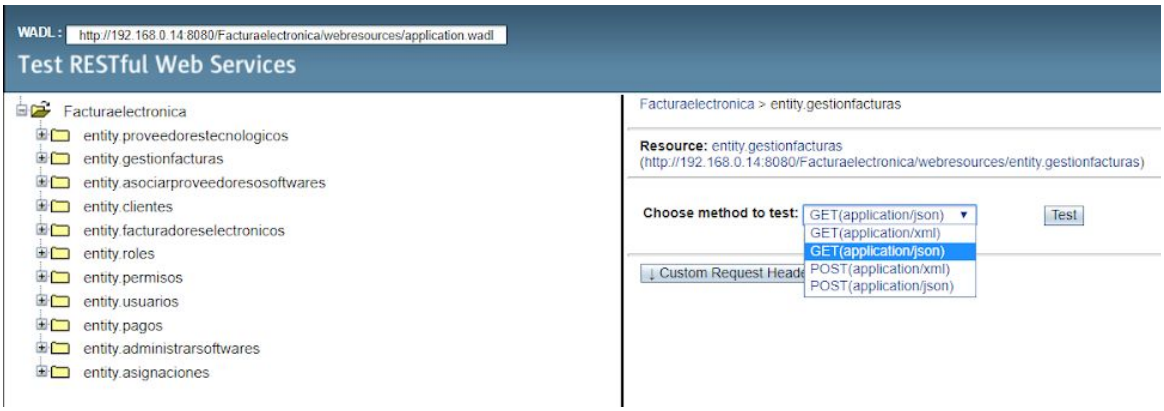



Figura 27. Interfaz gráfica del Web Service RESTFul con Netbeans.

Una vez seleccionado el método GET y el formato de salida, se hace la prueba, dando clic en el botón "Test", el cual nos arrojará los datos de la aplicación Factura Electrónica Web UD.

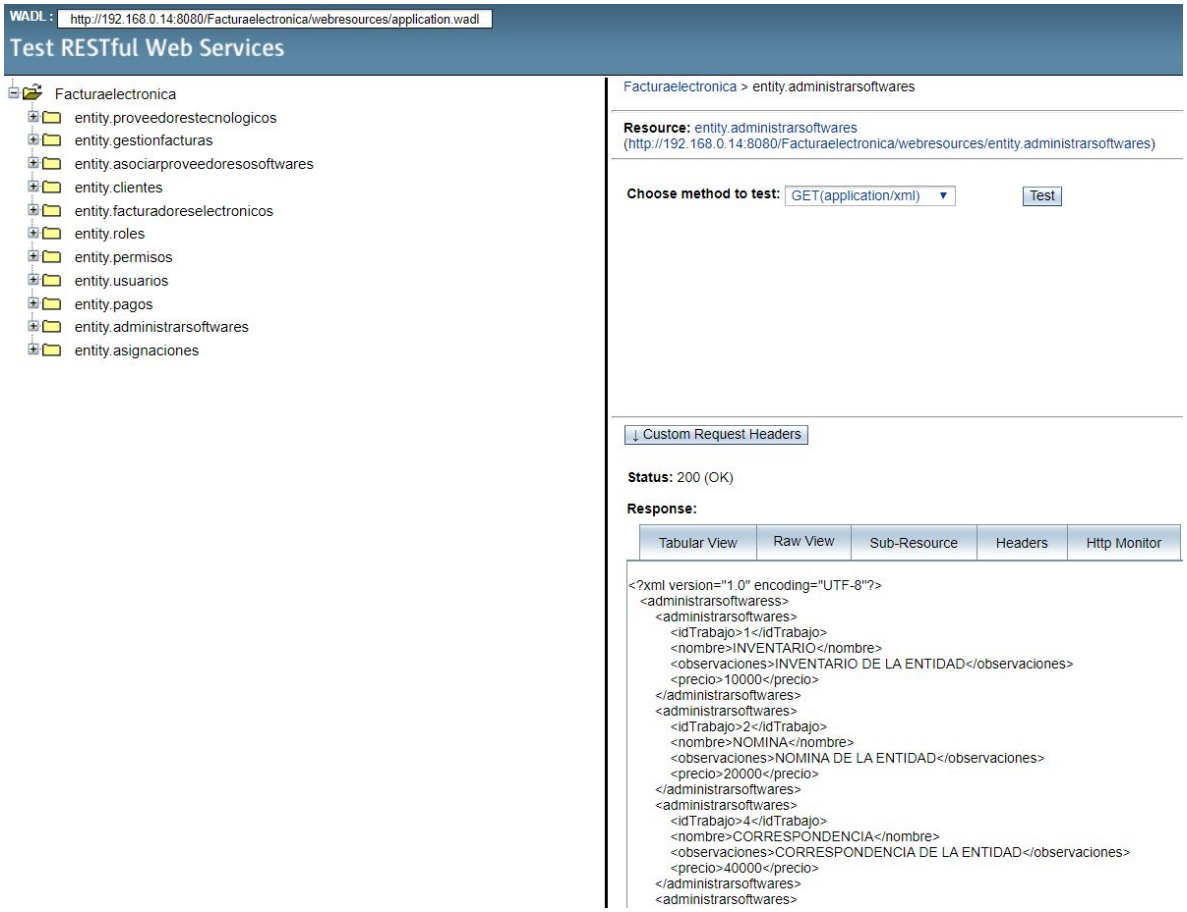



Figura 28. Consulta de datos consumiendo el Web Service RESTFul como cliente con GET.

Ahora vamos a probar el método POST, seleccionamos el formato preferido, agregamos el contenido y damos clic en "Test":

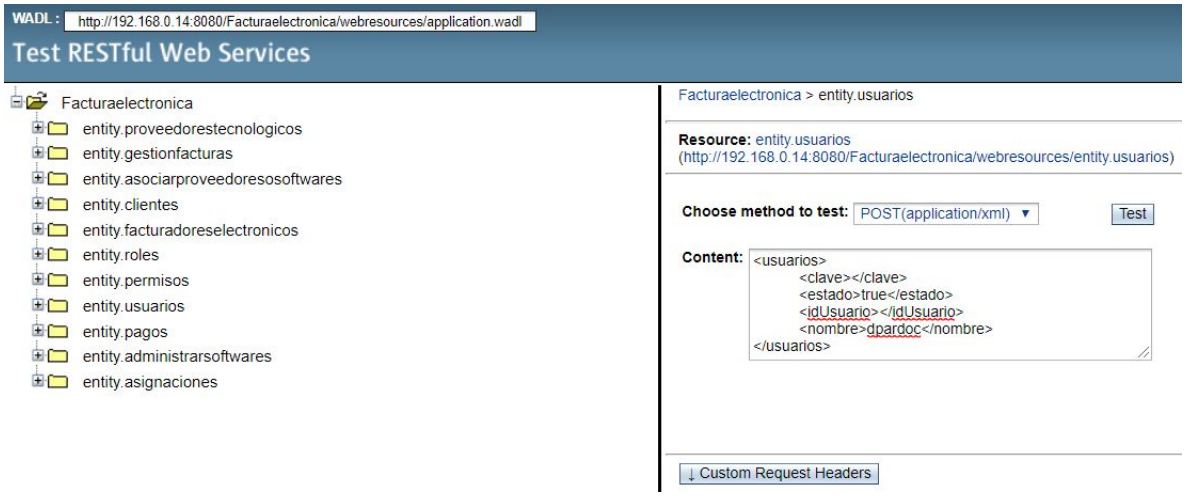



Figura 29. Ingreso de datos usando un Web Service RESTFul como cliente.

Cómo resultado aparece un apartado de respuesta, no devuelve ningún contenido, sin embargo, posteriormente verificamos el ingreso de datos:

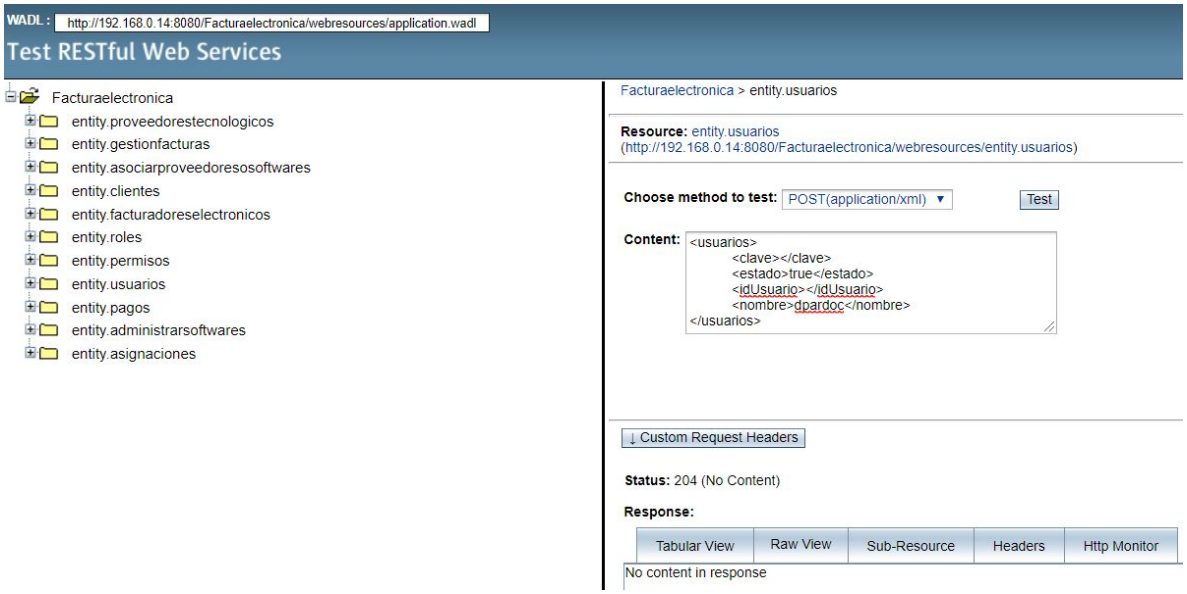



Figura 30. Respuesta de ingreso de datos usando un Web Service RESTFul como cliente.

Accedemos a la aplicación, vamos al módulo *Usuarios* y verificamos el registro ingresado:

Figura 31. Prueba de ingreso de información en el sistema por medio del WS.

## 5.4 Fase: Transición

### 5.4.1 Iteración 4 - Integración del aplicativo y el WS a través del módulo de seguridad

Finalmente, para realizar el despliegue en el ambiente de desarrollo y la integración a del aplicativo con el Web Service, se hizo a través de un módulo de seguridad de la solución, el cual consiste en la implementación de una librería llamada AuthorizationFilter la cual se expone a continuación, debido al impacto que tiene en el modelo de negocio del proyecto:

```
package com.journaldev.jsf.filter;
import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebFilter(filterName = "AuthFilter", urlPatterns = { "*.xhtml" })
public class AuthorizationFilter implements Filter {
	public AuthorizationFilter() {
	}

	@Override
```

```
        public void init(FilterConfig filterConfig) throws ServletException {
        }

        @Override
        public void doFilter(ServletRequest request, ServletResponse response,
                        FilterChain chain) throws IOException, ServletException {
                try {   HttpServletRequest reqt = (HttpServletRequest) request;
                        HttpServletResponse resp = (HttpServletResponse) response;
                        HttpSession ses = reqt.getSession(false);
                        String reqURI = reqt.getRequestURI();
                        if (reqURI.indexOf("/login.xhtml") >= 0
                                        || (ses != null && ses.getAttribute("nombreusu") != null)
                                        || reqURI.indexOf("/public/") >= 0
                                        || reqURI.contains("javax.faces.resource"))
                                chain.doFilter(request, response);
                        else
                                resp.sendRedirect(reqt.getContextPath() + "/faces/login.xhtml");
                } catch (Exception e) {
                        System.out.println(e.getMessage());
                }
        }
        @Override
        public void destroy() {
        }
}
```

Lo anterior permite asegurar que primero debe iniciar sesión como un usuario del aplicativo para hacer uso del Web Service, tanto vía web como móvil.

Figura 32. Ingreso al Web Service por medio del módulo de seguridad del aplicativo.

Una vez iniciada la sesión, se puede hacer consumo de los métodos expuestos en el servicio web, mediante el link http://ipservidor:8080/Factura Electrónica/test-resbeans.html:

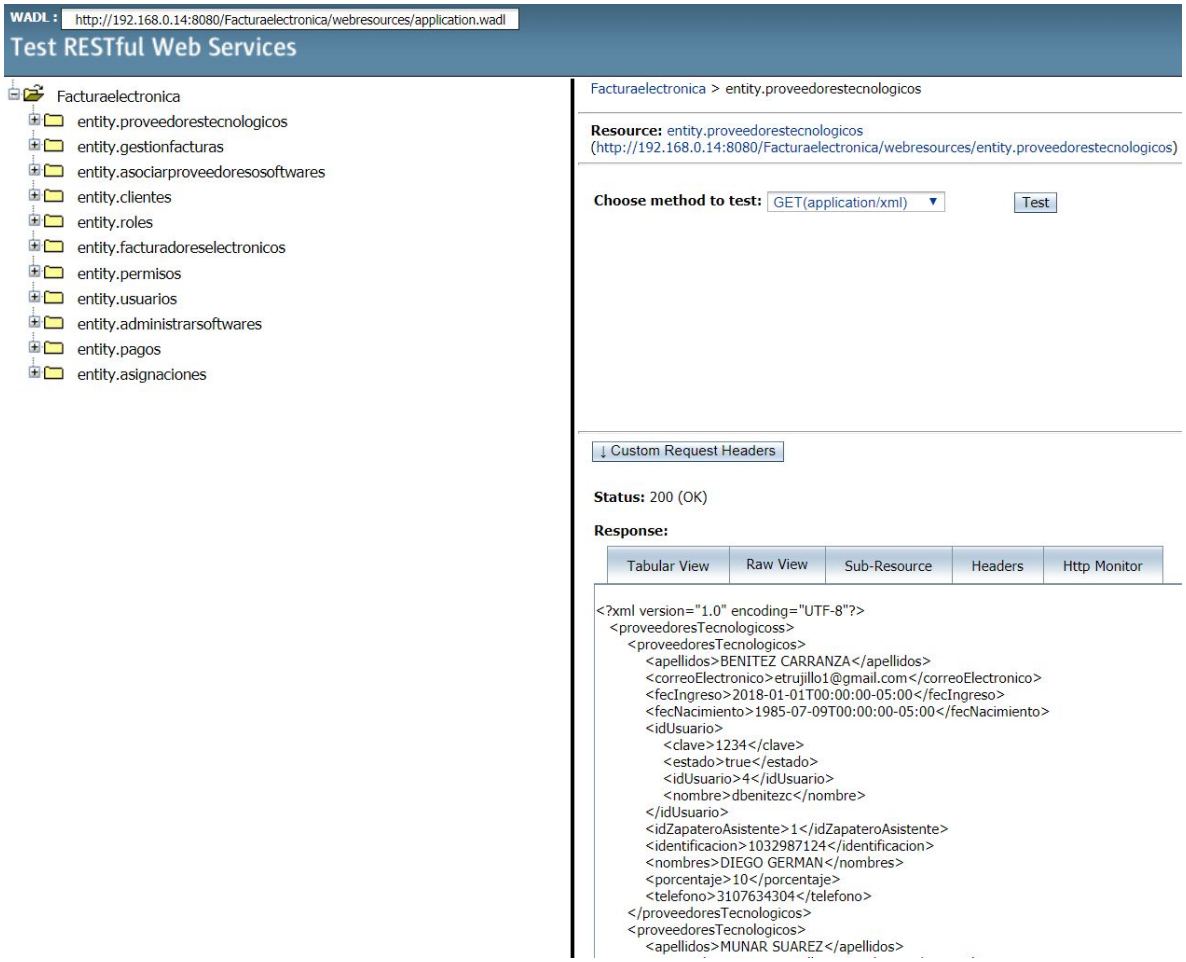



Figura 33. Interfaz gráfica del servicio web de la aplicación Factura Electrónica usando RESTful.

Se realizó pruebas unitarias con la herramienta JUnit a la clase encargada de gestionar la seguridad para el aplicativo y el Web Service:

```
75        }
76
77        /**
78         * Test of destroy method, of class AuthorizationFilter.
79         */
80        @Test
81        public void testDestroy() {
82            System.out.println("destroy");
83            AuthorizationFilter instance = new AuthorizationFilter();
84            instance.destroy();
85            // TODO review the generated test code and remove the default call to fail.
86            if (instance!=instance){
87                fail("The test case is a prototype.");
88            }
89        }
90
91    }
92
```

```
Tests passed: 100.00 %
All 3 tests passed. (0.057 s)
filter.AuthorizationFilterTest passed
    testInit passed (0.002 s)
    testDoFilter passed (0.001 s)
    testDestroy passed (0.001 s)
```

```
init
doFilter
null
destroy
```

Figura 34. Pruebas con JUnit para Clase AuthorizationFilter

# 6. PROPUESTA TÉCNICA DE LA AUDITORÍA EN EL MODELO

## 6.1 Escenario para la auditoría de servicios web con SQL Injection

Como se muestra en la imagen a continuación, se virtualizó un Kali Linux (distribución del sistema operativo Linux para la auditoría y seguridad informática en general), para ejecutar la propuesta de la auditoría a la aplicación Factura Electrónica web y específicamente a su servicio web expuesto.

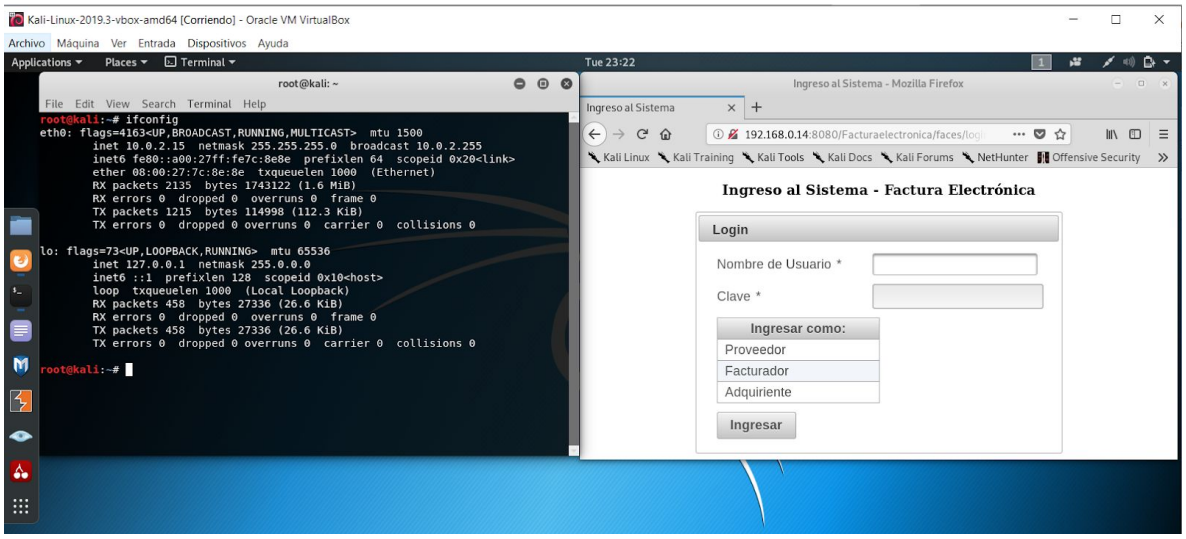



Figura 35. Virtualización de Kali Linux para realizar las pruebas del modelo con SQLi

## 6.2 Inyección de SQL en la aplicación y servicio web propuestos

El objetivo en esta sección consiste en aplicar lo revisado en la sección de SQL Injection a un dominio de información específico (Factura Electrónica) para evitar ataques de SQL en servicios web mediante una auditoría.

El escenario del caso estudio es: Implementación de un prototipo desarrollado con JSF usando Web Service, con un servidor y un cliente.

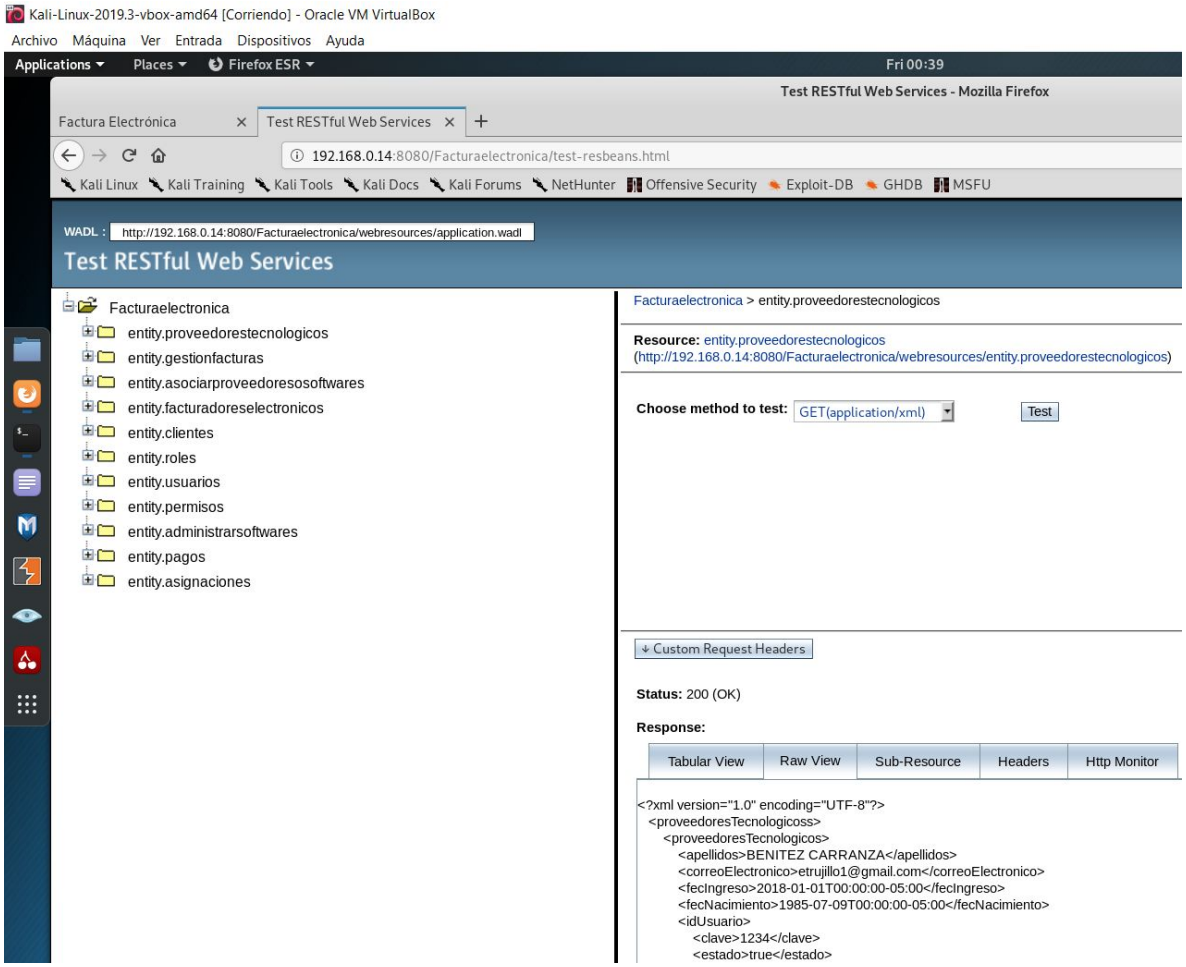



Figura 36. Acceso desde Kali al servicio web registrado para pruebas de auditoría con SQLi

# 7. EJECUCIÓN DE LA AUDITORÍA EN EL MODELO

## 7.1 JERARQUÍA DEL MODELO BASE DE CONOCIMIENTO PARA AUDITORÍAS DE SERVICIOS WEB

En éste apartado se encuentra la justificación de cada uno de los segmentos de la jerarquía del modelo de base de conocimiento para la auditoría en servicios web para evitar los ataques de tipo SQL Injection como vector de ataque del proyecto y que servirá para que el usuario tenga una orientación de la taxonomía manejada en este modelo. La taxonomía del modelo se desarrolló extrayendo ideas propias y de varias investigaciones previas las cuales se irán referenciando a medida que va avanzando el documento, a continuación, se presenta cada uno de los segmentos. (Rodríguez, 2015, p. 10) (Contreras, 2017, p. 11). En cada subsección se mostrará al lado un ítem de obligatoriedad u opcionalidad de acuerdo al modelo que se está proponiendo, por ejemplo: **Escaneo de Puertos y versiones** ***Obligatorio***, **Análisis SSL** ***Opcional***. En caso tal de que el usuario no desee desarrollar los ítems propuestos en el modelo, éste modelo no se hará responsable de los resultados finales en cuestión de la base de conocimiento y de la auditoría de los servicios web. Las secciones principales son ***Obligatorias*** en éste modelo en base a que se manejan en general en las auditorías de servicios de aplicaciones web.

### 7.1.1 ACERCA DE LA AUDITORÍA

En esta sección introductoria de vital importancia se realiza un primer acercamiento al negocio de la auditoría de aplicaciones web ya que se debe conocer los inicios del proyecto que abordará el usuario. Esta sección se desarrollará de acuerdo al modelo del negocio definido por el usuario. Esta sección se hace con el fin de orientar al usuario y ubicarlo en una primera instancia del modelo.

**Contexto** ***Obligatorio***

En esta subsección se realiza una descripción y contextualización del negocio al cual el usuario requiere hacerla la auditoría. Esta subsección se desarrollará con los modelos y diagramas que se obtengan del negocio definido por el usuario tales como el *modelo de arquitectura*, *diagrama de componentes* y/o *diagrama de clases*. Esta subsección se hace con el fin de que el usuario tenga disponibles sus modelos y diagramas iniciales del negocio objeto de auditoría en la base de conocimiento.

**Justificación** ***Obligatorio***

En esta subsección se redactan los motivos de la importancia de realizar la auditoría en un modelo de negocio predefinido por el usuario. Esta subsección se desarrollará con las

necesidades propias del negocio por el cual se quiere hacer la auditoría. Esta subsección se hace con el fin de que el usuario tenga siempre disponible las necesidades y motivos del negocio objeto de auditoría en la base de conocimiento.

**Objetivos *Obligatorio***
En esta subsección se redactan los objetivos generales y específicos a alcanzar por el usuario en la auditoría. Esta subsección se desarrollará con los objetivos propios del usuario en el contexto propio del modelo de negocio. Esta subsección se hace con el fin de que el usuario tenga siempre disponible los objetivos general y específicos del negocio objeto de auditoría en la base de conocimiento.

**Productos entregables *Obligatorio***

En esta subsección se describen los productos a entregar previstos por el usuario una vez realizada la auditoría. Esta subsección se desarrollará con los productos previstos propios del modelo de negocio que el usuario tiene que entregar. Esta subsección se hace con el fin de que el usuario tenga siempre disponible la lista de productos a entregar de acuerdo a la auditoría realizada.

**Recursos necesarios *Obligatorio***

En esta subsección se describen los recursos mínimos preliminares y necesarios que debe tener el proyecto y el modelo de negocio. Esta subsección se desarrollará con investigaciones previas que se han realizado a nivel de infraestructura (hardware y software) y de las herramientas necesarias en el proyecto. Esta subsección se hace con el fin de que el usuario tenga siempre disponible el listado de recursos mínimos necesarios para lograr los objetivos del proyecto en cuestión para realizar la auditoría

**Descripción de metodologías *Obligatorio***

En esta subsección se describen las metodologías y buenas prácticas utilizadas de acuerdo al proyecto y al modelo de negocio. Esta subsección se desarrollará con investigaciones previas que se han realizado sobre las metodologías acordes al proyecto y definidas en el anteproyecto, con aplicación en contexto nacional e internacional. Esta subsección se hace con el fin de que el usuario tenga siempre disponible las metodologías definidas para la elaboración del proyecto en cuestión y para realizar la auditoría.

**Planificación *Opcional***

En esta subsección se propone como opcional debido a que dependiendo de las metodologías implementadas por el proyecto y también la dimensión del modelo de negocio, el concepto de tiempo en la planificación entraría como temporal. Esta subsección se desarrollará con la herramienta Project de Microsoft (licenciada por la entidad en la que

laboro actualmente) con el proyecto que se está realizando en cuestión. Esta subsección se hace con el fin de que el usuario tenga siempre disponible un cronograma planificado con las iteraciones, fases, actividades y tiempos del proyecto actual, para realizar la auditoría**.**

### 7.1.2 ACTUAL SITUACIÓN Y METODOLOGÍA

En esta sección de referencia del modelo se realiza un estado del arte de la seguridad de las aplicaciones web y de las metodologías utilizadas en la misma ya que es necesario conocer el punto de partida del proyecto. Esta sección se desarrollará de acuerdo al estado del arte de seguridad de las aplicaciones web, las metodologías y las buenas prácticas actuales en este campo. Esta sección se hace con el fin de hacer que el usuario esté inmerso en éste tema de seguridad de aplicaciones web y para ver cuáles metodologías y buenas prácticas puede aplicar de acuerdo a su modelo de negocio

**Osstmm** ***Obligatorio***

En esta subsección se expone brevemente, el manual de metodologías de la OSSTMM y los casos de pruebas catalogados por el manual en cuestión, ya que es un elemento importante que se requiere en la auditoría y que ayudará al usuario en la práctica. Esta subsección se desarrollará con los tipos y casos de acuerdo a las últimas versiones propuestas por el OSSTMM. Esta subsección se hace con el fin de que el usuario tenga disponibles los conceptos, tipos y casos de prueba de seguridad en el modelo propuesto.

**Owasp** ***Opcional***

En esta subsección se exponen los tipos de ataques y las vulnerabilidades más populares actualmente, concernientes a la guía de seguridad de pruebas de OWASP, ya que es un referente mundial importante en el tema de la seguridad de aplicaciones y servicios web. Esta subsección se desarrollará de acuerdo al último top ten propuesto por el OWASP. Esta subsección se hace con el fin de que el usuario tenga disponibles los tipos de ataques y vulnerabilidades de seguridad de acuerdo al modelo propuesto.

**MinTIC** ***Opcional***

En esta subsección se exponen los lineamientos, modelos de seguridad y guías metodológicas y de auditorías propuestas actualmente por MinTIC, ya que es el referente nacional en el tema de la seguridad de aplicaciones y servicios web. Esta subsección se desarrollará de acuerdo a las últimas versiones de los documentos mencionados previamente. Esta subsección se hace con el fin de que el usuario tenga disponibles los lineamientos, modelos de seguridad y guías metodológicas y de auditorías de seguridad de acuerdo al modelo propuesto.

### 7.1.3 AUDITORÍA WEB

En esta sección se explican los temas más destacados de auditoría web en un ambiente organizacional.

**Perfil de auditor** ***Opcional***

Esta subsección describe el perfil del auditor informático, encargado de aplicar sus conocimientos para lograr unos resultados de calidad y generar así recomendaciones de valor para la auditoría; se marca como opcional ya que no es un ítem bloqueante para poder proceder con la auditoría. Esta subsección se desarrollará con la información que se tenga de perfiles de auditores en revisiones en auditorías de información. Esta subsección se hace con el fin de que el usuario tenga disponibles las características del perfil de un auditor de seguridad informático de acuerdo al modelo propuesto.

**Tipos de test** ***Obligatorio***
En esta subsección se exponen los tipos (interna o externa) y enfoques de la auditoría, ya que puede ser aplicada a diferentes contextos. Esta subsección se desarrollará con la información de revisiones que se tenga de los diferentes enfoques. Esta subsección se hace con el fin de que el usuario tenga disponibles los enfoques de auditoría de acuerdo al modelo propuesto.

**Aspectos contractuales de auditoría** ***Obligatorio***
En esta subsección se exponen los aspectos contractuales de la auditoría, ya que son los que el auditor firma con la organización en caso de ser aplicada. Esta subsección se desarrollará con la información de revisiones que se tenga de los tipos de permisos, acuerdos, reglas y del alcance al aplicar una auditoría. Esta subsección se hace con el fin de que el usuario tenga disponibles los aspectos más comunes de una auditoría.

**Aspectos normativos legales** ***Obligatorio***
En esta subsección se exponen las normas y leyes que apliquen en la auditoría, ya que si la organización encuentra vulnerabilidades que afecten información personal (Habeas Data), ésta debe blindarse según las leyes expuestas en éste apartado. Esta subsección se desarrollará con la información de revisiones que se tenga de las leyes y normas que apliquen en la auditoría. Esta subsección se hace con el fin de que el usuario tenga disponibles las leyes y normas, ya sean nacionales o internacionales, al realizar la auditoría.

**Ética profesional** ***Obligatorio***
En esta subsección se exponen algunos aspectos éticos del auditor de vital importancia, ya que la reputación y la confianza son temas que se gana el auditor en el desarrollo y resultados de su trabajo. Esta subsección se desarrollará con la información de revisiones que se tenga de asuntos destacables en el tema de la ética profesional en la auditoría. Esta

subsección se hace con el fin de que el usuario tenga disponibles los acuerdos de confidencialidad y de imparcialidad que se tenga al aplicar la auditoría.

**Informe de resultados *Obligatorio***
En esta subsección se expone la estructura y la forma como se elabora el informe de resultados final, ya que éste documento reúne todo el trabajo realizado en la auditoría. Esta subsección se desarrollará con la información de revisiones que se tenga de estructuras y elaboraciones de informes que se tenga en auditorías. Esta subsección se hace con el fin de que el usuario tenga disponible una estructura general del informe de resultados de la auditoría.

### 7.1.4 FASE DE CONOCIMIENTO

En esta sección se hace la recolección de toda la información que sea posible acerca del objetivo en el marco organizacional establecido o modelo de información definido; información como direcciones IP, nombres de máquinas, infraestructura de la red, perfiles y configuración de servidores, software, entre otros (Rodríguez, 2015, p. 31).

**Definición del alcance *Obligatorio***

En esta subsección se define con el cliente el alcance de la auditoría ya que se debe saber con exactitud lo que se va a hacer, qué tanto se va a hacer y hasta donde se va a hacer. Esta subsección se desarrollará con la información de los requerimientos funcionales y no funcionales del modelo de negocio. Esta subsección se hace con el fin de que el usuario tenga disponible el alcance de la auditoría y pueda tomar decisiones concernientes al tipo de prueba que se aplicará (caja negra, blanca o gris).

**Búsquedas en registros de internet *Obligatorio***

En esta subsección se definen y se aplican herramientas para búsqueda específica en la web ya que se debemos identificar datos particulares de la infraestructura básica del modelo de negocio objetivo de testeo. Esta subsección se desarrollará con datos clave, como el tipo de servidor de la máquina, bajo qué sistema operativo funciona, qué servicios tiene, qué puertos están abiertos, cuál es su configuración, con qué otras máquinas se conecta, entre otros. Esta subsección se hace con el fin de que el usuario tenga disponible los datos generados con las herramientas disponibles.

**Consultas en páginas públicas *Opcional***

Esta subsección muestra información asociada a la organización a la cual le estamos realizando la auditoría, sin embargo, se marca opcional ya que, si bien pueden ser útiles los datos que encontremos en la web pública, no es un ítem bloqueante para poder proceder

con la auditoría. Esta subsección se desarrollará con páginas, grupos, listas y motores de búsqueda generales públicos en la web. Esta subsección se hace con el fin de que el usuario tenga disponible las fuentes públicas de las cuales se extrajo información extra.

**Búsquedas con motores de búsqueda *Opcional***

Esta subsección muestra información relacionada con motores de búsqueda concretos, sin embargo, se marca opcional ya que, si bien pueden ser útiles los datos que encontremos con el motor, no es un ítem bloqueante para poder proceder con la auditoría. Esta subsección se desarrollará con información del fichero "robots.txt" y datos de motores de búsqueda concretos. Esta subsección se hace con el fin de que el usuario tenga disponible las fuentes concretas de las cuales se extrajo información extra.

**Búsqueda de sub-dominios *Obligatorio***

Esta subsección muestra información relacionada con la forma de encontrar nuevas máquinas objetivo ya que se puede comprobar si el dominio que estamos auditando tiene subdominios asociados. Esta subsección se desarrollará con información que se extraiga de la herramienta *Fierce* de Kali Linux. Esta subsección se hace con el fin de que el usuario tenga disponible los datos extraídos de la herramienta de penetración *Fierce*.

**Elaboración de diccionarios *Obligatorio***

Esta subsección muestra información relacionada con la elaboración de diccionarios ya que es otra técnica de hacking que nos puede servir en la auditoría. Esta subsección se desarrollará con información que se extraiga de la herramienta *CeWL* de Kali Linux. Esta subsección se hace con el fin de que el usuario tenga disponible los datos extraídos de la herramienta generadora de listas de palabras *CeWL*.

### 7.1.5 FASE DE MAPEADO

En esta sección se hace la recolección de toda la información que sea posible acerca aplicación web objetivo, el servicio web expuesto y el servidor web que los aloja. Para ello se emplearán distintas herramientas que generarán datos específicos como puertos, uso de protocolos y seguridad explícita de la aplicación.

**Escaneo de puertos y versiones *Obligatorio***

Esta subsección muestra información relacionada con el escaneado de puertos ya que en el tema de mapeado es una tarea inevitable para la auditoría. Esta subsección se desarrollará con información que se extraiga de la herramienta de escaneado de puertos TCP y UDP *Nmap*. Esta subsección se hace con el fin de que el usuario tenga disponible la información de los puertos extraídos con herramienta.

**Análisis SSL *Obligatorio***

Esta subsección muestra información relacionada con el análisis del protocolo seguro SSL ya que éste garantiza la confidencialidad de las comunicaciones vía web para la aplicación objeto de auditoría. Esta subsección se desarrollará con información que se extraiga de las herramientas *TLSSled* y *SSLDigger* las cuales analizan la seguridad de las implementaciones en los protocolos SSL/TLS de un servidor objetivo. Esta subsección se hace con el fin de que el usuario tenga disponible la información del análisis SSL realizado con dichas herramientas.

**Balanceadores de carga y WAF *Obligatorio***

Esta subsección muestra información relacionada con los balanceadores de carga y con cortafuegos de aplicaciones web (WAF) ya que con estos datos sabremos si la aplicación objeto de auditoría se encuentra alojada en varios servidores o sólo en uno y también si está protegido detrás de un cortafuegos. Esta subsección se desarrollará con información que se extraiga de las herramientas *Wafw00f* y *Halberd* las cuales analizan el balanceo de carga y protección de la aplicación web. Esta subsección se hace con el fin de que el usuario tenga disponible la información sobre balanceadores de carga y firewall de aplicaciones web.

**Análisis de configuración del software *Obligatorio***

Esta subsección muestra información relacionada a la configuración la aplicación objeto de auditoría ya que con estos datos entenderemos cómo está construido el sistema a nivel de software. Esta subsección se desarrollará con información que se extraiga de la herramienta *Nikto* la cual sirve para buscar estas vulnerabilidades en aplicaciones web. Esta subsección se hace con el fin de que el usuario tenga disponible la información sobre dichas vulnerabilidades, sí las hay.

**Spidering - Navegación manual *Obligatorio***

Esta subsección muestra información relacionada a la navegación manual en profundidad de la aplicación (también llamado rastreo de la aplicación) ya que con estos datos podemos conocer a fondo el sitio web e identificar todas las funciones que implemente. Esta subsección se desarrollará con información que se extraiga de la herramienta *ZAP* la cual sirve para identificar objetivos, recursos y parámetros, que luego se escanearon con el fuzzer contenido en la herramienta. Esta subsección se hace con el fin de que el usuario tenga disponible la información en base a los datos extraídos con ZAP.

**Spidering - Análisis de resultados *Obligatorio***

Esta subsección muestra información relacionada al análisis de los resultados una vez realizado el recorrido manual de todas las páginas ya que el objetivo es encontrar archivos

en el servidor web. Esta subsección se desarrollará con información que se extraiga con la función *Spider* de la herramienta *ZAP*. Esta subsección se hace con el fin de que el usuario tenga disponible la información en base a los datos encontrados con la función "Spider" de ZAP.

### 7.1.6 FASE FINAL DE EXPLOTACIÓN

En esta sección se desarrollarán dos tareas principales, las cuales son detección y verificación. Cada una de éstas tareas estará compuesta por hallazgos concretos que se logren encontrar en esta auditoría. Cada una de las demás auditorías será distinta sobre todo en esta fase, ya que los hallazgos serán distintos cada vez, así que las pruebas que habrá que realizar cambiarán. También cambiarán en función de otros factores, como la aplicación a auditar, o las tecnologías que use el prototipo, para éste caso específicamente son JSF, JPA, EJB, Primefaces, Glassfish, entre otros.

**Detección *Obligatorio***

Esta subsección muestra información relacionada a la búsqueda y detección vulnerabilidades a partir de los resultados obtenidos durante la fase de mapeado ya que se debe probar cómo se comporta la aplicación ante errores introducidos a propósito. Esta subsección se desarrollará con información que se arroje del ejercicio del comportamiento de la aplicación ante errores, luego se escanean los recursos o métodos que envían parámetros al servidor objetivo y, por último, se emplearán ataques de fuzzing (técnica de pruebas en software) sobre los parámetros encontrados en los puntos anteriores. Esta subsección se hace con el fin de que el usuario tenga disponible la información obtenida de toda la tarea de detección.

**Verificación SQL Injection *Obligatorio***

Esta subsección muestra información relacionada a la verificación manual de los descubrimientos previos ya que debemos analizar cada hallazgo sospechoso que aún no haya sido confirmado. Esta subsección se desarrollará con información específica del segmento de SQL Injection en el modelo propuesto en base a la revisión que se hizo en dicho apartado. Esta subsección se hace con el fin de que el usuario tenga disponible la información obtenida de todo el ejercicio de verificación de vulnerabilidades usando el vector de ataque SQL Injection.

### 7.1.7 RESULTADOS Y CONCLUSIONES

En esta sección se muestran los resultados finales de la auditoría y unas conclusiones del trabajo realizado y se compara con los objetivos propuestos al inicio del proceso, para comprobar si efectivamente se cumplieron.

## 7.2 IMPLEMENTACIÓN DEL MODELO EN LA HERRAMIENTA DE BASE DE CONOCIMIENTO OPENKM

Una vez se definió el modelo, se implementó en el portal de RITA la herramienta en la cual se soporta la base de conocimiento, de acuerdo a investigaciones descritas anteriormente y seleccionando la más idónea para el proyecto, ésta solución es OpenKM en su versión libre, vamos a ingresar como administrador, para ver todo lo que ofrece:

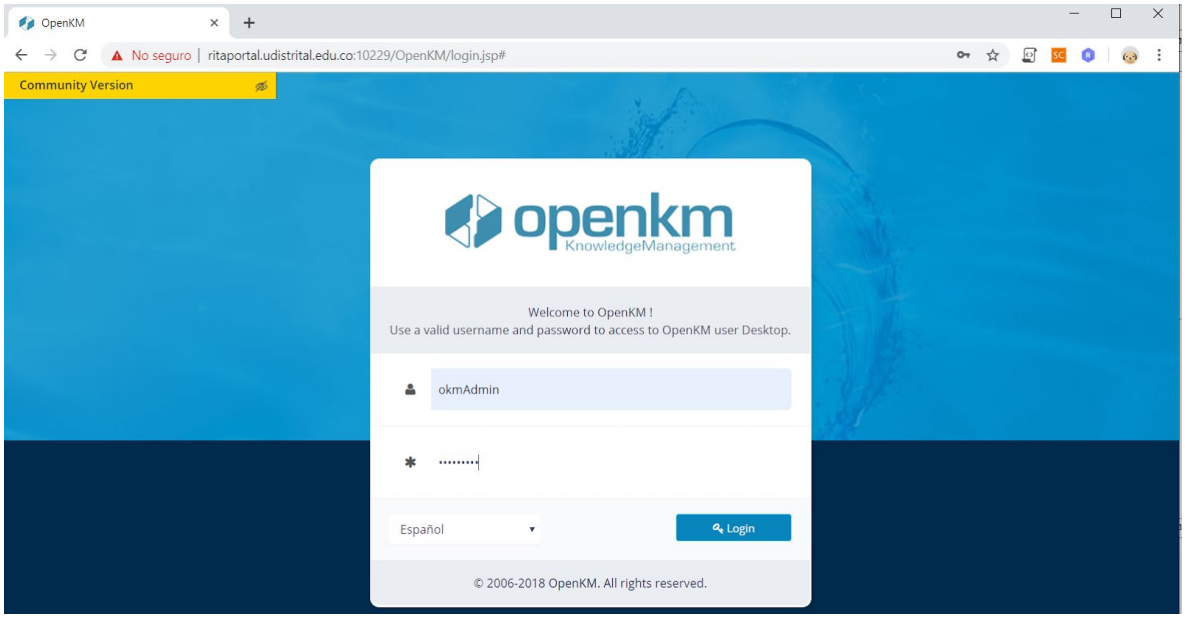



Figura 37. Ventana de inicio de sesión de la herramienta OpenKM que soporta el modelo.

Se ejemplificó el modelo en la herramienta OpenKM, como se demuestra a continuación:

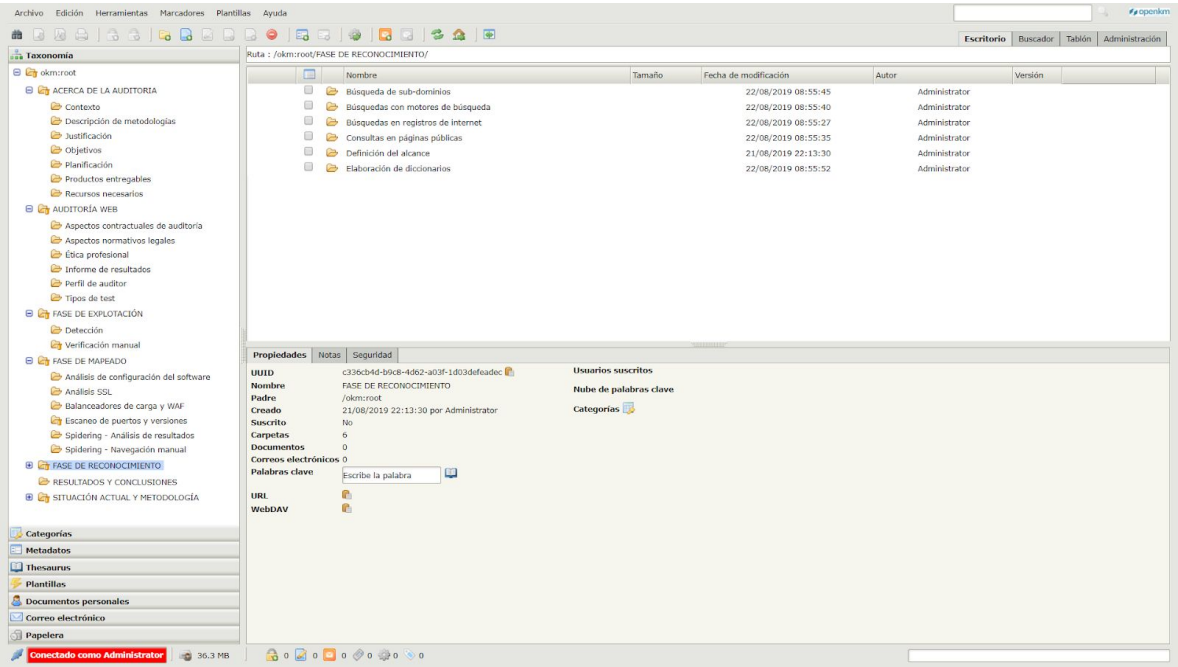



Figura 38. Implementación del modelo propuesto en la herramienta OpenKM

La herramienta permite visualizar los archivos en pdf almacenados en la base, con formato .doc para documentos, .xls para hojas de cálculo y .ppt para presentaciones.

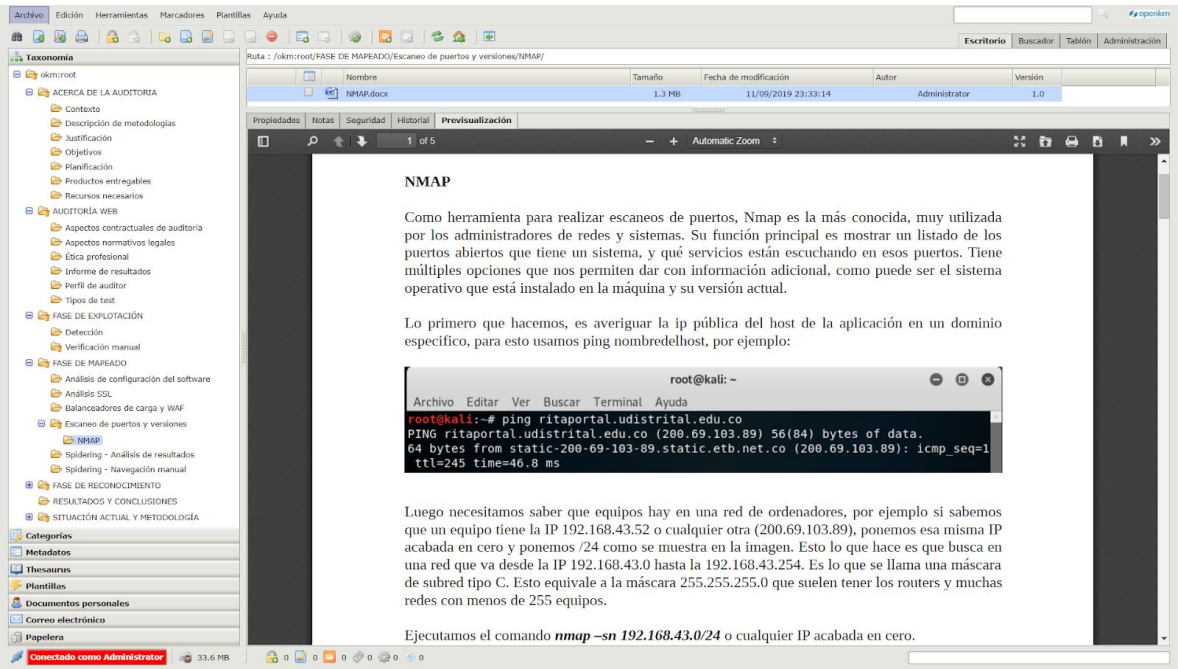



Figura 39. Visualización de archivos del modelo propuesto en la herramienta OpenKM.

### 7.3 Categorización del modelo

En la Base de Conocimiento se reúnen categóricamente los artefactos, conceptos, ejemplos, guías, lineamientos y listas de chequeo que se refieren en el desarrollo del modelo y categorizan los archivos del modelo; éstos se describen de la siguiente forma:

**Artefactos**: un subproducto generado durante el proceso de la auditoría de los servicios web.

**Concepto:** definiciones de los elementos necesarios para llevar a cabo la auditoría de los servicios web.

**Ejemplos:** casos de prueba de ejercicios propios de la auditoría en un momento específico del proceso de evaluación.

**Guías:** Documento con un paso a paso a seguir para lograr un objetivo específico de la auditoría

**Lineamientos:** directrices, normas, reglas y lineamientos propios en el contexto internacional y nacional en el cual se va a aplicar y a guiar la auditoría.

**Listas de chequeo:** listas particulares generadas durante todo el proceso para la evaluación de los elementos objetivo de auditoría.

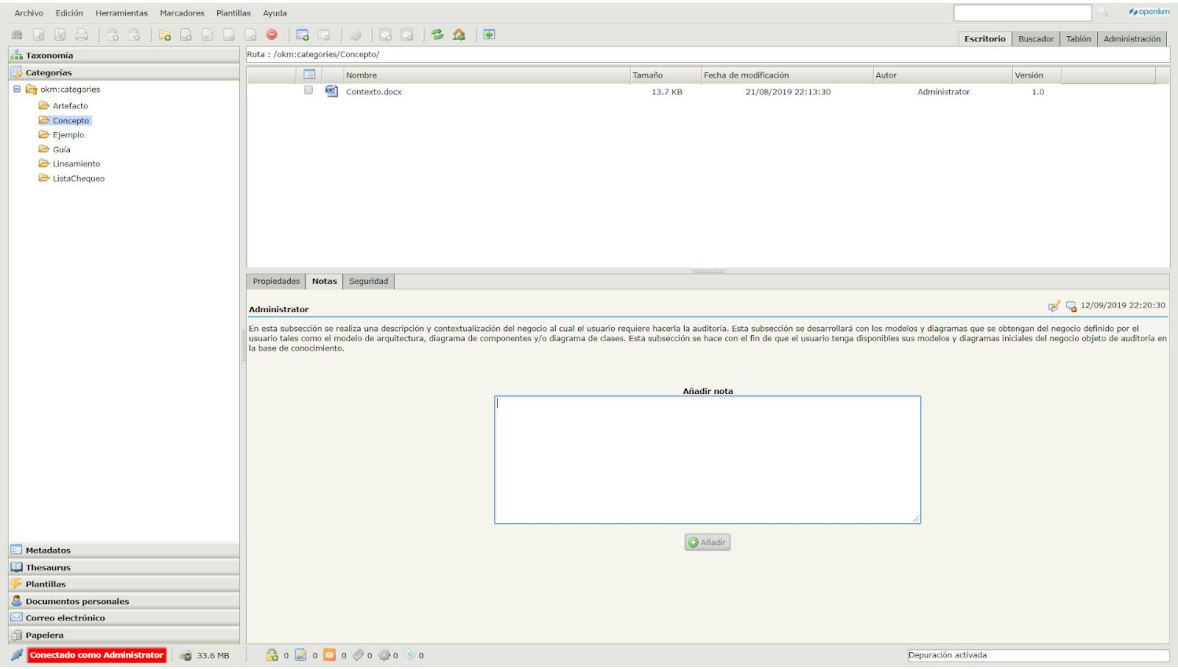



Figura 40. Categorías del modelo propuesto en la herramienta OpenKM.

## 7.4 Metadata del modelo

En la herramienta OpenKm viene por defecto una estructura para la generación de la metadata (descripción de los datos) del contenido, para éste caso se realizaron unas modificaciones de acuerdo a las necesidades propias del modelo y se definió que al marcar un archivo de auditoría como ***Obligatorio*** se almacena en una etiqueta ***True***, si se marca como ***Opcional***, se almacena en la etiqueta False.

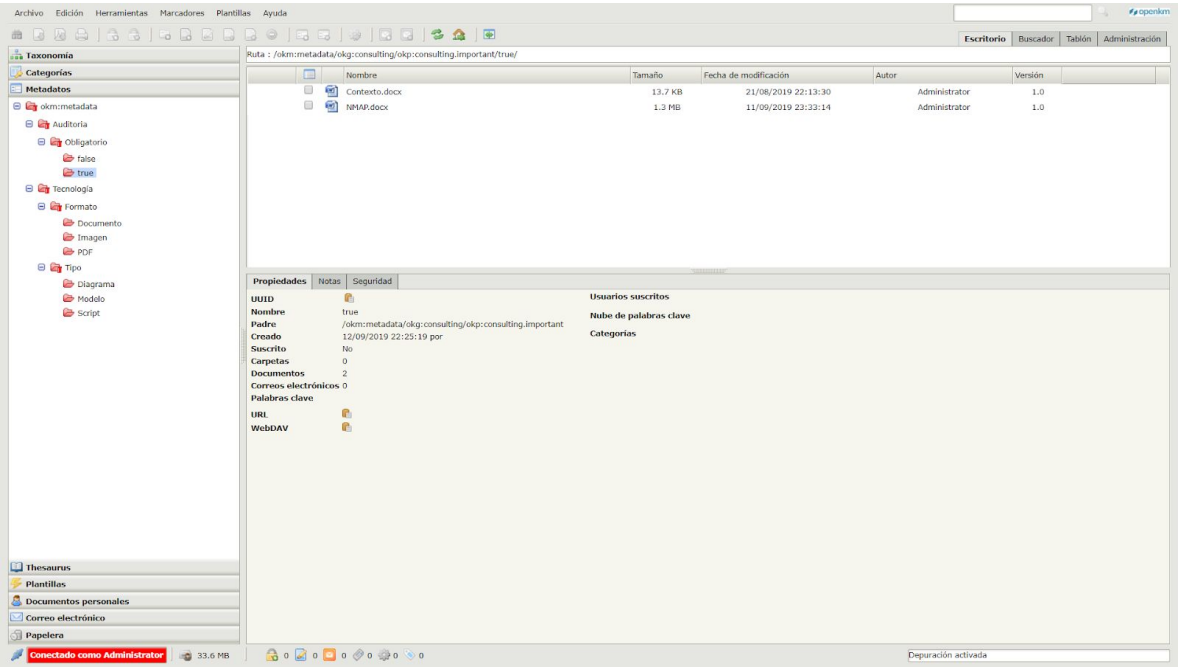



Figura 41. Metadata del modelo propuesto en la herramienta OpenKM.

## 7.5 Plantillas usando los elementos de evaluación del modelo en OpenKM

La herramienta OpenKm permite agregar plantillas, son estructuras de árbol de carpetas para organizar la principal fuente de documentos. La idea es que esa organización crea nuevos documentos (para éste caso se incluyó los elementos de evaluación de la base de conocimientos descritos previamente en el apartado de METODOLOGÍA del presente documento) a partir de documentos dentro del panel de plantillas:

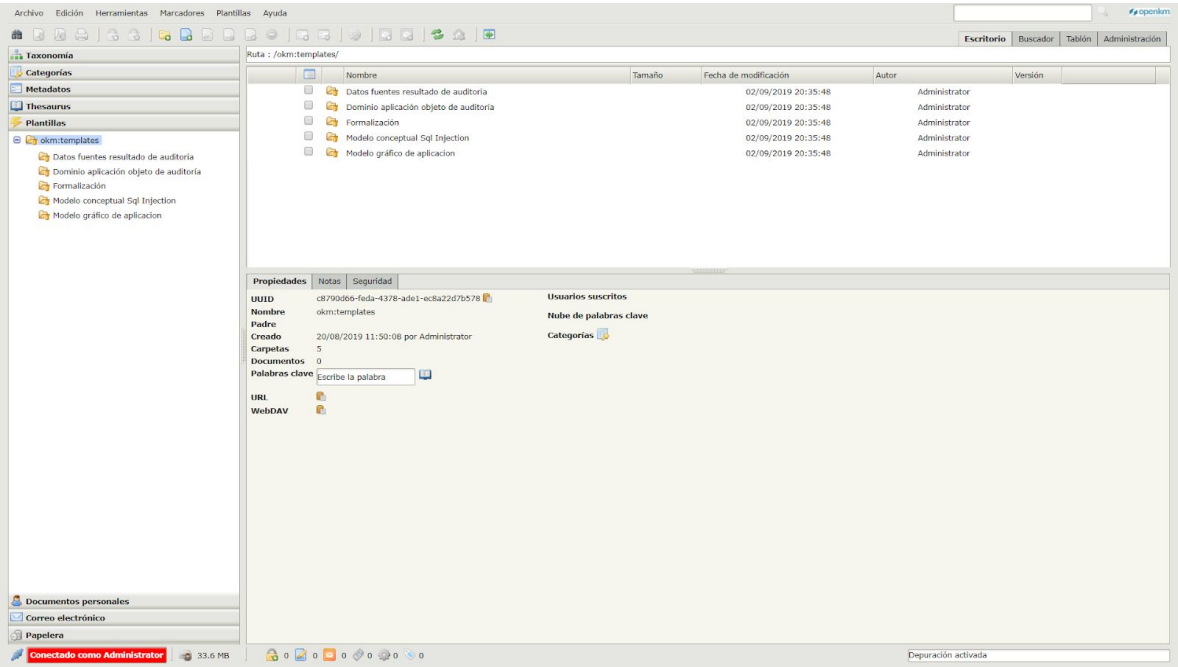



Figura 42. Plantillas en la herramienta OpenKM del modelo propuesto.

## 7.6 Búsquedas en OpenKM como herramienta base de conocimiento

Las búsquedas pueden hacerse por palabras en el contenido de los archivos, por el nombre (título) de los documentos o por el tipo de archivo (tipo MIME). Del mismo modo, si se proporciona una palabra clave para los archivos, puede buscar utilizando este campo. También puede hacer búsquedas por valores de grupo de propiedades si se han definido.

La pestaña Buscar, ubicada en la esquina superior derecha de la pantalla, cambia los paneles de escritorio por los de Búsqueda, o bien se puede ingresar a la pestaña "Buscador".

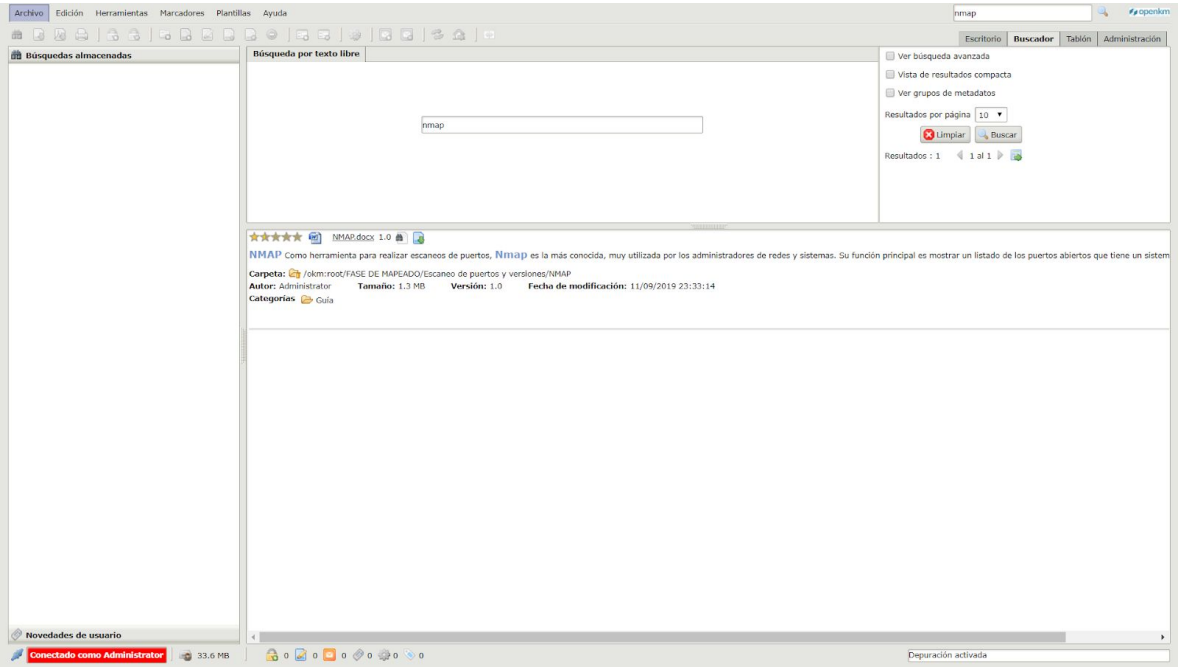



Figura 43. Búsquedas en la herramienta OpenKM del modelo propuesto.

## 7.7 Tablero en OpenKM como herramienta base de conocimiento

El Tablero tiene varios servicios de información y utilidades para usuarios, documentos generales y de correo electrónico, vistas de consultas de documentos específicos, el servicio de búsqueda push (noticias), tareas de flujo de trabajo y el mapa de palabras clave (vista de nube de etiquetas).

La pestaña Tablero, ubicada en la esquina superior derecha de la pantalla, cambia los paneles de escritorio por los del Tablero.

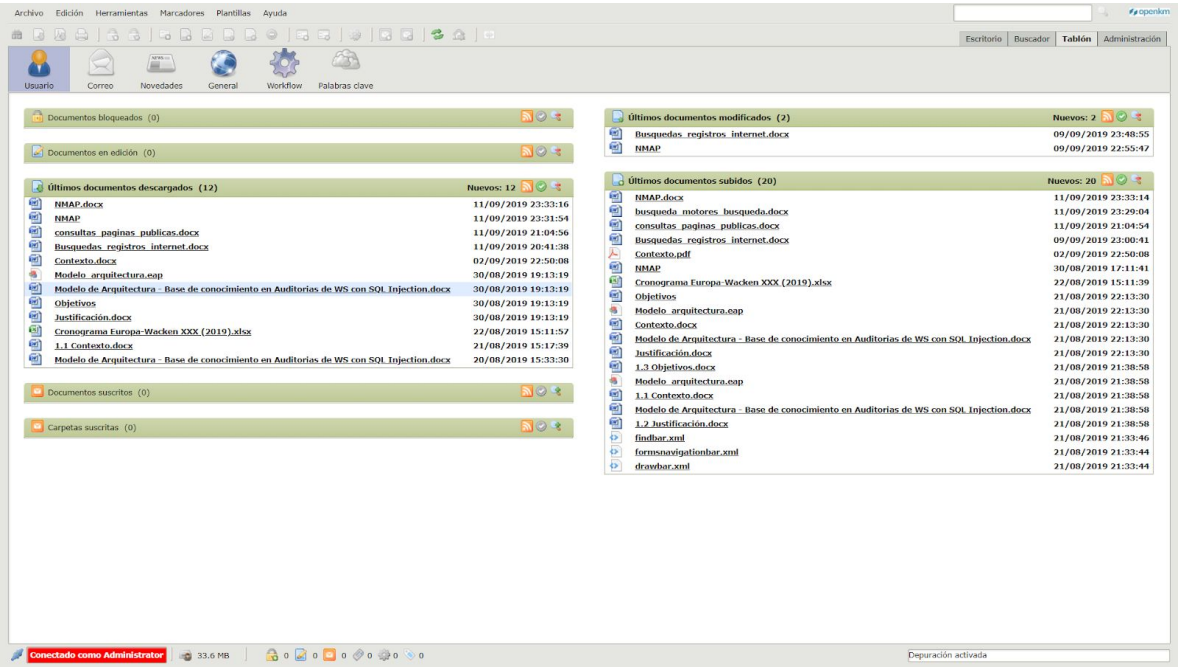



Figura 44. Tablero en la herramienta OpenKM del modelo propuesto.

## 7.8 Administración en OpenKM como herramienta base de conocimiento

OpenKM es una herramienta para administrar documentos generados dentro de una empresa, para mejorar el acceso dentro de todo el entorno corporativo y para tratar de controlar la duplicación innecesaria de información.

La administración de OpenKM proporciona la solución para:

- Controlar el creciente volumen de información a gestionar
- Administrar usuarios
- Ver las propiedades reales del archivo de configuración de OpenKM.cfg cargadas por OpenKM al iniciarse.
- Scripting
- Acceso de bajo nivel al repositorio
- Metadatos
- Consultas y herramientas de informes
- Auditar usuarios (registro de seguimiento)
- Administrar el flujo de trabajo

- Importar y exportar datos desde su servidor
- Gestionar thesaurus

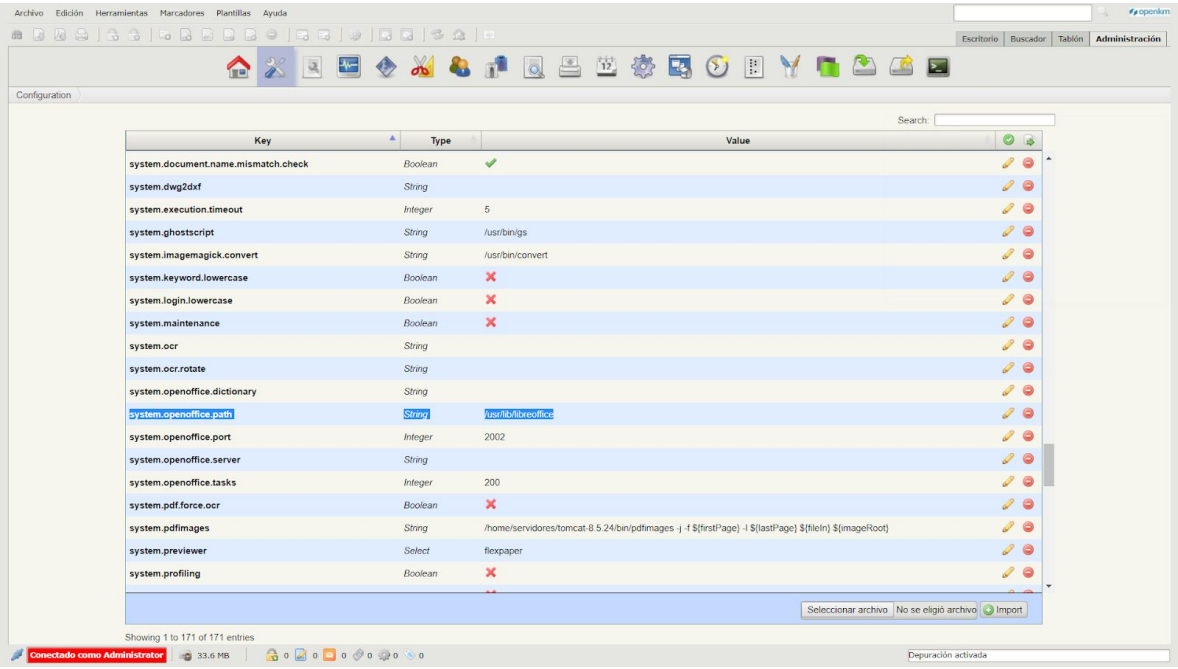



Figura 45. Administración en la herramienta OpenKM del modelo propuesto.

## 7.9 Escenario usuario en OpenKM como herramienta base de conocimiento

Para hacer el ingreso como usuario a la herramienta que soporta el modelo de base de conocimiento propuesto, y también acceder a la auditoría completa se creó un perfil de usuario y se parametrió desde la herramienta:

Link: http://ritaportal.udistrital.edu.co:10229/

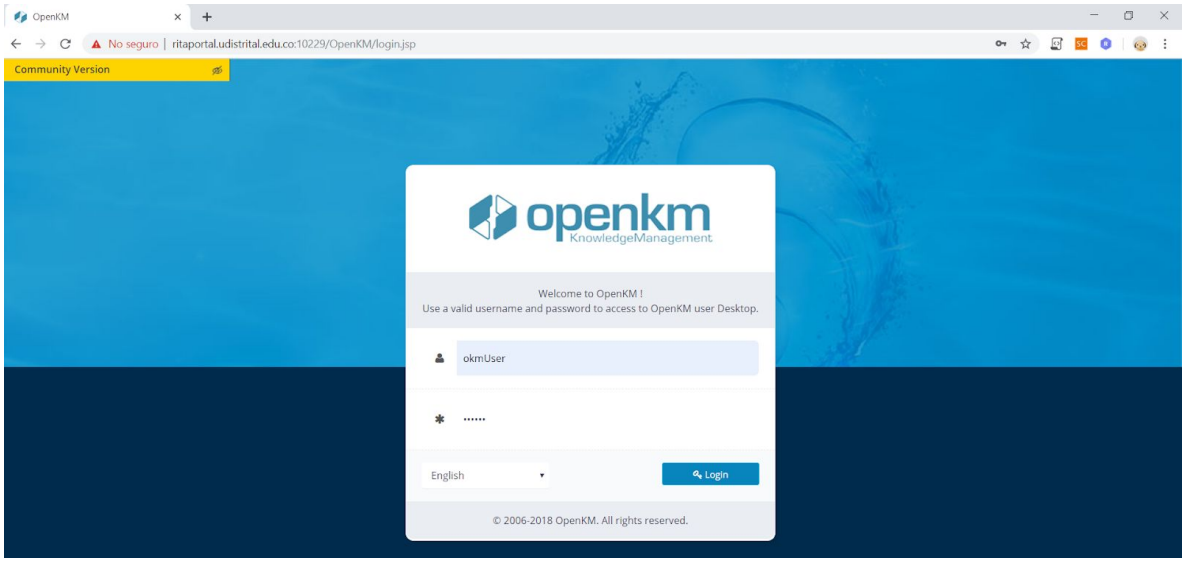



Figura 46. Ingreso a la herramienta OpenKM del modelo propuesto.

Una vez se accede a la herramienta OpenKM, nos aparecerá la pantalla principal de la aplicación, como se ve a continuación:

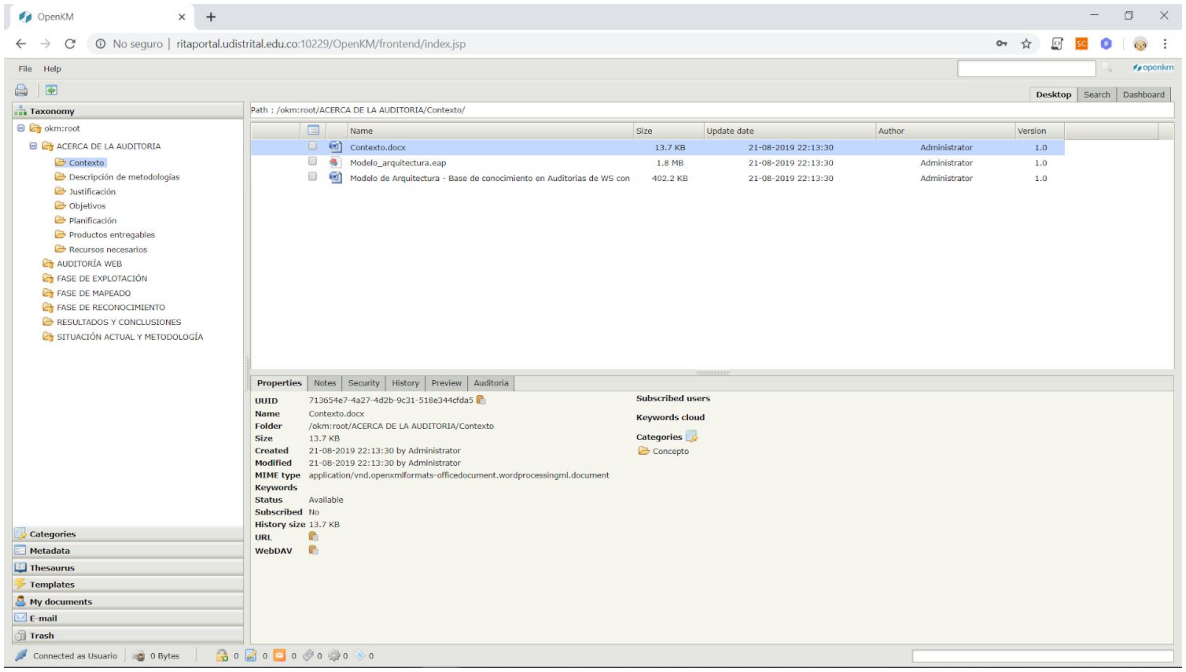
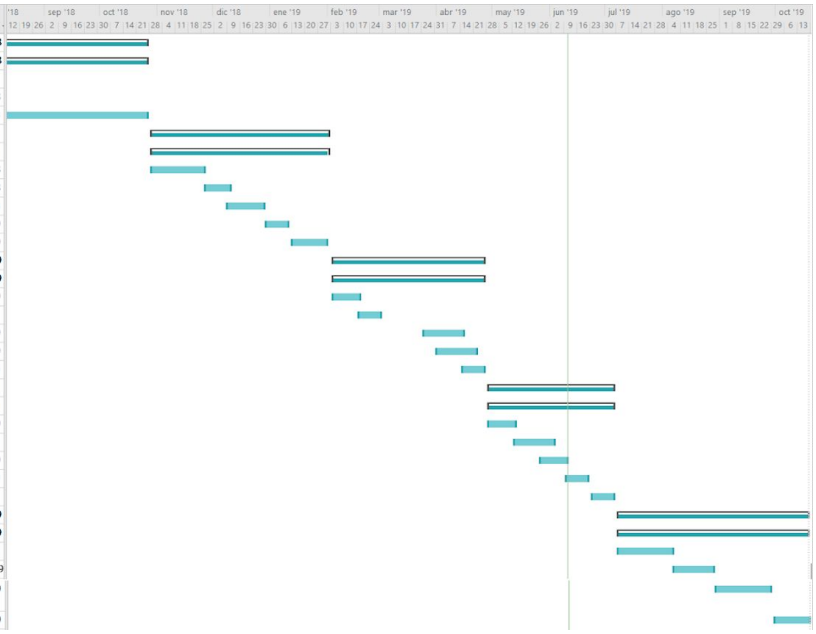



Figura 47. Pantalla inicial de la herramienta OpenKM del modelo propuesto.

# 8. CRONOGRAMA

| Nombre de tarea | Duración | Comienzo | Fin |
|---|---|---|---|
| **ITERACION 1** | **291 días** | **sáb 16/09/17** | **vie 26/10/18** |
| **FASE DE INICIO** | **291 días** | **sáb 16/09/17** | **vie 26/10/18** |
| **Requerimientos** | **100 días** | sáb 16/09/17 | jue 1/02/18 |
| **Análisis** | **100 días** | vie 2/02/18 | jue 21/06/18 |
| **Diseño del Prototipo** | **91 días** | vie 22/06/18 | vie 26/10/18 |
| **ITERACION 2** | **70 días** | **lun 29/10/18** | **vie 1/02/19** |
| **FASE DE ELABORACIÓN** | **70 días** | **lun 29/10/18** | **vie 1/02/19** |
| **Requerimientos** | **21 días** | lun 29/10/18 | lun 26/11/18 |
| **Análisis** | **10 días** | mar 27/11/18 | lun 10/12/18 |
| **Diseño del Prototipo** | **16 días** | dom 9/12/18 | vie 28/12/18 |
| **Arquitectura del Modelo** | **10 días** | dom 30/12/18 | jue 10/01/19 |
| **Codificación SQL Injection** | **15 días** | dom 13/01/19 | jue 31/01/19 |
| **ITERACION 3** | **60 días** | **lun 4/02/19** | **vie 26/04/19** |
| **FASE DE CONSTRUCCIÓN** | **60 días** | **lun 4/02/19** | **vie 26/04/19** |
| **Requerimientos** | **11 días** | lun 4/02/19 | lun 18/02/19 |
| **Análisis** | **10 días** | lun 18/02/19 | vie 1/03/19 |
| **Diseño del Prototipo** | **16 días** | lun 25/03/19 | lun 15/04/19 |
| **Arquitectura del Modelo** | **16 días** | lun 1/04/19 | lun 22/04/19 |
| **Codificación SQL Injection** | **10 días** | lun 15/04/19 | vie 26/04/19 |
| **ITERACION 4** | **50 días** | **lun 29/04/19** | **vie 5/07/19** |
| **FASE DE CONSTRUCCIÓN** | **50 días** | **lun 29/04/19** | **vie 5/07/19** |
| **Análisis** | **11 días** | lun 29/04/19 | lun 13/05/19 |
| **Diseño del Prototipo** | **16 días** | lun 13/05/19 | lun 3/06/19 |
| **Arquitectura del Modelo** | **11 días** | lun 27/05/19 | lun 10/06/19 |
| **Codificación SQL Injection** | **10 días** | lun 10/06/19 | vie 21/06/19 |
| **Despliegue** | **10 días** | lun 24/06/19 | vie 5/07/19 |
| **ITERACION 5** | **75 días** | **lun 8/07/19** | **vie 18/10/19** |
| **FASE DE TRANSICION** | **75 días** | **lun 8/07/19** | **vie 18/10/19** |
| **Diseño del Prototipo** | **22 días** | lun 8/07/19 | mar 6/08/19 |
| **Arquitectura del Modelo** | **16 días** | mié 7/08/19 | mié 28/08/19 |
| **Integración de elementos del modelo** | **22 días** | jue 29/08/19 | vie 27/09/19 |
| **Despliegue** | **15 días** | lun 30/09/19 | vie 18/10/19 |



# 9. CONCLUSIONES

A continuación, respondemos el objetivo general y los objetivos específicos, demostrando si las actividades realizadas cumplen con el planteamiento inicial de éste proyecto.

**General**

*"Implementar un modelo base de conocimiento para la auditoría de seguridad en Servicios Web":*

- Con este proyecto se logró crear un modelo de base de conocimiento implementado sobre una herramienta libre, ejecutando una auditoría en seguridad de servicios web con SQL Injection sobre un prototipo organizacional.

**Específicos**

*"Implementar metodologías y buenas prácticas en el tema de seguridad de la información"*

- Se logró establecer una metodología propia para el desarrollo del modelo propuesto, incluyendo metodologías ya definidas, métodos, buenas prácticas, técnicas, recomendaciones y tendencias actuales relacionadas con conceptos de base de conocimiento, auditoría de aplicaciones web, seguridad de la información y desarrollo de software.

*"Establecer una metodología propia para el modelo, con metodologías definidas, métodos y buenas prácticas"*

- En contexto con el anterior objetivo, se hicieron revisiones literarias sobre investigaciones en el tema y se estudiaron guías, técnicas y herramientas, para llevar a cabo la auditoría en servicios web, determinando que para el vector de ataque SQL Injection, en el contexto establecido, la prevención no es responsabilidad del JSF, la forma de evitar el ataque depende del API de persistencia que se esté utilizando (JPA para el caso), pero todo se reduce a que nunca se debe concatenar la entrada controlada por el usuario en cadenas SQL, siempre se debe usar consultas parametrizadas cuando corresponda.

*"Alimentar un software de base de conocimiento con datos resultantes de pruebas de Inyección SQL"*

- Se realizó la instalación, configuración y alimentación del software OpenKM, como herramienta que soporta el modelo propuesto, con los datos y la información resultante del proceso de auditoría de los servicios web en un prototipo definido.

# BIBLIOGRAFÍA

Allen AU and Walter S. L. FUNG. (2018). Knowledge Audit Model for Information Security. The Chinese University of Hong Kong, Shatin, Hong Kong. The Hong Kong Polytechnic University, Hunghom. China.

Yogesh Kumar Mittal , Dr Santanu Roy and Dr. Manu Saxena. (2010). A Knowledge Management Model to Improve Information Security. IJCSI International Journal of Computer Science Issues. India.

Anna Bennett. (2015). Development of expert system for information security audit. New Developments in Expert Systems Research. Pages 19-65. USA.

Atymtayeva L., Kozhakhmet K., Bortsova G. (2014). Building a Knowledge Base for Expert System in Information Security. Kazakh-British Technical University. Kazakhstan.

Shuangyan Liu, Ching-hang Cheung, Lam-for Kwok. (2006). A Knowledge Framework for Information Security Modeling. City University of Hong Kong. China.

Yuangang YAO, Hui LIU, Jin YI, Xianghui ZHAO, Lin LIU. (2014). A Semantic Knowledge Base Construction Method for Information Security. China Information Technology Security Evaluation Center Beijing. China.

Nader Sohrabi Safa, Rossouw Von Solms. (2016). An information security knowledge sharing model in organizations. Centre for Research in Information and Cyber Security, School of ICT, Nelson Mandela Metropolitan University, Port Elizabeth. South Africa.

Walter S. L. Fung ; Richard Y. K. Fung. (2008). Knowledge-Centric Information Security. International Conference on Security Technology. Hainan Island, China.

Gruber T. (1993). Principles for the design of ontologies used for knowledge sharing. Available as Technical Report KSL 93-04. Knowledge Systems Laboratory, Stanford University. Disponible en: http://citeseer.ist.psu.edu/gruber93toward.html

Amirmohammad Sadeghian, Mazdak Zamani, Azizah Abd. Manaf. (2013). A Taxonomy of SQL Injection Detection and Prevention Techniques. Universiti Teknologi Malaysia. Malaysia.

Amirmohammad Sadeghian, Mazdak Zamani, Shahidan M. Abdullah. (2013). A taxonomy of SQL Injection Attacks. Universiti Teknologi Malaysia. Malaysia.

Amirmohammad Sadeghian, Mazdak Zamani, Suhaimi Ibrahim. (2013). SQL Injection is Still Alive:A Study on SQL Injection Signature Evasion Techniques. Universiti Teknologi Malaysia. Malaysia.

Rodríguez Sánchez Miriam. (2015). Auditoría de aplicaciones web: metodología y práctica profesional. Universitat Oberta de Catalunya. España.

Contreras Flórez Johan Lorenzo. (2017). Propuesta de auditoría a las aplicaciones web de la empresa C&M consultores aplicando herramientas de software libre. Universidad Nacional Abierta y a Distancia de Bogotá.

Farías Terrens María Gabriela, Infante Salgado Andrea. (2005). Seguridad informática en web services. Universidad Javeriana de Bogotá.

Gunawan, T.S. Lim, M.K. Kartiwi, M. Malik, N.A. Ismail, N. (2018). Penetration testing using Kali linux: SQL injection, XSS, wordpress, and WPA2 attacks.

Huang, Y. -., Huang, S. -., Lin, T. -., & Tsai, C. (2003). Web application security assessment by fault injection and behavior monitoring. Paper presented at the *Proceedings of the 12th International Conference on World Wide Web, WWW 2003,* 148-159. doi:10.1145/775152.775174.

Ayewah, N., Hovemeyer, D., Morgenthaler, D. J., Penix, J., & Pugh, W. (2008). Using static analysis to find bugs. IEEE Software, 25(5), 22-29. doi:10.1109/MS.2008.130

Y. Kosuga, K. Kono, M. Hanaoka, M. Hishiyama and Y. Takahama. (2007). "Sania: Syntactic and Semantic Analysis for Automated Testing against SQL Injection," Twenty-Third Annual Computer Security Applications Conference (ACSAC). doi: 10.1109/ACSAC.2007.20

Guaman, D., Guaman, F., Jaramillo, D., & Sucunuta, M. (2017). Implementation of techniques and OWASP security recommendations to avoid SQL and XSS attacks using J2EE and WS-security. Paper presented at the Iberian Conference on Information Systems and Technologies, CISTI, doi:10.23919/CISTI.2017.7975981

G. Serme, A. S. De Oliveira, J. Massiera, and Y. Roudier. (2012). “Enabling message security for RESTful services,” Proc. - IEEE 19th Int. Conf. Web Serv. ICWS.

G. McGraw. (2004). “Software Security,” Secur. Privacy, IEEE, vol. 2, no. 2.

M. Yag. (2015). “Integrando la Ingeniería de Seguridad en un Proceso de Ingeniería Software”.

A. Nadalin, C. Kaler, P. Hallam-Baker, R. Monzillo. (2004). "Web Services Security: SOAP Message Security 1.0," Security.

L. E. Moser, P. M. Melliar-Smith, and W. Zhao. (2007). "Building dependable and secure web Services," J. Softw., vol. 2, no 1.

Y. Makino, S., Tamura, K., Imamura, T., & Nakamura. (2004). "WSSecurity, Implementation and performance of Security," Int. J. Web Serv. Res., vol. 1.

M. Naedele. (2003). "Standards for XML and web services security," Computer (Long. Beach. Calif)., vol. 36, no. 4.

ISO/IEC 27000. (2018). Information technology — Security techniques — Information security management systems — Overview and vocabulary.

CONPES 3701. (2011). Consejo Nacional de Política Económica y Social - República de Colombia.

Feigenbaum, E.A. (1980). Knowledge Engineering: the Applied Side of Artificial Intelligence. Report STAN-CS-80-812. Department of Computer Science, Stanford University

Chenjian Hao. (2011). Research on Knowledge Model for Ontology-Based Knowledge Base. Xi'an University of Industry Xi'an, China.

Rauch-Hindin, W. B. (1989). Aplicaciones de la inteligencia artificial en la actividad empresarial, la ciencia y la industria. Díaz De Santos, S. A.

Eleanor A. Varela-Tapia, Delia Estrada-Becerra, Iván L. Acosta-Guzmán. (2017). Wiki, herramienta informática para la base de conocimiento para el proyecto PROMEINFO de la Universidad de Guayaquil.

Moreno Fernández, F. (2001). "Prototipos y modelos de lengua". *Carabela. Modelos de uso de la lengua española 50: 5-20.*

Carvajal Villaplana Álvaro. (2002). Teorías y modelos: Formas de representación de la realidad. Red de revistas científicas de América Latina y el Caribe, España y Portugal.

Anisleiby Fernández Hernández, María José López Huertas, Yusnerkis Prevot Urgellés. (2015). Modelo de sistema de organización del conocimiento basado en ontologías.

ISO/IEC 9075. (2003). Information technology -- Database languages -- SQL -- Part 1: Framework (SQL/Framework)

Carme Martín Escofet. (2002). El lenguaje SQL. Universitat Oberta de Catalunya.

Paul Weinberg. James Groff. Andrew Oppel. (2010). SQL: The Complete Reference, Third Edition

Justin Clarke. (2012). SQL Injection Attacks and Defense.

OWASP Foundation. (2004). Sobre OWASP. Obtenido de https://www.owasp.org/index.php/Sobre_OWASP

OWASP Foundation. (2008). Top 10 2007-Fallas de Inyección. Obtenido de https://www.owasp.org/index.php/Top_10_2007-Fallas_de_Inyecci%C3%B3n

OWASP Foundation. (2016). LDAP injection. Obtenido de https://www.owasp.org/index.php/LDAP_injection.

OWASP Foundation. (2017). Top 10-2017. Obtenido de https://www.owasp.org/index.php/Top_10-2017_A1-Injection

Litchfield, D. (2005). Data Mining with SQL Injection and Inference. NGS Software. Obtenido de: https://dl.packetstormsecurity.net/papers/attack/sqlinference.pdf

Litchfield, D., Anley, C., Heasman, J., Grindlay, B. (2005). The Database Hacker's Handbook: Defending Database Servers: John Wiley & Sons.

Pinzón Cristian. (2010). Arquitectura multi-agente adaptativa para la detección de ataques en entornos dinámicos y distribuidos.

Halfond, W. G. J., Orso, A. (2005a). AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks. In 20th IEEE/ACM international Conference on Automated software engineering (ASE '05) (pp. 174-183). New York, NY, USA: ACM. doi:10.1145/1101908.1101935

Halfond, W. G. J., Orso, A. (2005b). Combining static analysis and runtime monitoring to counter SQL-injection attacks. In Third international workshop on Dynamic analysis (WODA '05) (pp. 1-7). St. Louis, Missouri: ACM. doi:10.1145/1083246.1083250 Bibliografía 210

Halfond, W. G. J., Viegas, J., Orso, A. (2006). A Classification of SQL-Injection Attacks and Countermeasures. In IEEE International Symposium on Secure Software Engineering. Arlington, VA, USA

Minero Guardado Jesús J. (2011). Identificación y Clasificación de las Mejores Prácticas para Evitar la Inyección SQL en Aplicaciones Desarrolladas en PHP y PostgreSQL.

Meléndez Sergio. (2014). Diseño e implementación de un algoritmo para detección de inyección SQL en sitios web para dispositivos móviles.

IBM Corporation. (2019). X-Force IRIS Data Breach Report. Obtenido de https://www.ibm.com/security/resources/xforce/xfisi/

Coronado Steven. (2017). Desarrollo de una guía metodológica basada en análisis SQL injection y formas de protección a las bases de datos.

Singh Jai Puneet. (2017). Analysis of SQL Injection Detection Techniques

K. Wei, M. Muthuprasanna, and S. Kothari. (2006). Preventing SQL injection attacks in stored procedures. In Australian Software Engineering Conference (ASWEC'06).

Chapman, J., Van der Stock, A., Keary, E., Perego, P., Lowry, D., Rook, D., Cruz, D. and Williams, J. (2017). OWASP Code Review Guide v2, OWASP Foundation, November.

Birbeck, M., Duckett, J., Gudmundsson, O.G., Kobak, P., Lenz, E., Livingstone, S., Marcus, D., Mohr, S., Pinnock, J., Visco, K., Watt, A., Willians, K., Zaev, Z., Ozu, N. (2001). Professional XML 2nd Edition. Wrox Press.

Lam, T.C., Ding, J.J., Liu, J.C. (2008). XML Document Parsing: Operational and Performance Characteristics. IEEE Computer, 41(9), pp. 30-37.

D. B. Networks. (2009). SQL Injection Attack: Detection in a Web Application Environment. Obtenido de: http://www.dbnetworks.com/pdf/sql-injection-detection-web-environment.pdf

Moradian, E., Håkansson, A. (2006). Possible attacks on XML Web Services. International Journal of Computer Science and Network Security (IJCSNS), 6(1B), 154-170.

Cerami, E. (2002). Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI & WSDL. First Edition Sebastopol, CA, USA: O'Reilly & Associates.

J. Krishnaveni, R. K. Kavitha, P. Sujitha, T. Jenitha. (2016). Securing Web Application from SQL Injection Attacks.

Xin Liu, Qingchen Yu, Xiaokang Zhou, Qingguo Zhou. (2018). OwlEye: An Advanced Detection System of Web Attacks Based on HMM.

Universidad de Berkeley. (2019). Cómo protegerse contra los ataques de inyección SQL.

MinTIC. (2016). Guía de gestión de riesgos. Seguridad y privacidad de la información. https://www.mintic.gov.co/gestionti/615/articles-5482_G7_Gestion_Riesgos.pdf

MinTIC. (2016). Guía de auditoría. Seguridad y privacidad de la información. https://www.mintic.gov.co/gestionti/615/articles-5482_G15_Auditoria.pdf

MinTIC. (2016). Modelo de seguridad y privacidad de la Información. Seguridad y privacidad de la información. https://www.mintic.gov.co/gestionti/615/articles-5482_Modelo_de_Seguridad_Privacidad.pdf

MinTIC. (2018). Resolución 0002007 de 2018. Política de tratamiento de datos. https://www.mintic.gov.co/portal/604/articles-62124_politica_tratamiento_datos_personales_v20180917.pdf

JUnit User Guide. (2019). The JUnit Team.

OpenUP . (2005). Introduction to OpenUP (Open Unified Process)

Guaman, D., Guaman, F., Jaramillo, D., Sucunuta, M. (2017). Implementation of techniques and OWASP security recommendations to avoid SQL and XSS attacks using J2EE and WS-Security | [Implementación de técnicas y recomendaciones de seguridad OWASP para evitar ataques de tipo inyección SQL, XSS utilizando J2EE y WS-Security]

RFC 4511. (2006). Lightweight Directory Access Protocol (LDAP): The Protocol

Méndez Giraldo Germán, Álvarez Pomar Lindsay. (2003). Metodología para la construcción de la base de conocimiento de un sistema experto. Universidad Distrital. Bogotá