ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DESARROLLO DE UN PROTOTIPO DE SISTEMA QUE PERMITA LA IDENTIFICACIÓN Y PREDICCIÓN DE ATAQUES A SISTEMAS DE BASES DE DATOS UTILIZANDO TÉCNICAS DE MINERÍA DE DATOS

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

CESAR WASHINGTON AÑASCO LOOR

cesar.anasco@epn.edu.ec

KAREN GABRIELA MOROCHO CAIZA

karen.morocho@epn.edu.ec

DIRECTOR: PhD. Hallo María

maria.hallo@epn.edu.ec

Quito, Julio 2021

DECLARACIÓN

Nosotros, **Cesar Washington Añasco Loor** y **Karen Gabriela Morocho Caiza**, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

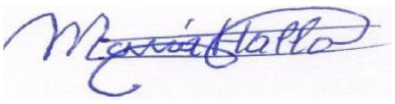
A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Cesar Washington Añasco Loor

Karen Gabriela Morocho Caiza

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Cesar Washington Añasco Loor y Karen Gabriela Morocho Caiza, bajo mi supervisión.



PhD. María Hallo

DIRECTOR DE PROYECTO

AGRADECIMIENTOS

Mi eterno agradecimiento a mis padres, el apoyo y esfuerzo constante que me han dedicado es una deuda que nunca acabaré de pagar.

Agradezco a mi hermano José, por brindarme el computador con el que termine la carrera.

A mis hermanas Yanira y Jazmín, por aceptarme y defenderme siempre.

Cesar Añasco

AGRADECIMIENTOS

A Dios por ser el motor principal de la vida, porque ha sido bueno conmigo y por las bendiciones que a diario recibo de su parte.

A mi padre Jorge, gracias por inculcarme valores, por guiarme, consentirme, y velar siempre porque sea una persona de bien.

A mi madre Mary, a quien admiro demasiado, gracias por mi computadora que fue la principal herramienta para culminar mi carrera, por nunca dudar de mi potencial y al contrario apoyarme en cada una de mis decisiones. Gracias también por estar junto a mí las noches de desvelo, por tener las palabras correctas en los momentos difíciles. En fin, por ser la mejor madre que la vida me pudo otorgar.

A mis hermanas Belén y Aurora que son un rayo de luz en mi vida, gracias por contagiarme su alegría y esperanza, han confiado en mí y me han tomado siempre como su ejemplo.

A la Dra. María Hallo directora del proyecto integrador, quien supo guiarnos con dedicación y paciencia para culminar con éxito nuestra tesis.

A mi compañero de tesis y amigo de toda la carrera Washito, gracias por compartir este último trabajo, gracias por acompañarme en las buenas y malas y hacer de esta experiencia universitaria algo mejor.

A una persona muy especial en mi vida, Jorge Luis gracias por tu apoyo, confianza y por estar junto a mí en todo momento.

Karen Morocho

DEDICATORIA

A mis padres por ser el mejor ejemplo de nobleza, sacrificio, perseverancia. En especial a mi madre Mary, sin su apoyo y compañía ningún triunfo hubiese sido posible.

Continuaré luchando por mis sueños para seguir siendo su orgullo y espero me alcance la vida para retribuir una parte de lo que han hecho por mí.

Karen Morocho

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

GLOSARIO DE TÉRMINOS

Registro de Transacciones. – Es un registro de información reportado por una aplicación, programa, sistema operativo, servidor, etc., que contiene sus acontecimientos cronológicos, problemas, errores, información de las actividades realizadas, entre otras.

Valores Atípicos. – Un valor atípico es una observación que se encuentra a una distancia anormal de otros valores en una muestra de datos.

Valor Normal. – Valor que se encuentra en el interior del conjunto de datos y sigue un comportamiento normal.

Anomalía. – Es un patrón dentro de un conjunto de datos que no sigue un comportamiento normal. Una anomalía no siempre se puede caracterizar como un ataque porque puede ser o no dañina.

Ataque. – Cualquier tipo de actividad maliciosa que intente recopilar, interrumpir, denegar, degradar o destruir los recursos del sistema de información o la información en sí.

Vulnerabilidad. – Es una debilidad aprovechada por los ciberdelincuentes para obtener acceso a un sistema de información sin autorización.

Normalizar. – Es una técnica aplicada a un conjunto de datos para disminuir la redundancia. Su objetivo principal es presentar todos los datos en una única forma (número para todos los casos).

Modelado. – Es un grupo de procesos en los que se combinan y analizan múltiples conjuntos de datos para descubrir relaciones o patrones. El objetivo del modelado de datos es utilizar datos pasados para informar los esfuerzos futuros.

Algoritmo. – Es un procedimiento paso a paso para resolver un problema o lograr algún fin.

Prototipo. – Es una primera representación y una versión incompleta del software a desarrollarse que simula algunos aspectos del producto final.

Tokenización. – Es el proceso de reemplazar datos confidenciales con símbolos de identificación únicos que retienen toda la información esencial sobre los datos sin comprometer su seguridad.

RESUMEN

En cualquier organización empresarial, las infraestructuras de bases de datos y de almacenamiento de la información están sujetas a diversos ataques por parte de sus usuarios para sacar provecho de esa información confidencial.

En el Centro de Respuesta a Incidentes de Seguridad Informática de la Escuela Politécnica Nacional (CSIRT-EPN) se ha detectado vulnerabilidad en los servidores de bases de datos y evidencias de filtración de información, principalmente con ataques de inyección de código SQL (Lenguaje de Consulta Estructurado).

Este proyecto de minería de datos permite detectar y prevenir ataques de inyección de código SQL mediante el análisis de registros de transacciones (*logs)*. El proyecto ha sido desarrollado utilizando la metodología CRISP-DM, considerada como una de las más recomendadas para el desarrollo de un proyecto de minería de datos. El algoritmo utilizado para la detección fue el KNN (Algoritmo de vecinos más cercanos), empleando como fuente principal de información los registros de transacciones del servidor PostgreSQL del CSIRT-EPN. Se realizó una evaluación del rendimiento del modelo y los resultados fueron muy satisfactorios en los diferentes escenarios que establecimos utilizando más de un millón de registros de transacciones.

Para el sistema de visualización de datos se implementó una interfaz que muestra la estadística diaria de los ataques encontrados, alertas de ataques y además se implementó la herramienta SQLMAP para la predicción de los mismos. Sobre esta herramienta se realizó una evaluación de funcionalidad y usabilidad, donde los resultados positivos están por encima del 80%, indicando que el prototipo cumple con los objetivos.

Palabras clave: Registros de Transacciones, Ataques a Bases Datos, Anomalías, CRISP-DM, IDS.

ABSTRACT

In any business organization, database and information storage infrastructures are subject to various attacks by their users to take advantage of that confidential information.

In the Computer Security Incident Response Center of the National Polytechnic School (CSIRT-EPN) vulnerability has been detected in the database servers and evidence of information leakage, mainly with SQL code injection attacks (Query Language Structured).

This data mining project allows detecting and preventing SQL injection attacks by analyzing logs. The project has been developed using the CRISP-DM methodology, considered one of the most recommended for the development of a data mining project. The algorithm used for the detection was the KNN (Closest Neighbors Algorithm), using as the main source of information the transaction log record of the PostgreSQL server of the CSIRT-EPN. An evaluation of the performance of the model was carried out and the results were very satisfactory in the different scenarios that we established using more than a million logs.

For the data visualization system, an interface was implemented that shows the daily statistics of the attacks found, alerts of attacks and also the SQLMAP tool was implemented for their prediction. An evaluation of functionality and usability was carried out on this tool, where the positive results are above 80%, indicating that the prototype meets the objectives.

Keywords: Logs, Database Attacks, Anomalies, CRISP-DM, IDS.

CAPÍTULO I: INTRODUCCIÓN

En cualquier organización empresarial, las infraestructuras de las bases de datos empresariales y de almacenamiento de información, contienen datos de tipo confidencial en su mayoría. Estos datos están sujetos a distintos ataques a su seguridad por parte de los usuarios de dichas bases de datos, buscando tomar ventaja de esa información con diferentes fines (Malik & Patel, 2016).

Los ataques más comunes se dan por la amplia gama de vulnerabilidades de las bases de datos; se deben principalmente a la falta de visibilidad en el momento que ocurren a nivel de base de datos y al escaso análisis de vulnerabilidades. Un ejemplo de ataques internos es la gestión inadecuada de derechos de usuarios y la falta de monitoreo para accesos no autorizados (Telefónica Company, 2015).

Las aplicaciones modernas de detección de vulnerabilidades de seguridad deben ser confiables, útiles y fáciles de administrar. En los últimos años, los sistemas de detección de intrusos (IDS) basados en la minería de datos han demostrado una alta precisión, una buena generalización de los nuevos tipos de intrusión y un comportamiento robusto en un entorno cambiante (Chetan & Ashoka, 2012).

En el escenario de evolución de software malicioso (*malware*) y distintas amenazas, los centros de respuesta a incidentes de seguridad (CSIRT por las siglas en inglés de Computer Security Incident Response Team) están cobrando relevancia. El Centro de Respuesta a Incidentes de Seguridad Informática de la Escuela Politécnica Nacional (CSIRT-EPN), trabaja de manera colaborativa con CSIRT´s nacionales e internacionales y tiene como finalidad brindar servicios de seguridad a las infraestructuras críticas de TI (Tecnologías de la Información) de la Institución, contribuyendo así al desarrollo de la ciberseguridad del Ecuador.

El presente proyecto permite detectar y predecir ataques de inyección de código SQL a servidores de bases de datos controlados por el CSIRT-EPN, haciendo uso de los registros de transacciones generados por el servidor gestor de base de datos PostgreSQL. En este documento se detalla las etapas implementadas para el desarrollo del proyecto de minería de datos que incluye el desarrollo de un prototipo de sistema para la visualización de los resultados. Los resultados muestran una alta tasa de detección de ataques con una técnica basada en un enfoque diferente, el cual es el análisis de los registros de transacciones.

1.1. OBJETIVOS

1.1.1. Objetivo General

- Desarrollar un prototipo de sistema que permita identificar ataques por inyección de código SQL y analizar información relevante para remediar, predecir, y prevenir actuales y futuros ataques a sistemas de bases de datos a partir de información de registros de transacciones generados en los sistemas DBMS utilizando técnicas de minería de datos.

1.1.2. Objetivos Específicos

- Realizar un estudio de trabajos relacionados con la identificación y predicción de ataques a sistemas de bases de datos, con el fin de conocer los avances y limitaciones de estos, y su aplicabilidad al presente estudio.
- Identificar técnicas y herramientas que permitan realizar la minería de datos para la predicción de ataques a bases de datos.
- Generar e integrar información de registros de transacciones de sistemas de bases de datos, proporcionados por el CSIRT, identificando los parámetros que serán utilizados en el modelo.
- Determinar el modelo de minería de datos para la implementación del prototipo.
- Desarrollar el prototipo de sistema que permita identificar y predecir posibles ataques a sistemas de bases de datos manejados por el CSIRT.

1.2. ALCANCE

El presente proyecto de minería de datos apoya la detección y predicción de amenazas por inyección de código SQL a servidores de bases de datos controlados por el Centro de Respuesta a Incidentes de Seguridad Informática (CSIRT) de la Escuela Politécnica Nacional (EPN), implementando un modelo y desarrollando un prototipo de visualización de datos, el cual se probó utilizando registros de transacciones de la base de datos PostgreSQL del CSIRT-EPN.

1.3. MARCO TEÓRICO

1.3.1 Sistemas de detección de intrusiones y su clasificación

Los sistemas de detección de intrusiones (IDS), son sistemas de software o hardware que automatizan el proceso de monitoreo de los eventos que ocurren en un sistema informático o red, analizándolos por signos de problemas de seguridad (Bace & Mell, 2001). Existen

diversas formas de clasificar un sistema de detección de intrusiones, pero se han sintetizado las más comunes en la siguiente Figura 1.

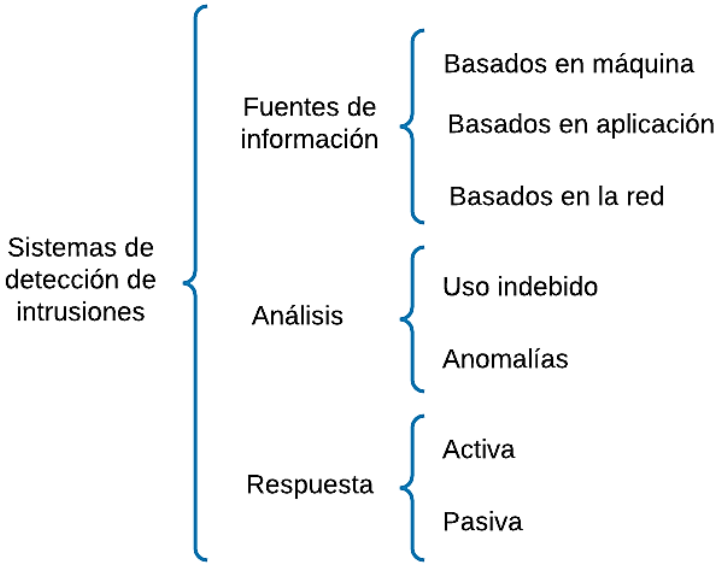


Figura 1. Clasificación de los Sistemas de Detección de Intrusos (IDS)
Fuente: Modelo de Detección de Intrusos Usando Técnicas de Aprendizaje de Máquina (Morales & Castaño, 2018)

1.3.1.1. Sistemas basados en máquina

Los IDS basados en máquina tiene como función manejar distancias, de tal forma que trabaja bajo un proceso denominado segundo plano *(background)*, que analiza la máquina periódicamente con la posibilidad de encontrar factores importantes que indiquen que el sistema se encuentra amenazado. El IDS basado en máquina alerta y toma las medidas necesarias para proteger el sistema (Solarte Martinez et al., 2017).

1.3.1.2. Sistemas basados en aplicaciones

Los IDS basados en aplicaciones analizan los eventos que ocurren dentro de una aplicación de software. Las fuentes de información más comunes utilizadas por los IDS basados en aplicaciones son los archivos de registro de transacciones de la aplicación (Bace & Mell, 2001).

1.3.1.3. Sistemas basados en la red

Los IDS basados en la red monitorean los paquetes que circulan por la red en busca de elementos que indiquen un ataque contra alguno de los sistemas ubicados en ella; el IDS basado en la red puede situarse en cualquiera de las terminales o en un elemento que analice todo el tráfico, como un enrutador de conexión a la red (Solarte Martinez et al., 2017).

1.3.1.4. Sistemas de detección de uso indebido

Los IDS de uso indebido monitorizan las actividades que ocurren en un sistema y las compara con una serie de firmas de ataques previamente almacenadas en una base de datos. Cuando se monitorizan actividades que coinciden con las firmas se genera una alarma. Este tipo de análisis se atiene al conocimiento previo de las secuencias y actividades que forman un ataque. Se detectan las tentativas de explotación de vulnerabilidades conocidas o patrones de ataques típicos (Montañana Pérez et al., 2003).

1.3.1.5. Sistemas de detección de anomalías

Los IDS detección de anomalías suponen que los ataques son diferentes de la actividad "normal" (legítima) y, por lo tanto, puede ser detectada por sistemas que identifican estas diferencias. Los IDS de detección de anomalías construyen perfiles que representen el comportamiento normal de usuarios, servidores o conexiones de red, a partir de datos históricos recopilados durante un período de funcionamiento normal. Luego, los detectores recopilan los datos de eventos y utilizan una variedad de medidas para determinar cuándo la actividad monitoreada se desvía de la norma (Bace & Mell, 2001).

1.3.1.6. Respuesta activa

Los IDS de respuesta activa reaccionan de forma automática ante los intentos de intrusión. Un sensor puede aumentar su capacidad para recolectar información que le permita dar seguimiento a cualquier evento sospechoso (Párrizas, 2005).

1.3.1.7. Respuesta pasiva

Los IDS de respuesta pasiva tienen como objetivo informar a los usuarios o al responsable de seguridad del sistema cualquier acción sospechosa o ataque del que se encuentre alguna evidencia, confiando en que los humanos tomen medidas posteriores basadas en esa información (Bace & Mell, 2001).

1.3.2. Sistema de detección de Intrusos en Base de Datos

Un sistema de detección de intrusiones en la base de datos (DIDS), se utiliza para detectar posibles violaciones en la seguridad de la base de datos. DIDS sigue otros mecanismos tradicionales de seguridad de bases de datos y mecanismos de seguridad de red, como el firewall y la detección de intrusos en la red. Por lo tanto, se enfrenta a la intrusión de usuarios internos o la intrusión que puede pasar a través de otras capas de seguridad. Esto significa que la cantidad de eventos de intrusión es rara en comparación con la cantidad de eventos normales (Nandasana & Barot, 2017).

1.3.2.1. Detecciones de intrusos por anomalías

Los IDS basados en anomalías analizan los comportamientos no usuales que existan en el modelado del comportamiento normal de usuarios, grupo de usuarios, accesos remotos, etc. Cualquier desviación de este comportamiento es considerada como una transacción sospechosa. Como ejemplo tenemos un sistema de detección encargado de monitorizar las máquinas origen desde las que un usuario sospechoso se conecta a nuestro sistema: si se utiliza un modelo de detección de intrusos por anomalías, tendríamos un listado con dos o tres direcciones más utilizadas por el usuario legítimo, alertando al responsable de seguridad en caso de que el usuario acceda desde otro lugar.

1.3.2.2. Detección de abuso de privilegios

Los IDS basados en abuso de privilegios detecta que las transacciones son sospechosas comparándolas con un conjunto de patrones de intrusión previamente establecidos, conocidos como entrenados de actividad normal.

1.3.3. Fuentes de inseguridad

Los incidentes de seguridad se originan en una o una combinación de las siguientes fuentes:

1.3.3.1. Fuente externa

Las amenazas externas se originan en fuentes ajenas a la organización. Los ejemplos incluyen piratas informáticos, grupos del crimen organizado y entidades gubernamentales, así como eventos ambientales, etc. Típicamente, no hay confianza o privilegio implícito para entidades externas (BID & OEA, 2020).

1.3.3.2. Fuente interna

Las amenazas internas son las que se originan dentro de la organización. Esto abarca a los activos humanos: ejecutivos, empleados y pasantes pertenecientes a la empresa. La mayoría de información privilegiada es confiada a expertos de altos rango. Los administradores de TI en particular tienen altos niveles de acceso y privilegios. (Mattsson, 2011).

1.3.3.3. Fuente socio comercial

Los socios son cualquier tercero que comparta una relación comercial con la organización. Esta cadena de valor de socios, vendedores, proveedores, contratistas y clientes se conoce como la empresa extendida. El intercambio de información es necesario para la empresa extendida y, por esta razón, generalmente se implica cierto nivel de confianza y privilegio entre los socios comerciales (Al-Sayid & Aldlaeen, 2013).

El Informe de Investigaciones de Violación de Datos (DBIR) en 2019 informó que: el 69% fue resultado de fuentes externas, el 34% fueron causados por personas con información privilegiada, el 2% fueron socios comerciales implicados y el 5% involucraron a múltiples partes (Verizon Business, 2019b).

1.3.4. Ataques a bases de datos

Las bases de datos actuales se encuentran expuestas a diferentes tipos de ataques. Antes de abordar las técnicas para proteger las bases de datos, es conveniente describir los ataques posibles (Malik & Patel, 2016). Los principales ataques a las bases de datos se pueden clasificar como se muestra en la Figura 2. Estos se detallan en las siguientes secciones.

Figura 2. Tipos de Ataques a las Bases de Datos
Fuente: Seguridad de las Bases de Datos: Ataques y Métodos de Control (Malik & Patel, 2016)

1.3.4.1. Exceso de privilegios

El exceso de privilegios se refiere al uso malicioso o inapropiado de los privilegios existentes en el manejo de los datos dentro de una organización. Usualmente el exceso de privilegios se da variedades tales como:

- Abuso de privilegios
- Mal manejo de la información
- Ejecución de trabajo no autorizado
- Mal uso del correo electrónico
- Hardware no autorizado
- Software no autorizado

Los motivos predominantes son de naturaleza financiera. Son comunes las tomas de datos confidenciales por parte de los empleados para proporcionar una ventaja ilegal (Verizon Business, 2019a).

1.3.4.2. Inyección de código SQL

El ataque por inyección de código SQL inserta sentencias de consulta SQL (Lenguaje de consultas estructuradas o por sus siglas en inglés Structured Query Language) dentro de una consulta antes establecida, logrando manipular de diversas formas los procesos lícitos de una aplicación. Esta inyección de código SQL se puede clasificar en el grupo de los denominados "Ataques o Vulnerabilidades del Control de Entrada (Racciatti, 2002). La vulnerabilidad de este ataque se puede atribuir a la práctica de programación inapropiada por parte de los desarrolladores del sitio web, dejando muchas puertas abiertas para que los atacantes las exploten y obtengan acceso a información confidencial existente en las bases de datos del servidor del sitio web (Ali et al., 2011).

1.3.4.3. Taxonomía de los ataques de Inyección de código SQL

La Figura 3 muestra la clasificación de las categorías de Inyección de código SQL existentes.

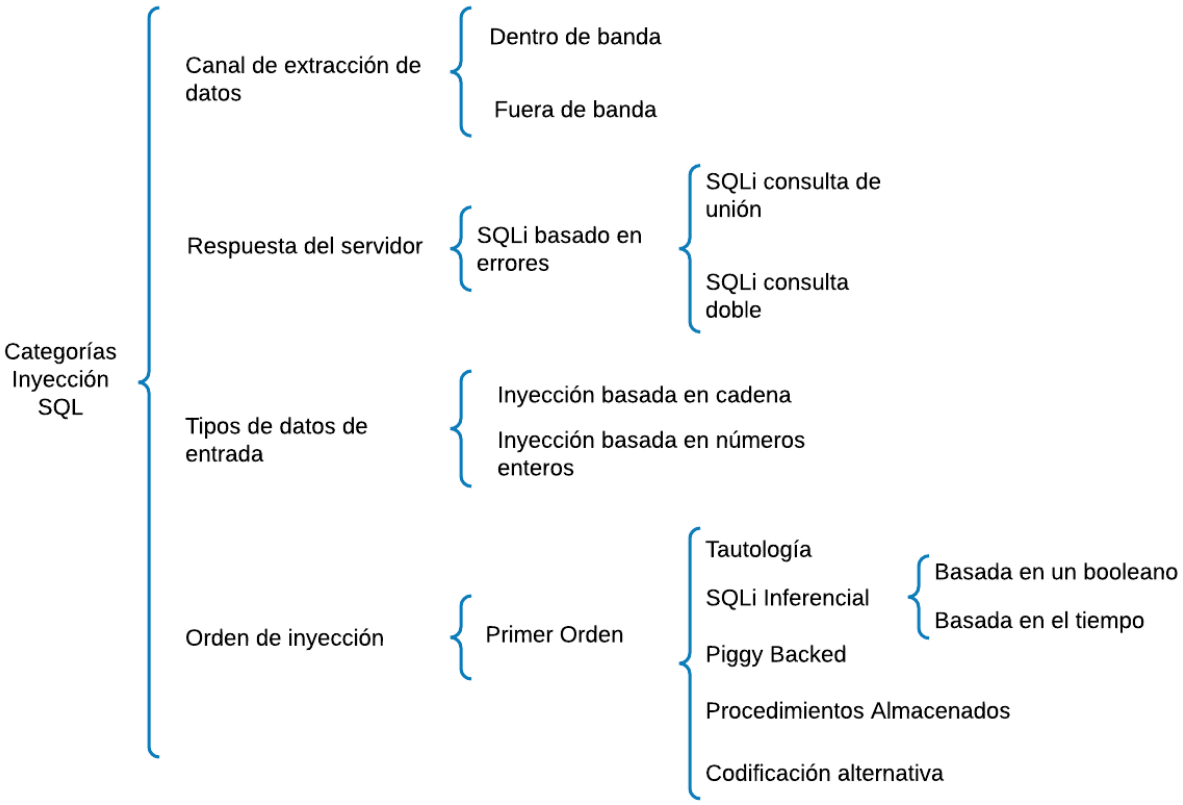


Figura 3. Taxonomía de las Inyecciones de Código SQL
Fuente: Mitigar los Ataques de Inyección de SQL a Través del Modelado de Amenazas Híbridas (Omotunde & Ibrahim, 2016)

a) Inyección de código SQL basada en tautologías

La inyección de código SQL basada tautologías se utiliza para evitar la autenticación en una página web, pues ignora la cláusula WHERE. La tautología más conocida es la expresión: “1 = 1”. El mecanismo consiste en poner cualquier condición y concatenarla con los criterios "OR" y "1 = 1 (que siempre es verdadero)” para que el resultado de toda la condición sea verdadero.

```
SELECT * FROM ACCOUNTS WHERE USERNAME = 'ADMINISTRATOR' AND PASSWORD
= '' OR 1='1'
```

b) Inyección de código SQL basada en consultas ilegales

El objetivo del ataque es recopilar información importante sobre el esquema y la estructura de tablas y campos dentro de la base de datos. 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 (Sadeghian et al., 2013). Un ejemplo es la adición de la comilla dentro de la cadena de texto suministrada desde el parámetro de entrada (John et al., 2019).

```
SELECT * FROM TblSupplier WHERE NameSupplier = 'O'Reill
```

c) Inyección de código SQL basada en consultas de unión

La inyección de código SQL por consulta de unión se realiza mediante el cambio de lógica, es decir que es posible modificar el conjunto de resultados de la estructura de una consulta original establecida por el sistema. En un parámetro de entrada vulnerable se adiciona una segunda consulta mediante el operador “unión”, con esta adición es posible recuperar los resultados de diferentes tablas las cuales son el resultado de la unión de la consulta original y de la segunda consulta adicionada (John et al., 2019).

```
SELECT * FROM TblSupplier WHERE NameSupplier = '' UNION ALL
SELECT * From TblConsumer WHERE 1 =1
```

d) Inyección de código SQL basada en consultas adicionales

La inyección de código SQL por consultas adicionales 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 (John et al., 2019).

```
SELECT * FROM TblUser WHERE UserName = "; DROP TABLE TblUser AND
pass ="
```

e) Inyección de código SQL basada en inferencia

En un ataque SQL inferencial, no se transfieren datos a través de la aplicación web y el atacante no podría ver el resultado de un ataque en banda (razón por la cual estos ataques se denominan comúnmente "ataques de inyección de código SQL ciega") (John et al., 2019). En cambio, un atacante puede reconstruir la estructura de la base de datos enviando cargas útiles, observando la respuesta de la aplicación web y del servidor.

- **Inyección de código SQL basada en un booleano**

La inyección de código SQL basada en un booleano permite concluir el comportamiento de la página cuando se envía una pregunta (verdadero/falso) al servidor. Si la respuesta es verdadera, la aplicación continúa con su funcionamiento normal. Caso contrario, el funcionamiento de la página difiere del habitual. 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 la velocidad de respuesta es posible determinar la rama seguida por la consulta e inferir en la respuesta (John et al., 2019).

```
DECLARE @s varchar (8000); SELECT @s = db_name ();
IF (ascii (SUBSTRING (@s, 1, 1)) & (power (2, 0))) > 0 waitfor
 delay '0:0:5'
```

- **Inyección de código SQL basada en el tiempo**

La inyección de código SQL basada en el tiempo es una técnica de inyección de código SQL inferencial que se basa en enviar una consulta SQL a la base de datos, lo que obliga a la base de datos a esperar una cantidad de tiempo específica (en segundos) antes de responder. El tiempo de respuesta le indicará al atacante si el resultado de la consulta es VERDADERO o FALSO. Dependiendo del resultado, se devolverá una respuesta HTTP con un retraso o se devolverá inmediatamente (John et al., 2019).

```
SELECT * FROM gestion.ges_catalogo WHERE cat_id=1-SLEEP(15);
```

f) Inyección de código SQL basada en procedimientos almacenados

El atacante integra códigos de inyección SQL maliciosos en los procedimientos almacenados existentes en las bases de datos. Los 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 (John et al., 2019)

```
SELECT * FROM news WHERE news_cat = 'sport'; SHUTDOWN;
```

g) Inyección de código SQL basada en codificación alternativa

La inyección de código SQL basada en codificación alternativa funciona al confundir la base de datos con diferentes codificaciones en las sentencias SQL. Por ejemplo, los atacantes podrían usar hexadecimal, ASCII o Unicode en una declaración SQL. Al hacer esto, los atacantes evitarán cualquier validación básica realizada por la aplicación. En consecuencia, los comandos SQL podrían codificarse en hexadecimal y la validación no los detectaría, lo que significa que el servidor ejecutaría cualquier comando que el atacante quiera que ejecute (John et al., 2019).

```
SELECT usr_id FROM gestion.ges_usuario
WHERE usr_nombre= 'legalUser';
exec(CHAR(0x73687574646f776e)) -- AND usr_password=''
```

1.3.4.4. Código malicioso

El código malicioso es el nombre colectivo de una serie de variantes de software malicioso, incluidos virus, software espía (*spyware)* y de robo de información *(ransomware)*. Consiste en código desarrollado por ciber atacantes, diseñado para causar daños importantes a los datos para obtener acceso no autorizado a una red (Malik & Patel, 2016).

El código malicioso se puede aprovechar de numerosas maneras para establecer o avanzar ataques. El comando y control (C2) y las puertas traseras se encuentran tanto en incidentes de seguridad como en infracciones (Verizon Business, 2019). El ransomware sigue siendo un problema importante para las organizaciones y no está obligado a depender del robo de datos para ser lucrativo.

1.3.4.5. Auditoria débil

Auditoría débil es el registro automatizado de transacciones de bases de datos que involucran datos confidenciales, debe ser parte de cualquier implementación de base de datos. El hecho de no recopilar registros detallados de auditoría de la actividad de la base de datos representa un riesgo organizacional serio en muchos niveles (Shulman, 2006).

Un seguimiento de auditoría adecuada debe recopilar y archivar registros detallados de los datos almacenados en sus bases de datos, particularmente aquellos que almacenan datos confidenciales como registros financieros o de salud. El error que comete la mayoría de las organizaciones es asumir que sus pistas de auditoría integradas son suficientes para ayudarlas a cumplir y garantizar su seguridad.

1.3.4.6. Exposición de copia de respaldo

Los medios de almacenamiento de respaldo a menudo están completamente desprotegidos contra ataques. Como resultado, numerosas violaciones de seguridad han implicado el robo de discos y cintas de respaldo de bases de datos (Shulman, 2006). Además, no auditar y monitorear las actividades de los administradores que tienen acceso de bajo nivel a información confidencial puede poner en riesgo sus datos.

1.3.4.7. Autenticación débil

La autenticación débil tiene muchas facetas, desde la fuerza bruta utilizada desde la interfaz de usuario hasta el almacenamiento inseguro de las credenciales de la base de datos utilizadas para una aplicación.

1.3.4.8. Vulnerabilidades y mala configuración de la base de datos

Es común encontrar bases de datos vulnerables y sin parches, o descubrir bases de datos que todavía tienen cuentas y parámetros de configuración que ya vienen predeterminados. Estas vulnerabilidades son conocidas por varios atacantes que saben cómo penetrar una organización por esta falta de seguridad y la utilizan a su favor (Malik & Patel, 2016).

1.3.4.9. Datos sensibles sin manejo

Muchas compañías luchan por mantener un inventario preciso de sus bases de datos y los objetos de datos críticos contenidos en ellas. Las bases de datos olvidadas pueden contener información confidencial y pueden surgir nuevas bases de datos, por ejemplo, en entornos de prueba de aplicaciones, sin visibilidad para el equipo de seguridad.

1.3.4.10. Denegación de servicio

La denegación de servicio comprende ataques basados en el consumo de recursos, como el envío repetido de consultas de búsqueda complejas hasta agotar los recursos del servidor (Pill, 2019).

1.4. TRABAJOS RELACIONADOS

La detección y prevención de ataques de inyección de código SQL son temas de investigación activa en el mundo académico y la industria. Para lograr estos propósitos se implementaron herramientas automáticas y sistema de seguridad, pero ninguno de ellos fue lo suficientemente completo o preciso para garantizar un nivel absoluto de seguridad de las aplicaciones web (Varshney & Ujjwal, 2019). Sin embargo, el análisis de seguridad basado en registros de transacciones es un campo poco explorado. A continuación, en la Tabla 1, se muestra un cuadro con los estudios más relevantes con respecto a la temática.

Tabla 1. Especificación de trabajos relacionados y su aplicabilidad al presente estudio

1.4.1. Identificación de Ataques

La detección es uno de los primeros pasos para la prevención de ataques a sistemas de base de datos. Para cumplir con el primer objetivo de búsqueda bibliográfica, se encontraron los siguientes estudios.

Título del estudio	Idea principal	Tipos de inyección de código SQL	Aporte al trabajo desarrollado
Uso de tokenización en consultas SQL (Ntagwabira & Kang, 2010)	Este estudio implica tokenizar una consulta inicial y otra que tiene una inyección de código SQL. Se comparan las dos matrices obtenidas y se detecta un ataque si sus longitudes difieren, si sus longitudes son iguales no hay inyección SQL.	Si detecta inyecciones de código SQL, pero no especifica cuáles.	Análisis de texto en las consultas de los registros de transacción para identificar si en la cadena de texto existen indicios de un ataque. Los indicios de ataques serían los espacios, comillas simples y/o guiones dobles.
Método de coincidencia de cadenas y subsecuencia común (Anitha et al., 2015)	Este estudio propone dos métodos: 1. El método de fuerza bruta convierte la entrada del usuario en una matriz de caracteres y luego las compara con el patrón predefinido. Si el patrón es una subcadena de la entrada del usuario, devuelve la posición del patrón en la entrada del usuario, de lo contrario retorna un error. 2. El segundo método encuentra la subsecuencia común más larga entre 'x' y 'y', luego juzga si las acciones maliciosas contenidas en la cadena de consulta y luego devuelven el carácter de subsecuencia.	Tautología Consulta Unión Consultas Adicionales	Uso de algoritmos de clasificación para implementar un método de coincidencia de cadenas, donde se analiza caracteres comunes usados en ataques de inyección de código SQL.

Anomalías basadas en datos de tráfico de red (Vartouni et al., 2018)	Este estudio utiliza el modelo basado en caracteres para construir características a partir de datos HTTP. El tamaño de las características aumenta exponencialmente debido al uso de los datos recopilados y luego se aplica el codificador para resolver esto e identificar las anomalías.	No se especifica	Uso del algoritmo Isolation Forest para identificar anomalías en registros de transacciones web. El tipo de consulta y/o errores son un parámetro importante para la identificación de anomalías.
Sistema DetAnom (Bossi et al., 2017)	En este estudio se plantea que se debe crear y enviar una firma y un conjunto de restricciones con cada consulta enviada. Los autores afirman que los rastros de inyecciones de código SQL se pueden detectar observando los registros de errores.	Si detecta inyecciones de código SQL, pero no especifica cuáles.	Análisis de anomalías en base a las consultas realizadas por los usuarios. La creación de perfiles de usuario con diferentes permisos ayuda al análisis de anomalías.
1.4.2. Predicción y Prevención de Ataques La predicción y prevención es la continuidad de los resultados de la identificación, a continuación, resaltamos los estudios más relacionados con este proyecto.			
Modelo de aprendizaje profundo (Gong et al., 2019)	Este estudio se basa en dos partes: Una red neuronal convolucional (CNN) como extractor de características y un modelo bayesiano como clasificador. Este enfoque permite analizar los ataques desde más aspectos, en el pre procesamiento, cada carácter de los datos de entrada se ha transformado en un código ASCII, luego se incrusta en un vector. Después de que CNN extrae las características, se utiliza como entrada para el clasificador.	No se especifica	Ejemplo de implementación de un sistema de detección de ataques SQL realizados en la web, mediante el aprendizaje automático para la predicción.
Técnica de inserción inversa e(RIT) (Raj & Sherly, 2018)	En este algoritmo, la entrada se invirtió primero y se organizó en un grupo de dos caracteres y se insertó un		Uso del algoritmo de inserción inversa para probar, desde un sitio web, ataques de

	carácter especial antes de almacenarlo en la base de datos. En el momento del inicio de sesión, la entrada del usuario experimentó la misma transformación y se comparó con el código almacenado en la base de datos.	Si detecta inyecciones de código SQL, pero no especifica cuáles.	inyecciones de código SQL. Esta es una forma simple de encontrar brechas de seguridad para poder prevenir ataques en el futuro.
Análisis de vulnerabilidad de sistemas de gestión de contenido a la inyección de SQL mediante SQLMAP(Ojagbule et al., 2018)	Este estudio plantea una solución que predice incidencias futuras desconocidas de ataques de inyección de código SQL basado en expectativas. Esto se logró calculando la probabilidad de ocurrencia del ataque en el conjunto de datos de caracteres especiales y el conjunto de datos típico respectivamente, y en la segunda fase se detecta el ataque de inyección de código SQL mediante SQLMAP.	Detecta los siguientes ataques de inyección de código SQL. Inferencial Booleano Inferencial Tiempo Basado en errores Consultas Adicionales Operador Unión	Los ataques de inyección pueden prevenirse con pruebas, para lo cual la herramienta de SQLMAP permite el análisis de bases, tablas y datos. Información que pude ser usada para la prevención de ataques en el futuro.
1.4.3. Estudios Aplicados en el Proyecto			
Un método de detección de anomalías basado en registros con búsqueda eficiente de vecinos y selección automática de vecinos K (Wang et al., 2020)	Este estudio destaca el concepto de valor atípico, el cual se deriva de la minería de datos y consiste en la observación del comportamiento que se desvía del comportamiento normal de los demás, creando así un límite sospechoso sobre sí mismo al ser generado por un mecanismo distinto. Con la detección de valores atípicos se puede encontrar transacciones anormales de la base de datos y al ser tratadas mejora la seguridad de esta.	Detecta los siguientes ataques de inyección de código SQL. Tautologías Consultas Ilegales Operador Unión Consultas Adicionales Inferenciales basados en booleanos Inferenciales basados en el tiempo Procedimientos Almacenados Codificación Alternativa	Referencia principal del uso de KNN para el análisis de anomalías, haciendo uso de los registros de transacciones.
Papel del análisis de valores atípicos en la detección de intrusiones en bases de datos (Brahma & Panigrahi, 2020)	Este estudio destaca el análisis de valores atípicos en la minería de datos para automatizar el proceso de detección de intrusiones con mayor precisión.		Referencia principal que establece las técnicas de minería de datos para el análisis de intrusiones con mayor precisión.

La recopilación de estudios realizada muestra dos aspectos importantes:

- La identificación de rastros de ataques de inyección de código SQL pueden ser recopilados desde el sistema gestor de base de datos.
- La prevención y predicción de ataques de inyección de código SQL es más eficiente cuando las técnicas y herramientas son probadas y evaluadas desde un cliente del servidor web.

Con estos principios definidos, hemos basado este proyecto en los estudios "A Log-Based Anomaly Detection Method with Efficient Neighbor Searching and Automatic K Neighbor Selection – Un método de detección de anomalías basado en registros con búsqueda eficiente de vecinos y selección automática de vecinos K (Wang et al., 2020)" y "Role of Soft Outlier Analysis in Database Intrusion Detection – Papel del análisis de valores atípicos en la detección de intrusiones en bases de datos (Brahma & Panigrahi, 2020)", donde se destaca el análisis de valores atípicos en la minería de datos para automatizar el proceso de detección de intrusiones con mayor precisión.

El concepto de valor atípico se deriva de la minería de datos y consiste en la observación del comportamiento que se desvía del comportamiento normal de los demás, creando así un límite sospechoso sobre sí mismo al ser generado por un mecanismo distinto. Con la detección de valores atípicos se puede encontrar transacciones anormales de la base de datos y al ser tratadas mejora la seguridad de esta.

La base de este estudio es la detección de valores atípicos basada en clústeres para detectar intrusos en bases de datos. Con el análisis de conglomerados divide a los datos en grupos significativos. Esta agrupación se realiza de acuerdo a dos criterios importantes:

- El algoritmo de detección de intrusos analiza diferentes características de los datos de la red mediante el empleo de dos medidas: rareza y aislamiento.
- El algoritmo clave en esta investigación, kNN (Algoritmo de vecinos más cercanos), funciona almacenando todos los casos disponibles y clasificando los nuevos datos o casos en función de una medida de similitud.

En la Figura 4 se presenta la implementación del algoritmo kNN (Algoritmo de vecinos más cercanos) que será utilizado para el desarrollo de este proyecto integrador.

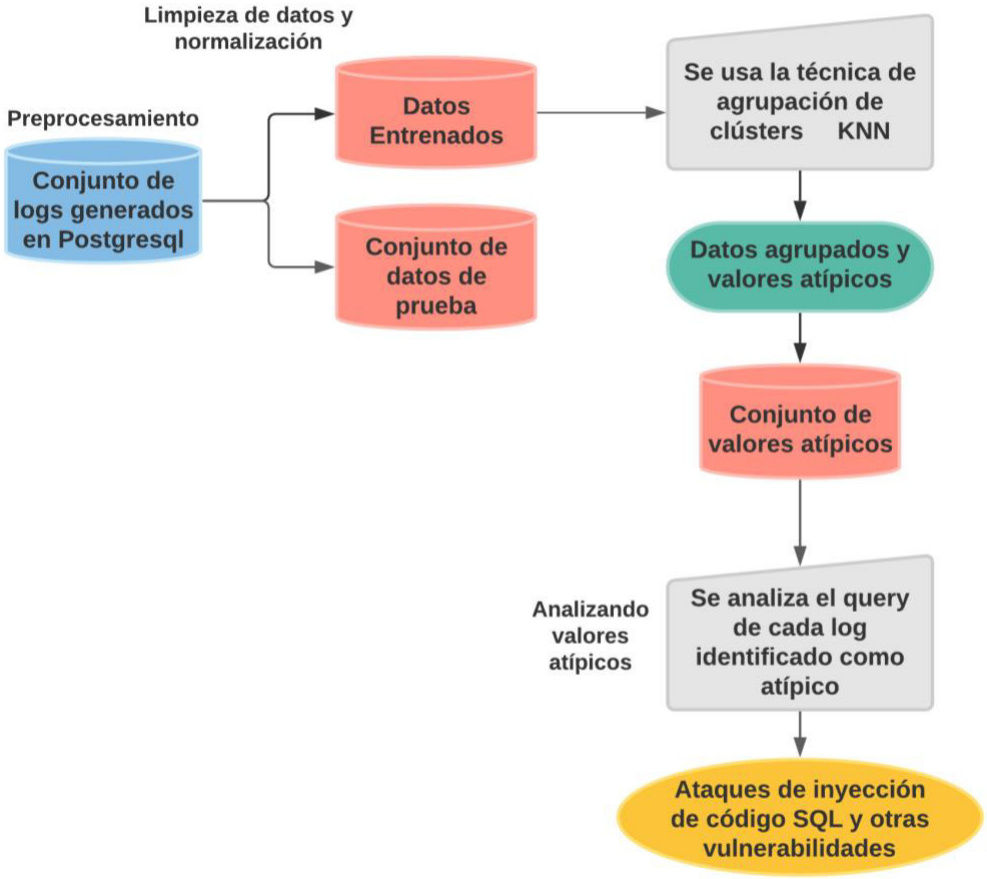


Figura 4. Implementación del algoritmo kNN

1.4.3.1. Herramientas

La Tabla 2 describe los lenguajes utilizados en el desarrollo del proyecto.

Tabla 2. Lenguajes de Programación Utilizados

Nombre	Descripción	Aplicación
Python 3.9 	Python es un lenguaje de programación potente, tiene eficientes estructuras de datos que son de alto nivel y maneja un enfoque simple para la programación orientada a objetos (Python Software Foundation, 2020).	Lenguaje utilizado en la implementación del código para el programa de detección de intrusos.
Ruby 	Ruby es un lenguaje de programación surgido en 1993 de la mano de Yukihiro Matsumoto. Fue creado por la complejidad de otros lenguajes como Java y C++. Es flexible, preciso y eficiente (Bächle & Kirchberg, 2007).	Lenguaje utilizado para la implementación del prototipo de sistema web.

La Tabla 3 describe las herramientas que se utilizan para el proyecto y su aplicación.

Tabla 3. Herramientas Utilizadas		
Nombre	**Descripción**	**Aplicación**
PostgreSQL 	PostgreSQL es un sistema de código abierto para base de datos relacionales. Entre sus características destacan la confiabilidad, robustez de funciones y el rendimiento (Grupo Global de PostgreSQL, 2020).	Simular un servidor de base de datos parecido al que maneja el CSIRT, donde se ejecutaron distintas consultas las cuales generaron registros de transacciones que presentan dos comportamientos: ataque y uso normal.
VMware Workstation 	VMware es un hipervisor de escritorio que permite ejecutar distintas máquinas virtuales en la PC, tiene compatibilidad con los Sistemas Operativos Linux o Windows (VMware Inc, 2020).	Crear una máquina virtual con el Sistema Operativo Windows, que será nuestro ambiente de trabajo con los distintos programas a utilizar.
PyOD 	Conjunto de herramientas de Python que permite la detección de objetos atípicos en datos. Cuenta con más de 30 algoritmos para la detección de anomalías (Zhao et al., 2019).	Comparar el rendimiento de diferentes algoritmos que pueden identificar valores atípicos en un conjunto de datos y visualizar los mismos en el plano.
Ruby on Rails 	‘Ruby on Rails” es un marco de aplicación web que incluye todo lo necesario para crear aplicaciones web respaldadas por bases de datos de acuerdo con el patrón Modelo-Vista-Controlador (MVC) (Bächle & Kirchberg, 2007).	Implementar una base que guarde los datos exhibidos en el prototipo de sistema.
Herocku 	Heroku es una plataforma en la nube como servicio (PaaS) basada en contenedores. (Kemp & Gyger, 2013).	Desplegar el prototipo de sistema de manera rápida y funcional.
Webpacker 	Webpacker es un paquete de módulos de Ralis que proporciona una configuración estándar de paquetes web (RailsGuides, n.d.).	Gestionar dependencias que existen entre las interfaces y la base de datos del sistema.

Bootstrap 	Bootstrap es un marco CSS gratuito y de código abierto dirigido al desarrollo web. Contiene plantillas de diseño basadas en CSS y JavaScript.(Bootstrap, 2019).	Desplegar interfaces de usuario para visualizar los resultados.
SQLMAP 	SQLMAP es una herramienta automatizada de prueba de inyección de código SQL abierto desarrollada para encontrar y explotar vulnerabilidades de SQLi. Viene con una colección de características que ayudan al usuario a especificar la profundidad del ataque y el nivel de riesgo que están dispuestos a atravesar (Charania & Vyas, 2016).	Testear y prevenir ataques de inyección de código SQL a servidores de bases de datos.

CAPÍTULO II: METODOLOGÍAS

Este proyecto se realizó utilizando la metodología CRISP-DM, y para el desarrollo del prototipo de sistema se utilizó la metodología en cascada.

2.1. METODOLOGÍA CRISP-DM

La metodología para el desarrollo de proyectos de minería de datos CRISP-DM (Cross Industry Standard Process for Data Mining) propone una serie de tareas generales que se desglosan en tareas específicas con los pasos necesarios de un proyecto minería de datos. A continuación, en la Figura 5, se describe las fases de CRISP-DM y como se utilizó cada una en el desarrollo del proyecto. Se debe recalcar que la secuencia de las fases no es rígida y permite desplazarse hacia adelante o hacia atrás entre las diferentes etapas (Nadali et al., 2011).

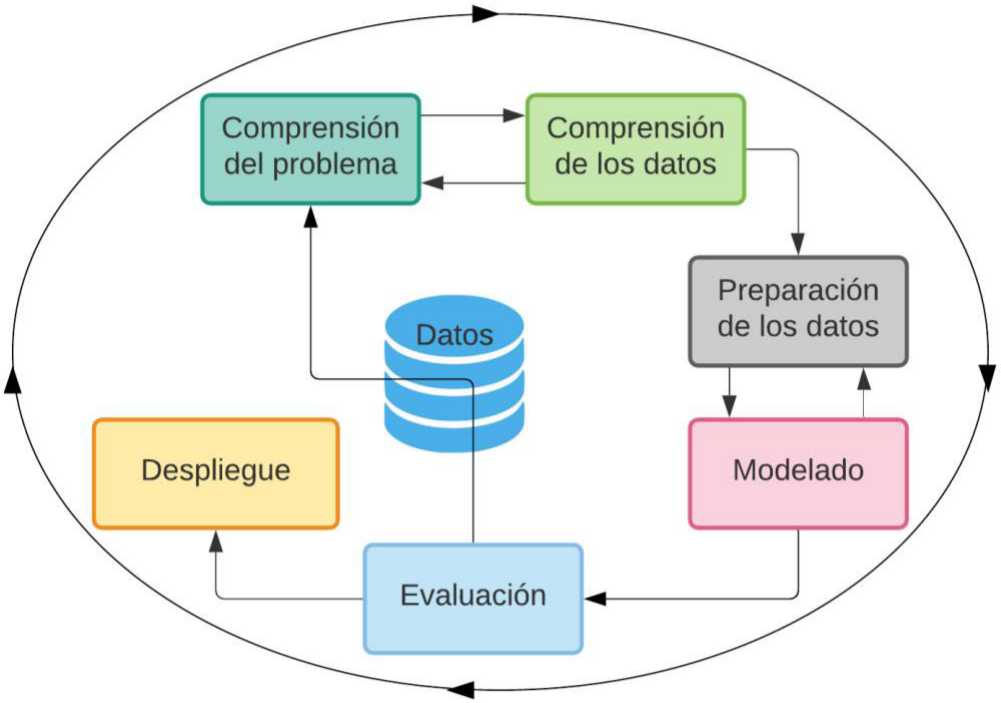


Figura 5. Etapas de la Metodología CRISP-DM
Fuente: Evaluación del nivel de éxito de los proyectos de minería de datos basados en la metodología CRISP-DM mediante un sistema difuso de expertos (Nadali et al., 2011)

2.1.1. Comprensión del Problema

Es la primera y más importante fase de la metodología CRISP-DM porque se encarga de entender los objetivos y requisitos de un proyecto desde un punto de vista comercial, luego transformarlo en un problema de minería de datos y diseñar un plan que cumpla con los objetivos planteados (Nadali et al., 2011). Es importante entender de forma clara los

objetivos y el problema del proyecto, sino con ningún algoritmo obtendremos resultados fiables.

2.1.2. Comprensión de los datos

Es la segunda fase de la metodología CRISP-DM y se ocupa de recolectar datos iniciales que sean de calidad y permitan un acercamiento con el problema. Después realiza una descripción, exploración y verificación de los datos obtenidos para descartar datos innecesarios que impidan construir un conjunto de datos finales (Nadali et al., 2011).

2.1.3. Preparación de los datos

Es la tercera fase de la metodología CRISP-DM y su finalidad es preparar, seleccionar, limpiar, integrar, normalizar los datos obtenidos, de tal manera que los datos estén acorde a un formato apropiado para la técnica de minería de datos que se utilizará en pasos posteriores (Nadali et al., 2011).

2.1.4. Modelado

Es la cuarta fase de la metodología CRISP-DM y tiene relación directa con la fase anterior porque los datos son procesados en función de la técnica de modelo seleccionada.

Esta fase se centra principalmente en elegir la técnica de modelamiento adecuada para el proyecto de minería de datos que se está desarrollando, se encarga además de tareas como la generación del plan de prueba, construcción y evaluación del modelo (Nadali et al., 2011).

2.1.5. Evaluación

Es la quinta fase de la metodología CRISP-DM y se encarga de evaluar el modelo efectuado en pasos anteriores, mide también si se cumplieron de manera correcta los objetivos del negocio y evalúa los resultados obtenidos (Nadali et al., 2011). Es importante hacer una revisión del proceso en base a los resultados obtenidos, para repetir algún paso anterior en el que posiblemente se haya cometido un error.

2.1.6. Despliegue

Es la última fase de la metodología CRISP-DM y se lleva a cabo una vez terminada la evaluación y validación del modelo para el proyecto de minería de datos. Esta fase aglutina tareas como la planificación del despliegue del modelo, generación de reportes, revisiones del proyecto y sus mejoras a futuro (Nadali et al., 2011). Para cumplir con los objetivos de este proyecto se desarrollará el prototipo de sistema, que permitirá la visualización de los resultados del modelo, utilizando la metodología de desarrollo en cascada.

2.2. METODOLOGÍA EN CASCADA

La metodología en cascada se utilizó para el desarrollo del prototipo de sistema. El término "cascada" se refiere a una de las formas tradicionales de gestión de proyectos para gestionar el desarrollo de software en proyectos pequeños y con requisitos claramente establecidos. Este es un método de planificación del desarrollo secuencial, donde las actividades se realizan paso a paso reduciendo los riesgos y las incertidumbres. Cada actividad planificada debe completarse y aprobarse antes de pasar a la siguiente, sin crear superposiciones de diferentes fases (Chandrababu & Muddangula, 2019). Una representación de cada hito puede visualizarse en la Figura 6.

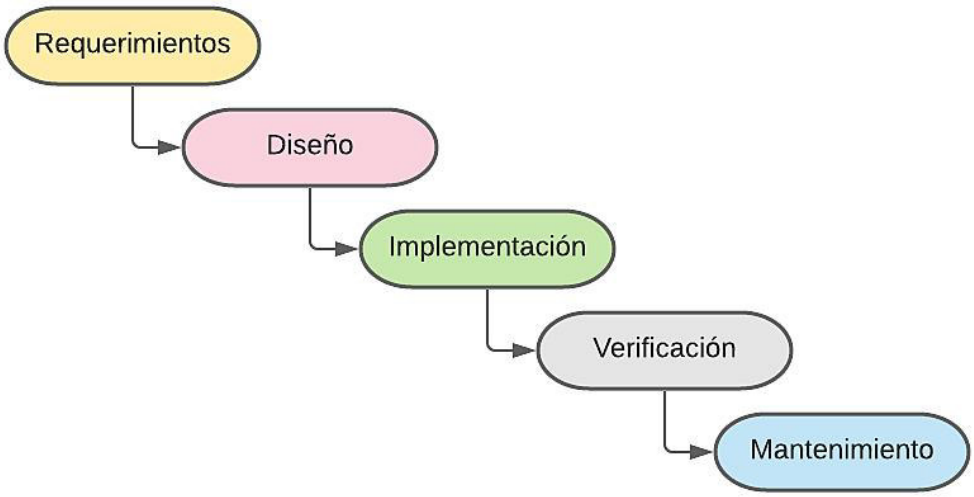


Figura 6. Etapas de la Metodología Cascada
Fuente: Cascada - Modelo de desarrollo de software (Scrum & Meade, 2010)

2.2.1. Requerimientos

En esta fase el equipo del proyecto junto con el cliente hace una lista detallada de los requisitos del sistema. El equipo del proyecto señala en detalle la funcionalidad y las limitaciones (si las hay) del software que están desarrollando. El documento que contiene toda esta información se denomina SRS (Scrum & Meade, 2010).

2.2.2. Diseño

Esta etapa se utiliza para desarrollar un concepto de resolución concreto en base a las necesidades, tareas y estrategias determinadas de antemano. Durante esta fase, los desarrolladores crean la arquitectura del software, así como un plan de construcción de software detallado y, por lo tanto, se centran en elementos concretos como interfaces, marcos o bibliotecas. (Chandrababu & Muddangula, 2019).

2.2.3. Implementación

La arquitectura del software desarrollada durante la fase de diseño se lleva a cabo durante la fase de implementación que incluye la programación del software, la resolución de problemas y las pruebas del módulo. Durante la fase de implementación, el proyecto de software se traduce al lenguaje de programación deseado. Los diferentes componentes de software se desarrollan por separado, se verifican como parte de las pruebas del módulo y se integran paso a paso en el producto general. El resultado de la fase de implementación es un producto de software que se probará por primera vez como producto global en la siguiente fase. (Scrum & Meade, 2010).

2.2.4. Verificación

Las pruebas se realizan según los pasos definidos en el plan de prueba, para asegurar que la entrada definida produzca resultados reales que concuerden con los resultados requeridos. Los autores desarrolladores del modelo generan un informe que contienen las pruebas realizadas (Scrum & Meade, 2010).

2.2.5. Mantenimiento

El software es entregado al cliente. Se toman sus comentarios y se realiza cualquier cambio, si es necesario. Esta fase continúa hasta que el software deje de ser utilizado.

CAPÍTULO III: APLICACIÓN Y RESULTADOS

3.1. METODOLOGÍA CRISP-DM

El proyecto se desarrolló siguiendo cada una de las fases de la metodología CRISP-DM utilizando los paquetes de registros de transacciones entregados por el departamento del CSIRT-EPN.

3.1.1. Comprensión del Problema

En esta fase de la metodología, se dará respuesta a cada una de las cuestiones planteadas a continuación.

3.1.1.1. Identificación del problema

El CSIRT-EPN brinda servicios de seguridad a las infraestructuras criticas de TI (Tecnologías de la Información) en la EPN. El problema que enfrenta actualmente es la existencia de vulnerabilidades en sus DBMS, en donde se ha encontrado varios registros de ataques internos y externos a los sistemas del CSIRT-EPN. El ataque que más preocupa a los encargados del área son las inyecciones de código SQL. Estos ataques se han vuelto cada vez más frecuentes, por lo que una solución debe plantearse para este problema. En esta etapa del proyecto se determina que hay que desarrollar un prototipo de sistema que permita identificar y predecir eventos de ataques que suceden en estos servidores del CSIRT.

3.1.1.2. Determinación de los Objetivos del CSIRT-EPN

El objetivo de la minería de datos que se va a aplicar en este proyecto es hacer una identificación lo más fiable posible de inyecciones de código SQL a partir de registros de transacciones registrados por los DBMS en el CSIRT-EPN. El objetivo es proporcionar un mejor método de identificación de inyección de código SQL y así poder evitar y predecir futuros ataques. Establecidos estos principios, se pretende:

- Visualizar los datos de los registros de transacciones más relevantes para identificar un registro de inyección de código SQL.
- Determinar un método de clasificación para la identificación de anomalías.
- Identificar y predecir los ataques de inyección de código SQL en los registros de registros de transacciones.

3.1.1.3. Evaluación de la situación actual

El CSRIT-EPN cuenta con una base de datos PostgreSQL 8.4.2, con una configuración previamente establecida para la generación de registros de transacciones en un ambiente

de prueba. Este ambiente nos otorga varios paquetes de registros de transacciones en donde existen evidencias de inyección de código SQL. Esta es la principal fuente de datos que se usó para el propósito de la minería de datos. La Tabla 2 especifica las herramientas utilizadas para el desarrollo de este proyecto.

En este paso también se realizó una extensa búsqueda bibliográfica en archivos, documentos, libros, etc. Los cuales están listados en el literal 1.4. Al necesitar información fiable y precisa se indagó estudios científicos consultados en bibliotecas académicas virtuales de Google Académico. Los estudios base utilizados para este proyecto se explicaron en el literal 1.4.3.

3.1.2. Comprensión de los Datos

En esta fase se realiza la recolección inicial de datos para familiarizarse con estos y averiguar su calidad, así como identificar las relaciones más importantes para formular las primeras hipótesis.

3.1.2.1. Recolección de datos iniciales

El proyecto se realizó en un ambiente de prueba en una máquina virtual que simule el escenario manejado por el CSIRT-EPN. Para ello se utilizó una máquina virtual VMWare Workstation 16.1 y el Sistema Operativo Windows 10 donde se instaló PostgreSQL 9.5, que será el sistema gestor de base de datos de interés en este proyecto.

La creación de usuarios es la primera tarea ejecutada en el servidor PostgresSQL. En la Tabla 4 se puede visualizar los usuarios creados y los privilegios que se les asignó a cada uno.

Tabla 4. Creación de Usuarios en PostgreSQL

Usuario	Puede iniciar sesión	Superusuario	Crear roles	Crear base de datos	Actualizar catálogo	Heredar derechos de los roles principales	Iniciar la reproducción en flujo y las copias de seguridad
karen	x		x				
cesar	x					x	
gabriela	x						

Con los usuarios definidos se procede a crear una nueva base de datos a la que nombramos como “prueba”, y dentro de esta se ejecutó el archivo de respaldo “icaf.backup”, script que contiene varias tablas y datos que se usará en el ambiente de prueba. Al hacer la restauración del archivo “icaf.backup” se obtiene una base de datos que

tiene 3 esquemas; activo, gestión y público. Cada esquema cuenta con distintas tablas sobre las cuales se ejecutará varias consultas de selección, inserción, actualización, borrado, etc.

A continuación, se lista toda la información contenida dentro de un registro de transacción:

- %a = application name (nombre de la aplicación)
- %u = user name (nombre de usuario)
- %d = database name (nombre de la base de datos)
- %r = remote host and port (puerto y servidor remoto)
- %h = remote host (servidor remoto)
- %p = process ID (identificador del proceso)
- %t = timestamp without milliseconds (marca de tiempo sin milisegundos)
- %m = timestamp with milliseconds (marca de tiempo con milisegundos)
- %i = command tag (etiqueta de comando)
- %e = SQL state (estado SQL)
- %c = session ID (identificador de la sesión)
- %l = session line number (número de línea de sesión)
- %s = session start timestamp (marca de tiempo de inicio de sesión)
- %v = virtual transaction ID (identificador de transacción virtual)
- %x = transaction ID (identificador de la transacción)

Los archivos de texto plano con registros de transacciones generados por PostgreSQL se obtuvieron mediante la modificación de varios parámetros del archivo de configuración *"postgresql.config"*, el cual se encuentra en la carpeta "data" del DBMS. Estos registros de transacciones son escritos con el siguiente formato establecido:

log_line-prefix = " Bdd:%d User:%u R.Host:%r Pid:%p Tstamp:%m SesionInicio:%s Comando:%i "

En la Figura 7 se muestra la línea "log_line_prefix" del archivo de configuración de PostgreSQL, la cual fue modificada para que cada registro de transacción contenga la siguiente información: nombre de la base de datos, nombre de usuario, servidor y puerto remoto, identificador del proceso, marca de tiempo con milisegundos, marca de tiempo de inicio de sesión y la etiqueta del comando. Este formato fue establecido por la CSIRT-EPN.

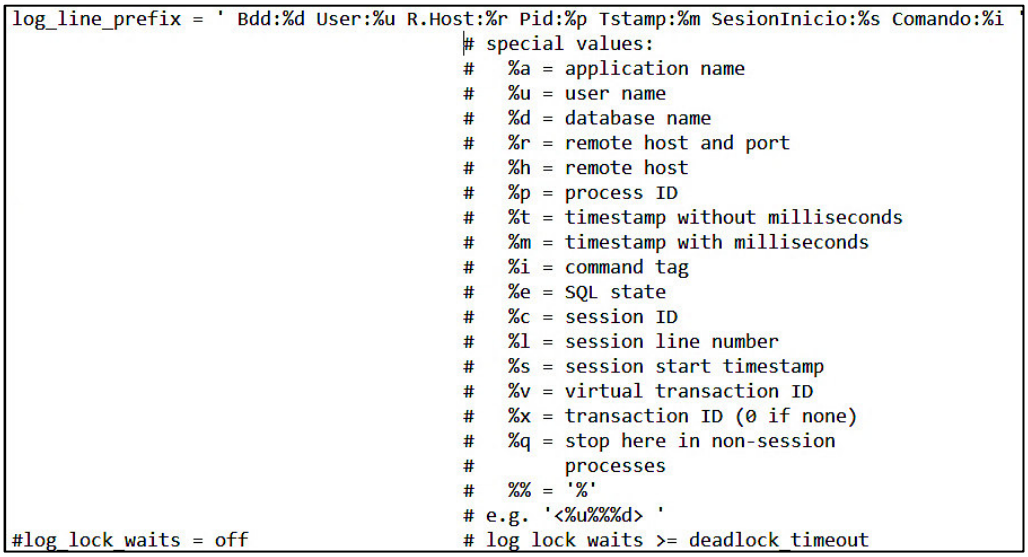


Figura 7. Archivo de Configuración "postgresql.config "de PostgreSQL

La información que será útil al momento de realizar la minería de datos es la siguiente:

- %u = user name (nombre de usuario)
- %d = database name (nombre de la base de datos)
- %r = remote host and port (puerto y servidor remoto)
- %p = process ID (identificador del proceso)
- %m = timestamp with milliseconds (marca de tiempo con milisegundos)
- %s = session start timestamp (marca de tiempo de inicio de sesión)
- %i = command tag (etiqueta de comando)

3.1.2.2. Descripción de datos

La Tabla 5 muestra la descripción de cada valor especial de los registros de transacciones que simulan un comportamiento normal.

Tabla 5. Descripción de los Atributos de los Registros de Transacciones

Valor	Efecto
%d	Nombre de la base de datos que se está utilizando para ejecutar las consultas.
%u	Nombre del usuario que se conectó a PostgreSQL.
%r	Servidor y puerto remoto. El cliente que realiza la conexión debe estar en la misma máquina que se encuentra el servidor PostgreSQL.
%p	ID del proceso. Identificador único asignado a cada registro de transacción.
%m	Es la marca de tiempo que incluye los milisegundos.
%s	Es la marca de tiempo de inicio de sesión.
%i	La etiqueta de comando devuelve la sentencia ejecutada como, por ejemplo: SELECT, INSERT, UPDATE, DELETE.

3.1.3. Preparación de los Datos

Una vez efectuada la recolección inicial de los registros de transacciones en el ambiente de prueba. Se procede a la preparación de los mismos para que puedan ser utilizados en la técnica de minería de datos escogida.

3.1.3.1. Selección de datos

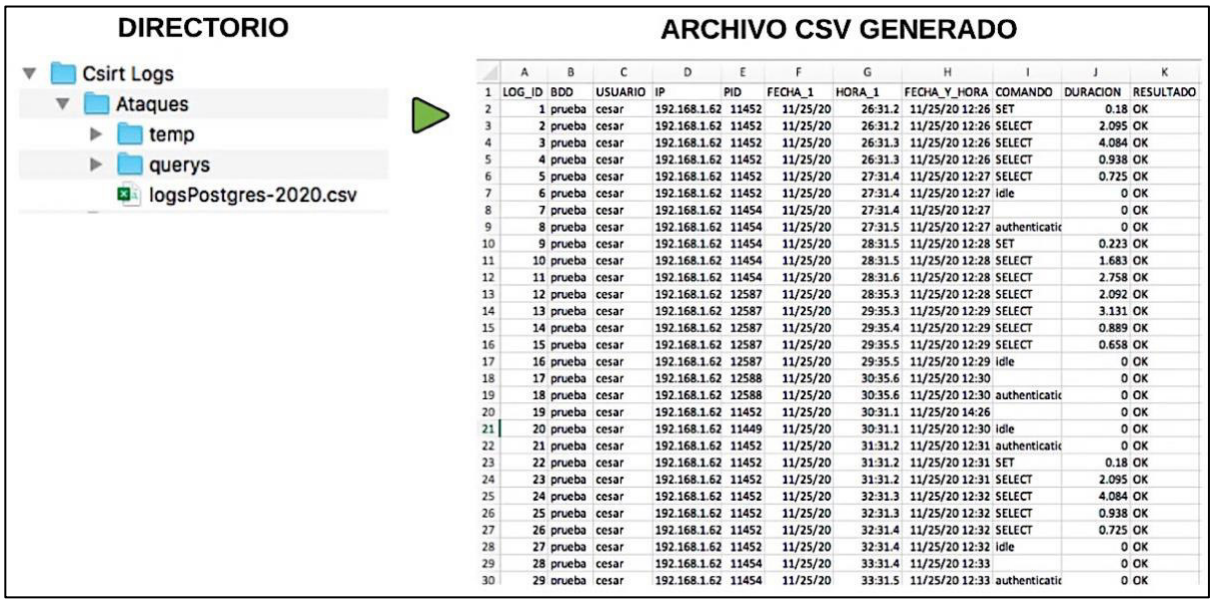
Los registros de transacciones son recolectados en un archivo de texto con el formato (.log), los cuales en un inicio están vagamente estructurados, por lo que se procede a una lectura de línea por línea del archivo y la extracción de los datos de cada registro de transacción. Para lograr esto, se utilizó el lenguaje de programación Python 3 en conjunto con las librerías: pandas y csv. En la Tabla 6 se especifican todos los datos seleccionados a partir de la lectura de registros de transacciones.

Tabla 6. Descripción de cada Atributo del Registro de Transacción

Atributo	Descripción	Tipo	Tipo de dominio
LOG_ID	Identificador único de cada registro de transacción.	discreto	integer
BDD	Nombre de la base de datos sobre la cual se está trabajando.	categórico	string
USUARIO	Nombre del usuario que se conectó a PostgreSQL.	categórico	string
IP	Dirección IP de cada computador	continuo	integer
FECHA_1	Fecha del día en el que se realizó la consulta para obtener el registro de transacción.	continuo	date
HORA_1	Hora de la consulta con milisegundos	continuo	time
FECHA_Y_HORA	Fecha y hora del día en el que se realizó la consulta para obtener el registro de transacción.	continuo	date
COMANDO	Tipo de sentencia ejecutada registrada en el registro de transacción.	categórico	string
DURACIÓN	Tiempo de duración en segundos que demora en ejecutarse la consulta.	continuo	double
RESULTADO	Resultado de ejecución de las consultas.	categórico	string

Los archivos generados a partir de este proceso de lectura son: un documento (.csv) con los atributos descritos en la Tabla 6 y un documento de texto aparte donde se almacena el LOG_ID y el QUERY DE CONSULTA de cada registro de transacción para su futuro análisis en la etapa de evaluación del modelo. Este paso es importante ya que para el uso

de la técnica de clasificado no es necesario tener una cadena de texto larga, como lo es la sentencia de consulta, por cada registro de transacción. En las Figuras 8 y 9 podemos apreciar el resultado de los nuevos archivos generados.



	A	B	C	D	E	F	G	H	I	J	K
1	LOG_ID	BDD	USUARIO	IP	PID	FECHA_1	HORA_1	FECHA_Y_HORA	COMANDO	DURACION	RESULTADO
2	1	prueba	cesar	192.168.1.62	11452	11/25/20	26:31.2	11/25/20 12:26	SET	0.18	OK
3	2	prueba	cesar	192.168.1.62	11452	11/25/20	26:31.2	11/25/20 12:26	SELECT	2.095	OK
4	3	prueba	cesar	192.168.1.62	11452	11/25/20	26:31.3	11/25/20 12:26	SELECT	4.084	OK
5	4	prueba	cesar	192.168.1.62	11452	11/25/20	26:31.3	11/25/20 12:26	SELECT	0.938	OK
6	5	prueba	cesar	192.168.1.62	11452	11/25/20	27:31.4	11/25/20 12:27	SELECT	0.725	OK
7	6	prueba	cesar	192.168.1.62	11452	11/25/20	27:31.4	11/25/20 12:27	idle	0	OK
8	7	prueba	cesar	192.168.1.62	11454	11/25/20	27:31.4	11/25/20 12:27		0	OK
9	8	prueba	cesar	192.168.1.62	11454	11/25/20	27:31.5	11/25/20 12:27	authenticatic	0	OK
10	9	prueba	cesar	192.168.1.62	11454	11/25/20	28:31.5	11/25/20 12:28	SET	0.223	OK
11	10	prueba	cesar	192.168.1.62	11454	11/25/20	28:31.5	11/25/20 12:28	SELECT	1.683	OK
12	11	prueba	cesar	192.168.1.62	11454	11/25/20	28:31.6	11/25/20 12:28	SELECT	2.758	OK
13	12	prueba	cesar	192.168.1.62	12587	11/25/20	28:35.3	11/25/20 12:28	SELECT	2.092	OK
14	13	prueba	cesar	192.168.1.62	12587	11/25/20	29:35.3	11/25/20 12:29	SELECT	3.131	OK
15	14	prueba	cesar	192.168.1.62	12587	11/25/20	29:35.4	11/25/20 12:29	SELECT	0.889	OK
16	15	prueba	cesar	192.168.1.62	12587	11/25/20	29:35.5	11/25/20 12:29	SELECT	0.658	OK
17	16	prueba	cesar	192.168.1.62	12587	11/25/20	29:35.5	11/25/20 12:29	idle	0	OK
18	17	prueba	cesar	192.168.1.62	12588	11/25/20	30:35.6	11/25/20 12:30		0	OK
19	18	prueba	cesar	192.168.1.62	12588	11/25/20	30:35.6	11/25/20 12:30	authenticatic	0	OK
20	19	prueba	cesar	192.168.1.62	11452	11/25/20	30:31.1	11/25/20 14:26		0	OK
21	20	prueba	cesar	192.168.1.62	11449	11/25/20	30:31.1	11/25/20 12:30	idle	0	OK
22	21	prueba	cesar	192.168.1.62	11452	11/25/20	31:31.2	11/25/20 12:31	authenticatic	0	OK
23	22	prueba	cesar	192.168.1.62	11452	11/25/20	31:31.2	11/25/20 12:31	SET	0.18	OK
24	23	prueba	cesar	192.168.1.62	11452	11/25/20	31:31.2	11/25/20 12:31	SELECT	2.095	OK
25	24	prueba	cesar	192.168.1.62	11452	11/25/20	32:31.3	11/25/20 12:32	SELECT	4.084	OK
26	25	prueba	cesar	192.168.1.62	11452	11/25/20	32:31.3	11/25/20 12:32	SELECT	0.938	OK
27	26	prueba	cesar	192.168.1.62	11452	11/25/20	32:31.4	11/25/20 12:32	SELECT	0.725	OK
28	27	prueba	cesar	192.168.1.62	11452	11/25/20	32:31.4	11/25/20 12:32	idle	0	OK
29	28	prueba	cesar	192.168.1.62	11454	11/25/20	33:31.4	11/25/20 12:33		0	OK
30	29	prueba	cesar	192.168.1.62	11454	11/25/20	33:31.5	11/25/20 12:33	authenticatic	0	OK

Figura 8. Registro de los Atributos de los Registros de Transacciones en un Archivo CSV

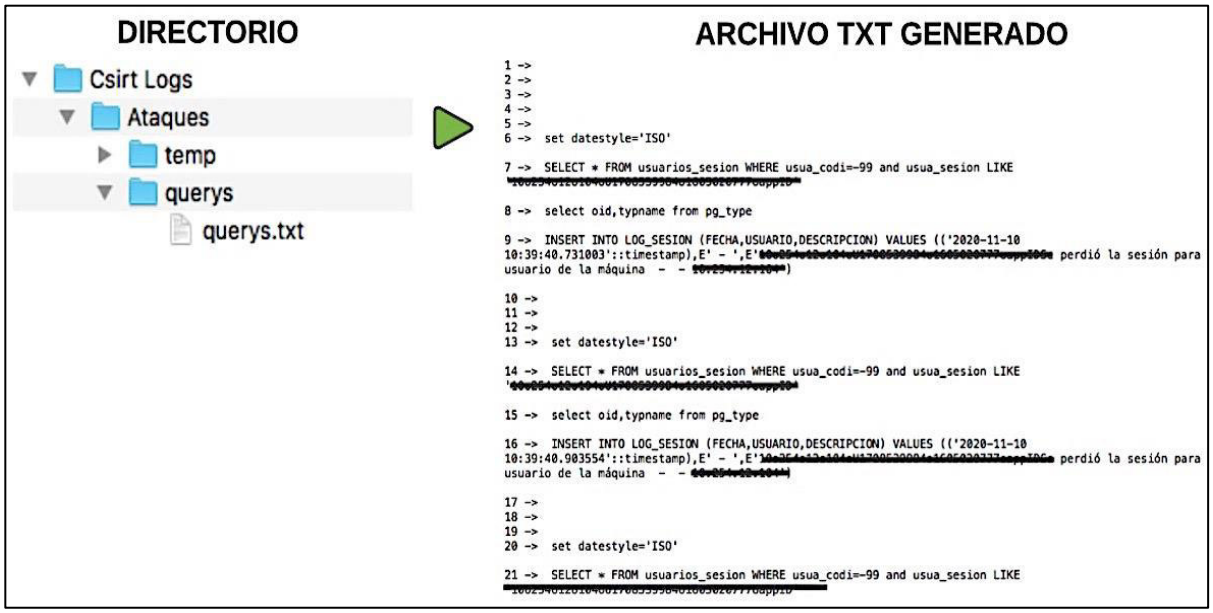


Figura 9. Registro de Sentencias de Consulta del Registro de Transacción en un Archivo TXT

La herramienta PyOD, conjuntamente con el archivo csv generado, nos permite tener una vista previa de cómo están distribuidos los registros de transacciones en el plano cartesiano según las variables que escojamos, sin necesidad de hacer cambios previos de normalización a los valores para observarlos gráficamente. De esta manera, PyOD ayuda al entendimiento del comportamiento de los registros de transacciones para tener visión clara de las mejores variables que se pueden utilizar para identificar anomalías.

En la Figura 10, generada mediante la biblioteca pyplot, se puede observar claramente que por cada hora del día hay una gran densidad de registros de registros de transacciones generados en cada consulta al servidor base de datos. Es importante y necesario conocer la dimensión de los datos a tratar para saber cómo utilizarlos eficientemente.

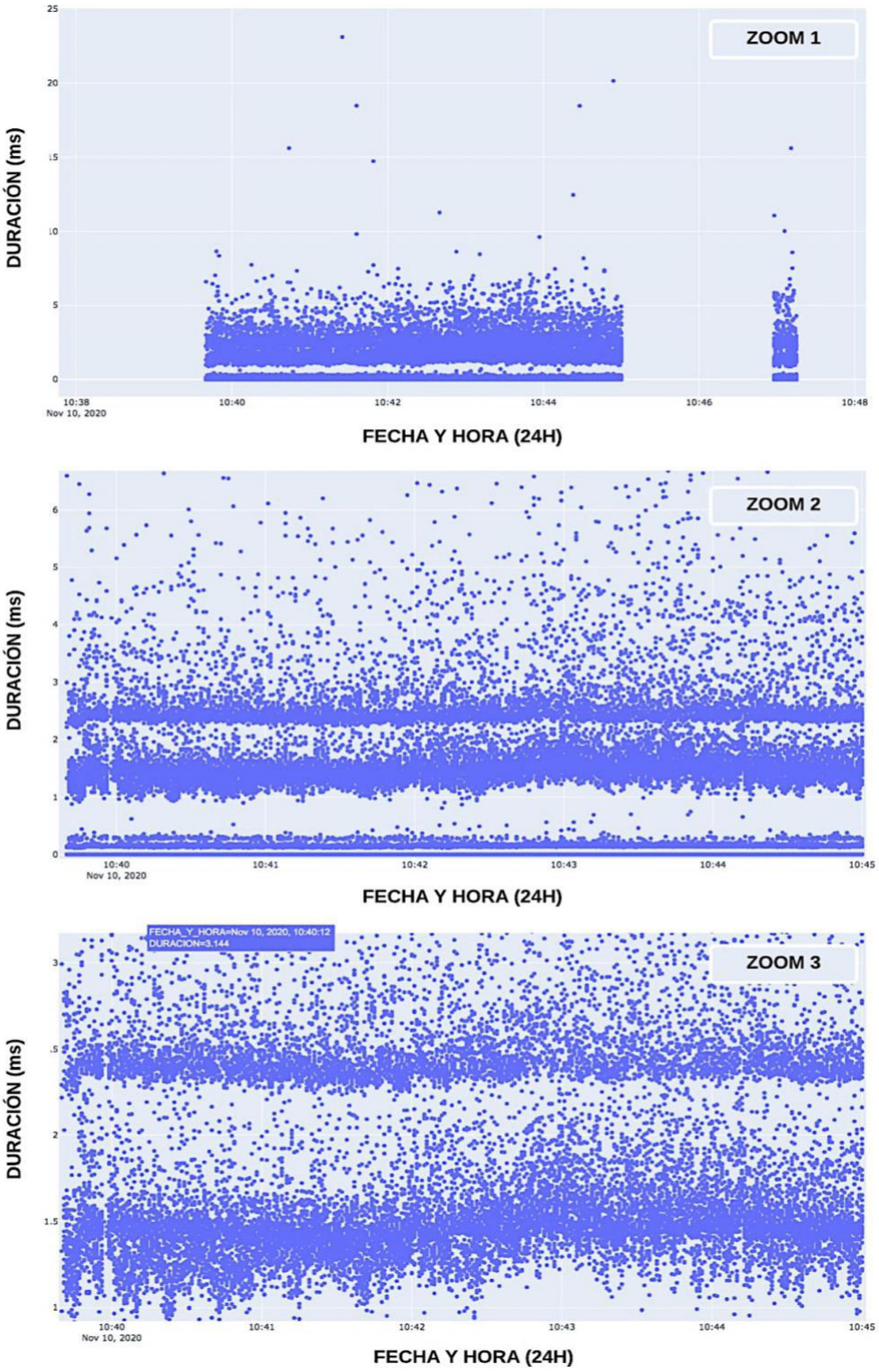


Figura 10. Duración del Registros de Transacciones y Hora de Consulta

La Figura 11 muestra el número de registros de transacciones que existe por hora, basándonos en esta representación gráfica podemos observar que existen ciertas horas del día en donde son menos activas las consultas realizadas al servidor. Por lo que este fue un factor importante a tomar en cuenta para la aplicación de la técnica de minería de datos.

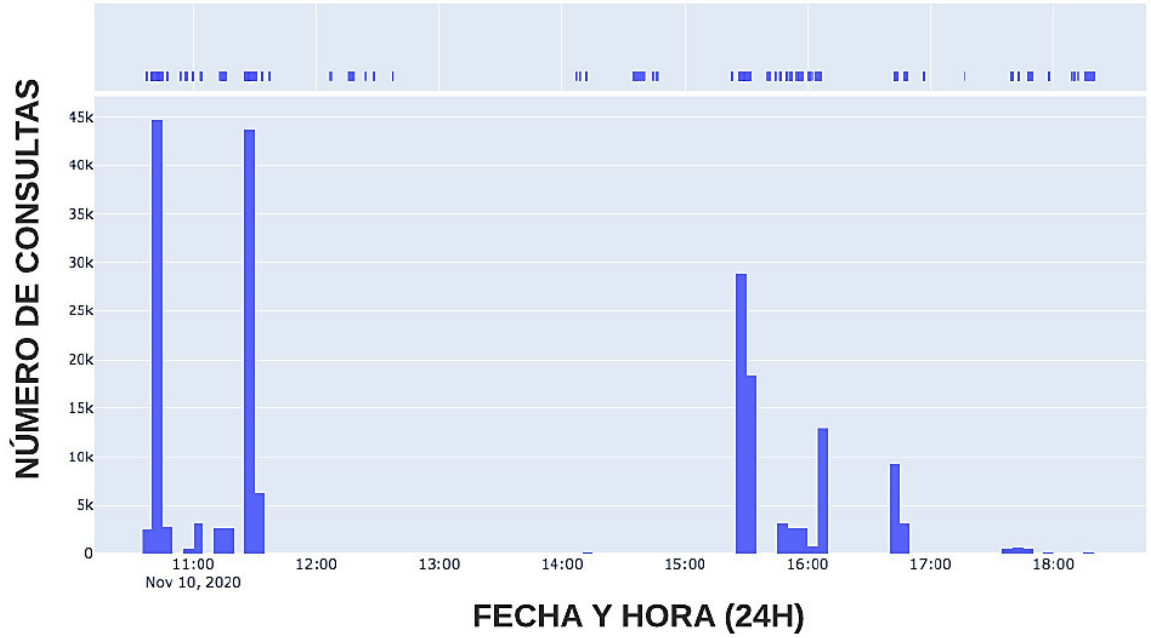


Figura 11. Número de consultas por hora

En la Figura 12 se muestra un plano diferente en el cual se visualiza el tipo de consulta que realizan los registros de transacciones en la base de datos, se puede observar con detalle que a lo largo del día los comandos más inusuales son: UPDATE, GRANT, ALTER, COPY.

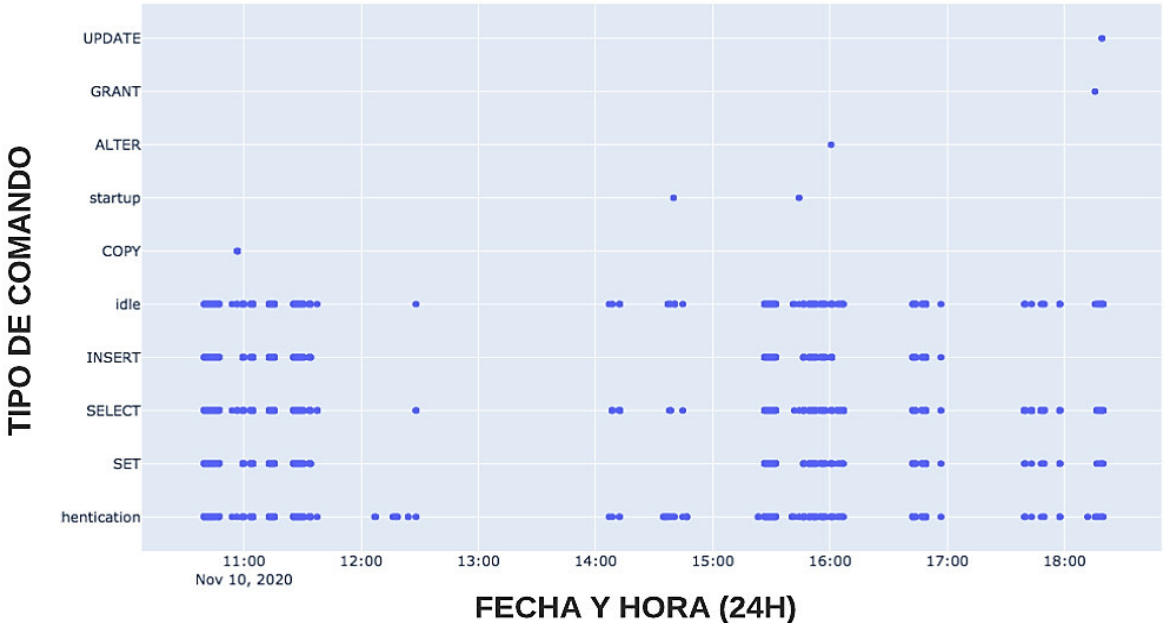


Figura 12. Tipo de comando y hora de consulta

Estas gráficas ayudaron a entender que tipo de valores son más importantes en los registros de transacciones generados, para usarlos en el modelo de minería el cual identificará apropiadamente un comportamiento diferente al normal.

3.1.3.2. Limpieza de datos

Posteriormente con el archivo (.csv) generado con los atributos señalados en la tabla 7, se crea un nuevo conjunto datos con los atributos que fueron considerados como los más influyentes para realizar una clasificación efectiva. Esto fue implementado utilizando la librería panda. Estos datos son detallados en la Tabla 7.

Tabla 7. Descripción de los Atributos del Registro de Transacción después de la Limpieza

Atributo	Descripción	Tipo	Tipo de dominio
LOG_ID	Identificador único de cada registro de transacción.	discreto	integer
USUARIO	Nombre del usuario que se conectó a PostgreSQL.	categórico	string
HORA_1	Hora de la consulta con milisegundos.	continuo	Time
FECHA_Y_HORA	Dirección IP de cada computador.	continuo	integer
COMANDO	Tipo de sentencia ejecutada registrada en el l registro de transacción.	categórico	string
DURACIÓN	Tiempo de duración en segundos que demora en ejecutarse la consulta.	continuo	double
RESULTADO	Resultado de ejecución de las consultas.	categórico	string

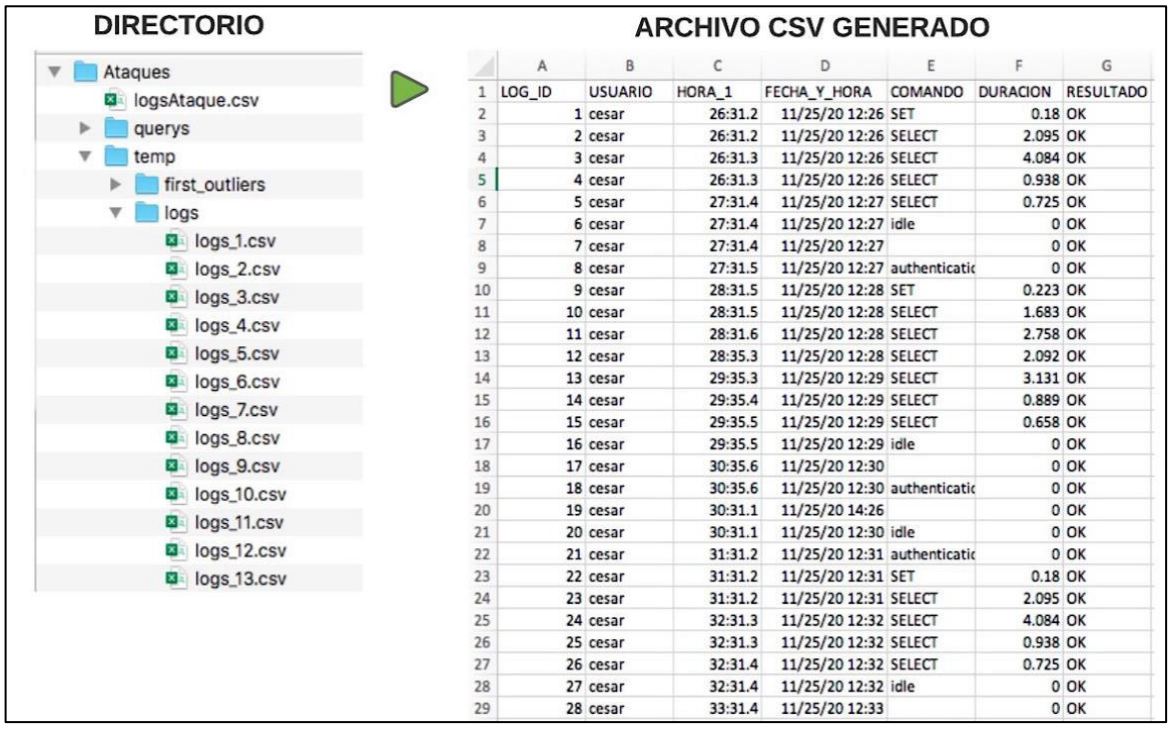
Ciertos registros de transacciones que fueron recolectados con valores nulos en la fecha y hora de consulta fueron eliminados para no entorpecer la identificación de anomalías. Un ejemplo del resultado de esta limpieza de datos puede observarse en la Figura 13.

```
MacBook-Pro% python SegundoPaso.py
   LOG_ID USUARIO        HORA_1          FECHA_Y_HORA COMANDO  DURACION RESULTADO
0       1   cesar  12:26:31.196  2020-11-25 12:26:31     SET     0.180        OK
1       2   cesar  12:26:31.229  2020-11-25 12:26:31  SELECT     2.095        OK
2       3   cesar  12:26:31.261  2020-11-25 12:26:31  SELECT     4.084        OK
3       4   cesar  12:26:31.337  2020-11-25 12:26:31  SELECT     0.938        OK
4       5   cesar  12:27:31.367  2020-11-25 12:27:31  SELECT     0.725        OK
FILE CREATED _ ---> Ataques/temp/logs/logs_1.csv
```

Figura 13. Atributos obtenidos a partir de la Limpieza de Datos

3.1.3.3. Construcción e integración de los datos

Una vez eliminadas las columnas innecesarias y los valores nulos en los registros de los registros de transacciones, es necesario identificar cual es la cantidad máxima de registros de transacciones con la que mejor trabaje el algoritmo. Teniendo en cuenta el análisis de datos realizado con la herramienta PyOD se tomó la decisión de generar archivos separados de 5000 registros de registro de transacción en cada archivo lo cual se observa en la Figura 14.



	A	B	C	D	E	F	G
1	LOG_ID	USUARIO	HORA_1	FECHA_Y_HORA	COMANDO	DURACION	RESULTADO
2	1	cesar	26:31.2	11/25/20 12:26	SET	0.18	OK
3	2	cesar	26:31.2	11/25/20 12:26	SELECT	2.095	OK
4	3	cesar	26:31.3	11/25/20 12:26	SELECT	4.084	OK
5	4	cesar	26:31.3	11/25/20 12:26	SELECT	0.938	OK
6	5	cesar	27:31.4	11/25/20 12:27	SELECT	0.725	OK
7	6	cesar	27:31.4	11/25/20 12:27	idle	0	OK
8	7	cesar	27:31.4	11/25/20 12:27		0	OK
9	8	cesar	27:31.5	11/25/20 12:27	authenticatic	0	OK
10	9	cesar	28:31.5	11/25/20 12:28	SET	0.223	OK
11	10	cesar	28:31.5	11/25/20 12:28	SELECT	1.683	OK
12	11	cesar	28:31.6	11/25/20 12:28	SELECT	2.758	OK
13	12	cesar	28:35.3	11/25/20 12:28	SELECT	2.092	OK
14	13	cesar	29:35.3	11/25/20 12:29	SELECT	3.131	OK
15	14	cesar	29:35.4	11/25/20 12:29	SELECT	0.889	OK
16	15	cesar	29:35.5	11/25/20 12:29	SELECT	0.658	OK
17	16	cesar	29:35.5	11/25/20 12:29	idle	0	OK
18	17	cesar	30:35.6	11/25/20 12:30		0	OK
19	18	cesar	30:35.6	11/25/20 12:30	authenticatic	0	OK
20	19	cesar	30:31.1	11/25/20 14:26		0	OK
21	20	cesar	30:31.1	11/25/20 12:30	idle	0	OK
22	21	cesar	31:31.2	11/25/20 12:31	authenticatic	0	OK
23	22	cesar	31:31.2	11/25/20 12:31	SET	0.18	OK
24	23	cesar	31:31.2	11/25/20 12:31	SELECT	2.095	OK
25	24	cesar	32:31.3	11/25/20 12:32	SELECT	4.084	OK
26	25	cesar	32:31.3	11/25/20 12:32	SELECT	0.938	OK
27	26	cesar	32:31.4	11/25/20 12:32	SELECT	0.725	OK
28	27	cesar	32:31.4	11/25/20 12:32	idle	0	OK
29	28	cesar	33:31.4	11/25/20 12:33		0	OK

Figura 14. Nuevos archivos CSV de 5000 registros de registros de transacciones

3.1.3.4. Normalización

Con cada archivo generado en los pasos anteriores damos paso a la normalización de los mismos, esto significa remodelar su espacio de datos de modo que la distribución de estos a lo largo de todas las dimensiones del plano cartesiano sea aproximadamente la misma. Esto hace más fácil y rápida la identificación de anomalías en los registros de transacciones.

El método tradicional de normalización utilizado para kNN es la normalización mínima-máxima. La normalización mínimo-máximo resta el valor mínimo de un atributo de cada valor del atributo y luego divide la diferencia por el rango del atributo. Estos nuevos valores se multiplican por el nuevo rango del atributo y finalmente se suman al nuevo valor mínimo del atributo. Estas operaciones transforman los datos en un nuevo rango, generalmente [0,1] como se observa en la Ecuación 1.

$$Xnew = \frac{X - Min(x)}{Max(X) - Min(X)} \qquad (1)$$

Como resultado de esta normalización de datos obtenemos valores dentro de rangos más pequeños, como se muestran en la Figura 15. Esta acción fue implementada con el uso de la librería sklearn.

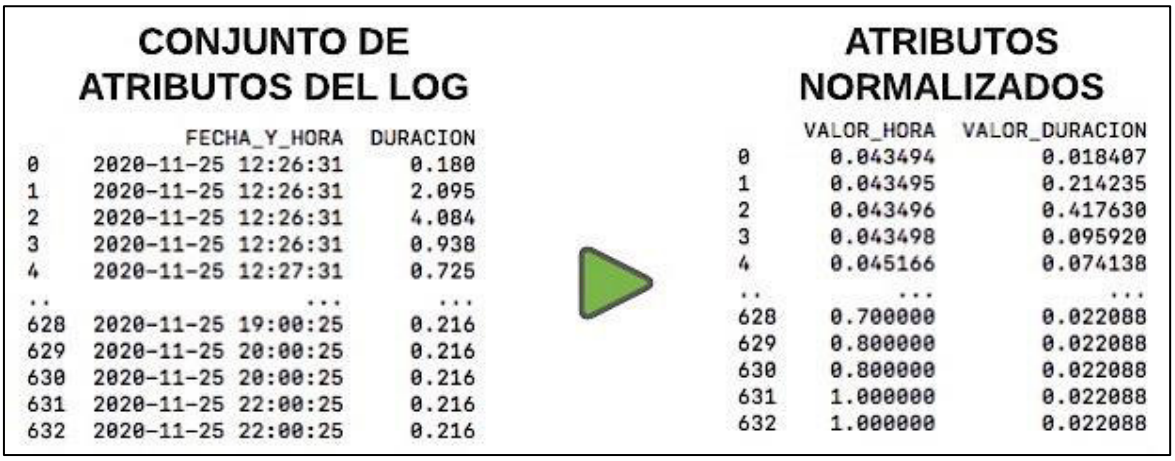


Figura 15. Normalización Aplicada al Conjunto de datos

3.1.4. Modelado

La identificación de las anomalías (valores atípicos) se realizó mediante el uso de la herramienta PyOD. Se ejecutaron 5 algoritmos que identifican anomalías: kNN, Isolation Forest, CBLOF, HBOS y Feature Bagging. Las características de cada uno de ellos se pueden visualizar en la Tabla 8, la cual nos permitió entender cómo trabaja cada algoritmo y cuál es el mejor para este proyecto. Los algoritmos utilizados se usaron dentro de un plano el cual compara el tiempo de demora de la ejecución de la consulta y la hora de la consulta en el día. Siendo el tiempo de demora el factor más importante al identificar los ataques de inyección de código SQL en el servidor de base de datos.

Tabla 8. Comparativa de los Algoritmos para Identificar Anomalías				
Nombre	**Tipo de Algoritmo**	**Supervisado Sí/No**	**Ventajas**	**Desventajas**
kNN	Algoritmo de clasificación.	Sí	Es sencillo, no hace suposiciones sobre la distribución que siguen los datos.	Lento si se trabaja con una gran cantidad de datos.
Isolation Forest	Basado en proximidad	No	El árbol no se debe construir por completo porque las anomalías se quedan en la parte superior. Muestras pequeñas de datos producen buenos resultados, porque evita identificar erróneamente datos normales como anomalías)	La complejidad de algunos de estos modelos aumenta con la dimensión y el tamaño de los datos.
CBLOF	Basado en proximidad	No	Tiene un rendimiento óptimo con un tamaño pequeño de dimensiones.	No es capaz de capturar anomalías para un conjunto de datos de altas dimensiones.
HBOS	Basado en proximidad	No	Tiene un buen desempeño en problemas de detección de anomalías globales. Su ejecución es rápida al trabajar con grandes conjuntos de datos.	No puede detectar valores atípicos locales.
Feature Bagging	Algoritmo de clasificación	No	Al entrenar un conjunto de datos toma una muestra de características aleatorias en lugar de la totalidad.	Debe combinarse con otros métodos para mejorar la precisión predictiva de valores atípicos.

Para obtener las gráficas representativas de cada algoritmo se utilizó la biblioteca pyod.models de python, obteniendo los siguientes resultados. Las figuras se encuentran estructuradas de la siguiente manera:

- **Primer Cuadro. -** Datos ubicados en el plano
  - El eje X consiste en el tiempo de duración en milisegundos
  - El eje Y consiste el numero de minutos totales en la hora del día
  - Los puntos azules representan el valor de cada registro de transacción en el plano.
- **Segundo Cuadro. -** Datos ejecutados por el algoritmo clasificador
  - El eje X consiste en el tiempo de duración en milisegundos normalizado con un valor del 0 a 1.
  - El eje Y consiste el numero de minutos totales en la hora del día normalizado con un valor de 0 a 1.
  - Los puntos blancos se encuentran dentro de la función de decisión aprendida y representan los registros de transacciones identificados como normales.
  - Los puntos negros se encuentran fuera de la función de decisión aprendida y representan los registros de transacciones identificados como anomalías.

El programa de código escrito hasta este punto incluye la lectura, transformación y limpieza los registros de transacciones hasta la implementación de los algoritmos clasificadores.

La Figura 16 muestra el resultado de la ejecución del programa utilizando el algoritmo Isolation Forest.

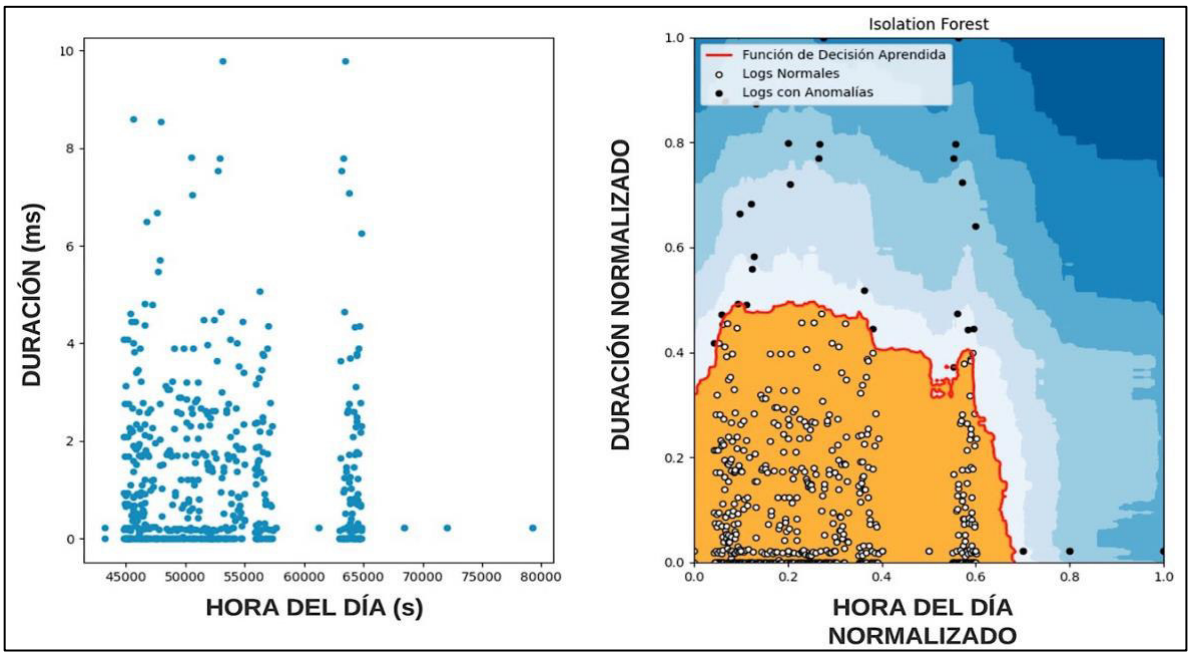


Figura 16. Anomalías Identificadas con Isolation Forest

La Figura 17 permite visualizar el resultado de la ejecución del programa al usar el algoritmo Feature Bagging.

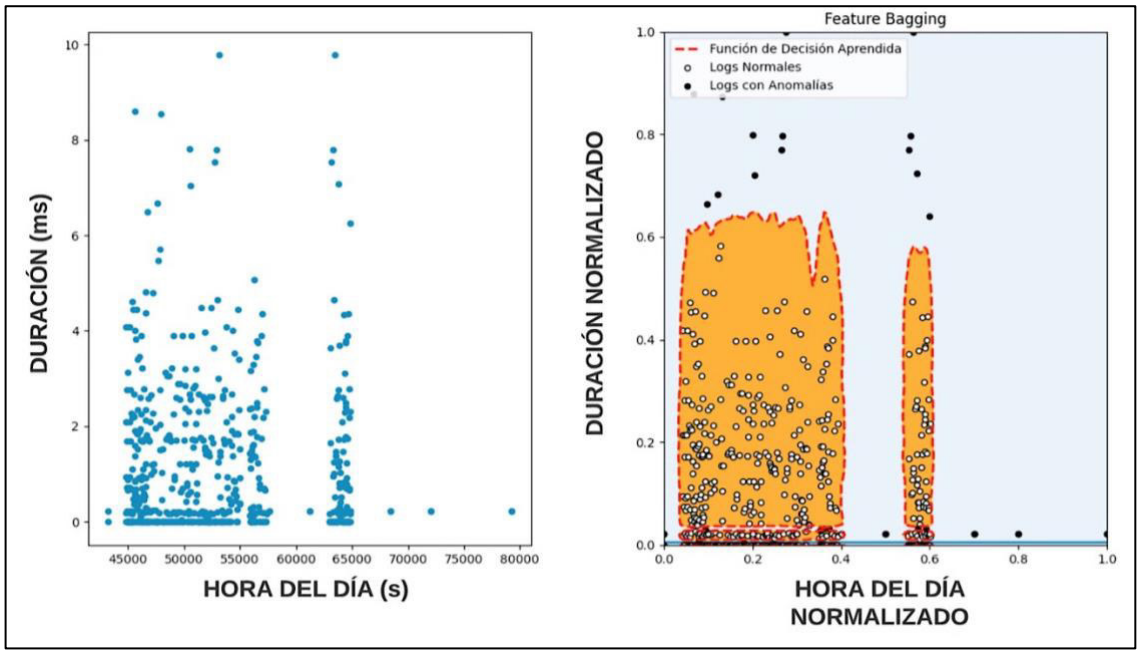


Figura 17. Anomalías Identificadas con Feature Bagging

En la Figura 18 podemos ver el resultado obtenido al utilizar el algoritmo HBOS en la ejecución del programa.

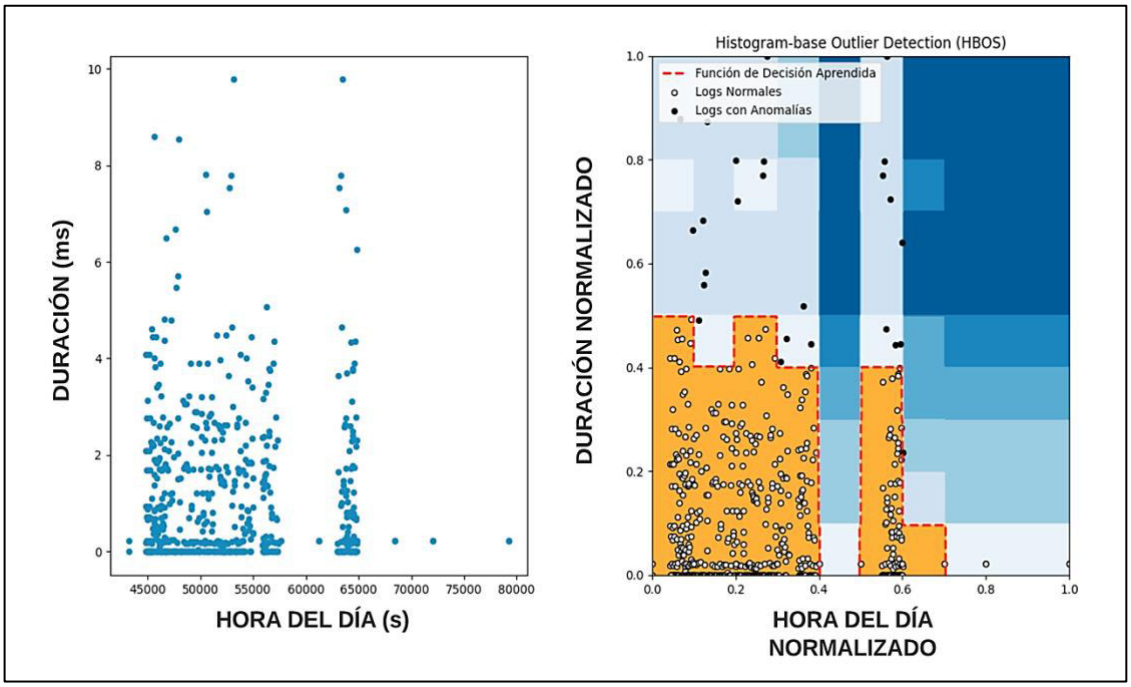


Figura 18. Anomalías Identificadas con HBOS

La Figura 19 nos muestra el resultado de ejecutar el programa utilizando el algoritmo CBLOF.

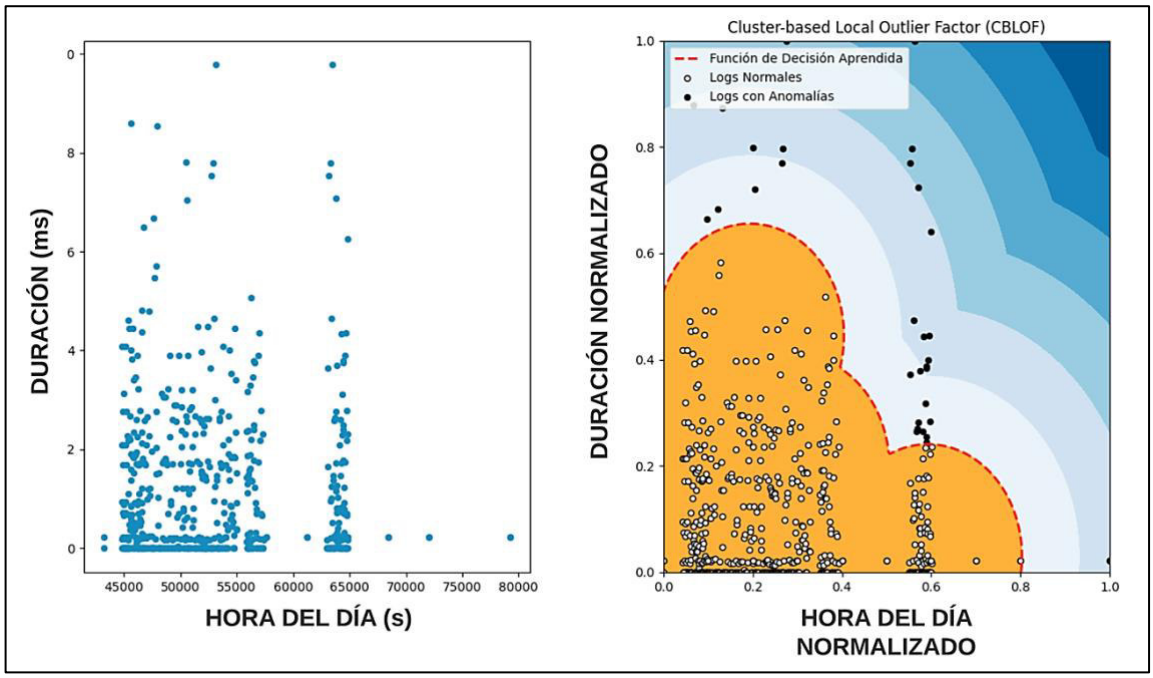


Figura 19. Anomalías Identificadas con CBLOF

Finalmente, en la Figura 20 se identifican las anomalías detectadas mediante el uso del algoritmo kNN que fue el utilizado en el desarrollo de este proyecto.

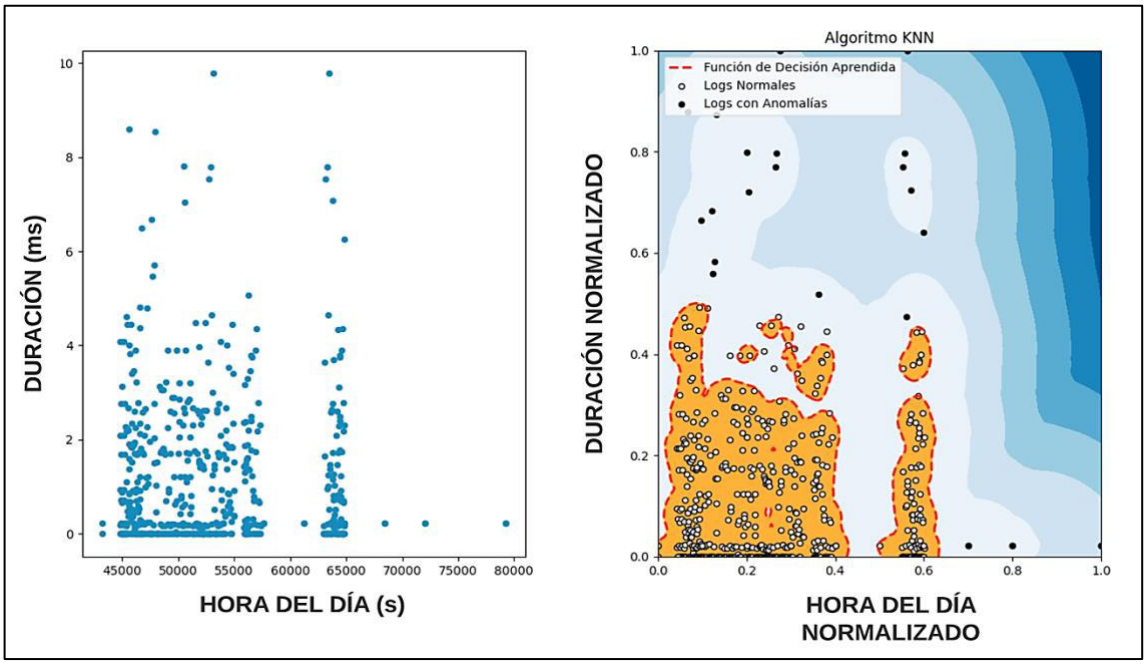


Figura 20. Anomalías Identificadas con kNN

La Figura 21 indica el número de valores atípicos (*outliers*) y de valores normales (*inliers*) por cada algoritmo ejecutado dentro del conjunto de 633 registros de transacciones ejecutados en el programa desarrollado hasta este paso.

```
OUTLIERS :  32 INLIERS :  601 Cluster-based Local Outlier Factor (CBLOF)
OUTLIERS :  32 INLIERS :  601 Feature Bagging
OUTLIERS :  25 INLIERS :  608 Histogram-base Outlier Detection (HBOS)
OUTLIERS :  32 INLIERS :  601 Isolation Forest
OUTLIERS :  24 INLIERS :  609 Algoritmo KNN
```

Figura 21. Resultados de Herramienta PyOD

Al realizar las gráficas comparativas de los algoritmos se obtuvo dos aspectos importantes.

- El primero es visualizar como los algoritmos identificadores de anomalías utilizan la función de decisión aprendida y como esta separa los registros de transacciones con comportamiento normal de los anormales.
- El segundo y más importante, el número de ataques que fueron correctamente clasificadas como valores atípicos.

Tomando en cuenta que, de los 633 registros de transacción ingresados 20 eran ataques de prueba, el algoritmo kNN separó el mayor número de registro de ataques con el menor número de anomalías. Por lo que se estableció el uso del kNN para este modelo, esta decisión también está basada en la documentación expuesta en el literal 1.4.3 así como también en la siguiente etapa de la metodología se realiza una evaluación más exhaustiva para asegurar su funcionalidad.

3.1.4.1. Uso del kNN

El clasificador kNN se basa en una función de distancia que mide la diferencia o similitud entre dos instancias. La distancia euclidiana estándar “d (x, y)” entre dos instancias “x” y “y” es definida mediante la fórmula:

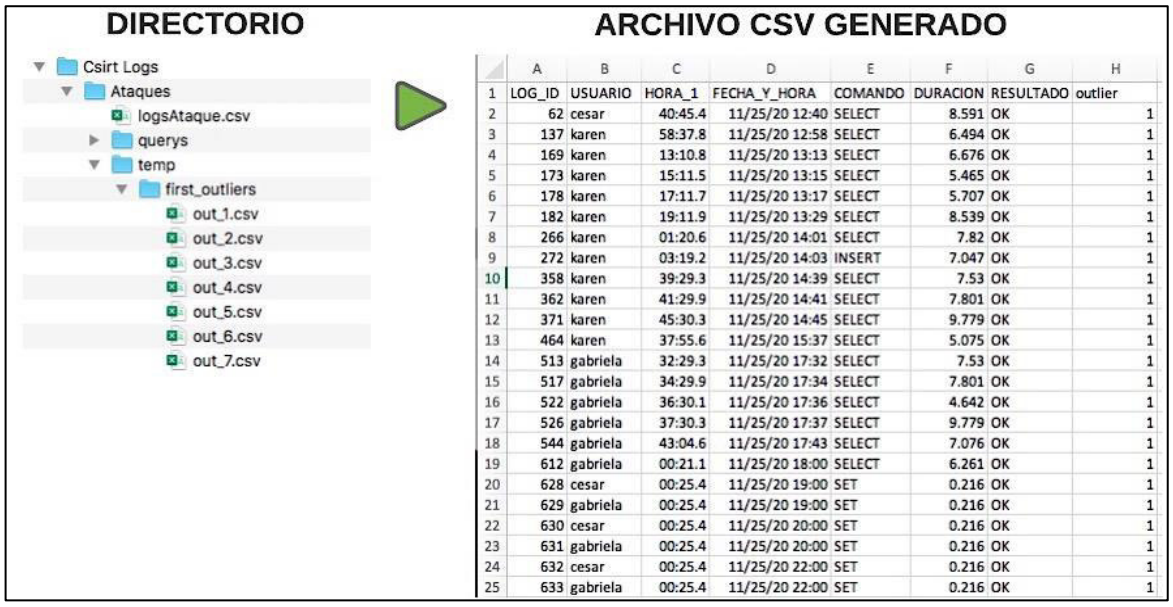
$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (2)$$

En la ecuación 2, 'x’ representa un punto no clasificado en el plano cartesiano, 'y' representa un punto ya clasificado en el plano cartesiano; siendo el cálculo de ambos el número total de distancias medidas dentro del conjunto de datos. Hay que tener en cuenta que el algoritmo kNN permite realizar predicciones no paramétricas, esto quiere decir que no necesita de la elaboración de un modelo previo. Para hacer una predicción, el algoritmo kNN no calcula un modelo predictivo a partir de un conjunto de datos de entrenamiento como en la regresión logística o lineal. Por lo tanto, para kNN, no hay una fase de

aprendizaje real. Es por eso que generalmente se clasifica como un método de aprendizaje perezoso (Wang et al., 2020).

3.1.4.2. Extracción de registros de transacciones

Una vez obtenido la función de decisión, que separa los registros de transacciones identificados como normales y anormales, se asignó el valor de 'atípico' a los registros de transacciones que siguieron a la siguiente etapa de la identificación, la cual consiste en el análisis de texto de la cadena de consulta de cada registro de transacción anómalo. Una vez obtenido la función de decisión, que separa los registros de transacciones identificados como normales y anormales, se asignó el valor de 'atípico' a los registros de transacciones que siguieron a la siguiente etapa de la identificación, la cual consiste en la extracción y análisis de texto de la cadena de consulta de cada registro de transacción anómalo. Para esto se creó una nueva columna en el conjunto de datos dentro del archivo .csv y se lo guardó en un nuevo archivo donde solo se contemple los datos de los registros de transacciones con anomalías. Tal como se muestra en la Figura 22.



	A	B	C	D	E	F	G	H
1	LOG_ID	USUARIO	HORA_1	FECHA_Y_HORA	COMANDO	DURACION	RESULTADO	outlier
2	62	cesar	40:45.4	11/25/20 12:40	SELECT	8.591	OK	1
3	137	karen	58:37.8	11/25/20 12:58	SELECT	6.494	OK	1
4	169	karen	13:10.8	11/25/20 13:13	SELECT	6.676	OK	1
5	173	karen	15:11.5	11/25/20 13:15	SELECT	5.465	OK	1
6	178	karen	17:11.7	11/25/20 13:17	SELECT	5.707	OK	1
7	182	karen	19:11.9	11/25/20 13:29	SELECT	8.539	OK	1
8	266	karen	01:20.6	11/25/20 14:01	SELECT	7.82	OK	1
9	272	karen	03:19.2	11/25/20 14:03	INSERT	7.047	OK	1
10	358	karen	39:29.3	11/25/20 14:39	SELECT	7.53	OK	1
11	362	karen	41:29.9	11/25/20 14:41	SELECT	7.801	OK	1
12	371	karen	45:30.3	11/25/20 14:45	SELECT	9.779	OK	1
13	464	karen	37:55.6	11/25/20 15:37	SELECT	5.075	OK	1
14	513	gabriela	32:29.3	11/25/20 17:32	SELECT	7.53	OK	1
15	517	gabriela	34:29.9	11/25/20 17:34	SELECT	7.801	OK	1
16	522	gabriela	36:30.1	11/25/20 17:36	SELECT	4.642	OK	1
17	526	gabriela	37:30.3	11/25/20 17:37	SELECT	9.779	OK	1
18	544	gabriela	43:04.6	11/25/20 17:43	SELECT	7.076	OK	1
19	612	gabriela	00:21.1	11/25/20 18:00	SELECT	6.261	OK	1
20	628	cesar	00:25.4	11/25/20 19:00	SET	0.216	OK	1
21	629	gabriela	00:25.4	11/25/20 19:00	SET	0.216	OK	1
22	630	cesar	00:25.4	11/25/20 20:00	SET	0.216	OK	1
23	631	gabriela	00:25.4	11/25/20 20:00	SET	0.216	OK	1
24	632	cesar	00:25.4	11/25/20 22:00	SET	0.216	OK	1
25	633	gabriela	00:25.4	11/25/20 22:00	SET	0.216	OK	1

Figura 22. Archivos con Registros de Transacciones Anormales

3.1.5. Evaluación del Modelo

3.1.5.1. Análisis del algoritmo de detección

El conjunto de registros de transacciones identificados por el algoritmo implementado es clasificado como anomalías. Posteriormente, solo queda identificar qué anomalías son las que eventualmente fueron un ataque malicioso. Para esto el código implementado analiza cada una de las cadenas de consulta de cada registro de transacción identificado como valores atípicos. Este análisis es puramente análisis de texto, ya que los ataques maliciosos

tienen sentencias únicas que los destacan de otros tipos de consultas normales que se realizan en una base de datos. En la Figura 23 se puede observar los registros de transacciones que fueron identificados como anomalías, ah estos datos se añadieron las dos últimas columnas; las cuales representan valores de datos normalizados.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	LOG_ID	BDD	USUARIO	IP	PID	FECHA_1	HORA_1	FECHA_Y_HC	COMANDO	DURACION	RESULTADO	VALOR_HOR	VALOR_DUR
2	62	prueba	cesar	192.168.1.62	12610	11/25/20	40:45.4	##########	SELECT	8.591	OK	0.06722286	0.87851519
3	137	prueba	karen	192.168.1.79	10082	11/25/20	58:37.8	##########	SELECT	6.494	OK	0.09701167	0.66407608
4	169	prueba	karen	192.168.1.79	10012	11/25/20	13:10.8	##########	SELECT	6.676	OK	0.12126223	0.68268739
5	173	prueba	karen	192.168.1.79	10015	11/25/20	15:11.5	##########	SELECT	5.465	OK	0.12461364	0.5588506
6	178	prueba	karen	192.168.1.79	10016	11/25/20	17:11.7	##########	SELECT	5.707	OK	0.12795289	0.5835975
7	182	prueba	karen	192.168.1.79	10016	11/25/20	19:11.9	##########	SELECT	8.539	OK	0.13129189	0.87319767
8	266	prueba	karen	192.168.1.79	10173	11/25/20	01:20.6	##########	SELECT	7.82	OK	0.2015331	0.79967277
9	272	prueba	karen	192.168.1.79	10169	11/25/20	03:19.2	##########	INSERT	7.047	OK	0.20482835	0.72062583
10	358	prueba	karen	192.168.1.79	11440	11/25/20	39:29.3	##########	SELECT	7.53	OK	0.26510747	0.77001738
11	362	prueba	karen	192.168.1.79	11443	11/25/20	41:29.9	##########	SELECT	7.801	OK	0.26845682	0.79772983
12	371	prueba	karen	192.168.1.79	11444	11/25/20	45:30.3	##########	SELECT	9.779	OK	0.2751351	1
13	464	prueba	karen	192.168.1.79	80813	11/25/20	37:55.6	##########	SELECT	5.075	OK	0.36250538	0.51896922
14	513	prueba	gabriela	192.168.1.79	11440	11/25/20	32:29.3	##########	SELECT	7.53	OK	0.55344055	0.77001738
15	517	prueba	gabriela	192.168.1.79	11443	11/25/20	34:29.9	##########	SELECT	7.801	OK	0.55678991	0.79772983
16	522	prueba	gabriela	192.168.1.79	11444	11/25/20	36:30.1	##########	SELECT	4.642	OK	0.56012913	0.47469066
17	526	prueba	gabriela	192.168.1.79	11444	11/25/20	37:30.3	##########	SELECT	9.779	OK	0.56180152	1
18	544	prueba	gabriela	192.168.1.79	11627	11/25/20	43:04.6	##########	SELECT	7.076	OK	0.57108867	0.72359137

Figura 23. Registros de Transacciones Identificados como Anomalías.

En la Figura 24 se pueden el archivo de texto donde fueron almacenadas el registro de consulta de todos los registros de transacciones. Estas consultas son arbitrarias y aun no representan las clasificadas como anomalías.

querys.rtf — Edited

Helvetica Regular 18

querys.rtf

177 -> SELECT usua_codi, usua_pasw FROM usuario WHERE usua_login='[redacted]

178 -> select oid,typname from pg_type

179 -> SELECT *, (('2020-11-25 13:23:11.644783'::timestamp)-usua_fech_sesion) as "tiempo" FROM usuarios_sesion WHERE usua_codi=[redacted]

180 -> INSERT INTO LOG_ACCESO (FECHA,USUARIO,IP,INTENTOS,ACCESO) VALUES (('2020-11-25 13[redacted]::timestamp),E'U[redacted]' - - 10[redacted])

181 -> UPDATE USUARIOS_SESION SET USUA_FECH_SESION=('2020-11-25 13[redacted]::timestamp),USUA_INTENTOS=1 WHERE USUA_CODI=[redacted]

182 -> SELECT * FROM USUARIO WHERE USUA_LOGIN ='[redacted] AND ([redacted]or USUA_NUEVO=0) and (inst_estado=1 or usua_logi[redacted]') order by tipo_usuario asc, usua_codi asc

Figura 24. Consultas de Texto Asociados a las Anomalías.

a) Ataques de inyección de código SQL basados en tautologías

El propósito de los ataques de inyección de código SQL basados en tautologías es formar enunciados condicionales que son siempre verdaderos. Es importante destacar que estos ataques pueden ser efectuados no solo desde un enlace, si no también mediante el ingreso de campo en un formulario.

- Enlace utilizado:

  http://localhost:3434/users.php?name=gabriela

- Consulta enviada a la base de datos:

```
SELECT * FROM gestion.ges_usuario
WHERE usr_name = 'gabriela'
AND usr_password = ''
```

- El ataque simulado fue realizado mediante el uso de la expresión:

  http://localhost:3434/users.php?name=gabriela and 1=1--+

- Nueva consulta a la base de datos:

```
SELECT * FROM gestion.ges_usuario
WHERE usr_name = 'gabriela'
AND usr_password = '' OR '1'='1';
```

Esta consulta al ser evaluada da siempre el resultado positivo. Con lo que se evita los controles y se accede a la base de datos.

b) Ataques de inyección de código SQL basados en consultas ilegales

Los ataques basados en consultas ilegales hacen uso de los mensajes de error para obtener información de la base de datos.

- Enlace utilizado:

  http://localhost:3434/groups.php?id=5?

- Consulta enviada a la base de datos:

```
SELECT gru_nombre, gru_cod, gru_nivel FROM activo.grupo
WHERE gru_id=5;
```

- El ataque simulado fue realizado mediante el uso de la expresión:

  *http://localhost:3434/users.php?name=gabriela and select * from gestion.ges_catalogo where cat_nombre = 'belleza'latina'*

- Nueva consulta a la base de datos:

```
SELECT gru_nombre, gru_cod, gru_nivel FROM activo.grupo
WHERE gru_id=5 AND SELECT * FROM gestion.ges_catalogo
WHERE cat_nombre = 'belleza'latina'
```

c) Ataques de inyección de código SQL basados en el operador unión

Los ataques de inyección de código SQL basados en el operador unión son consultas más elaboradas a diferencia de las tautologías, ya que se debe tener conocimiento de la vulnerabilidad para enviar la consulta.

- Enlace utilizado:

  http://localhost:3434/groups.php?id=5?

- Consulta enviada a la base de datos:

```
SELECT gru_nombre, gru_cod, gru_nivel FROM activo.grupo
WHERE gru_id=5;
```

- El ataque simulado fue realizado mediante el uso de la expresión:

  http://localhost:3434/users.php?name=gabriela and union select prov_ruc, prov_razon_social, 22 from activo. proveedor where '1'='1' --+

- Nueva consulta a la base de datos:

```
SELECT gru_nombre, gru_cod, gru_nivel
FROM activo.grupo
WHERE gru_id=5
UNION SELECT prov_ruc, prov_razon_social, 22
FROM activo.proveedor
WHERE '1'='1';
```

Estas consultas utilizan el operador unión de varias maneras para conseguir información de bases, tablas y atributos. Estos ataques permiten al atacante combinar más de un comando SQL en una sola consulta. Mientras mayor sea la creatividad del atacante, más variedad de consultas de unión se podrá crear.

d) Ataques de inyección de código SQL basados en consultas adicionales

En estos tipos de ataques de inyección de código SQL el atacante manipula los datos usando las cláusulas: INSERT, UPDATE y DELETE sin alterar la lógica de la consulta original. Lo que pretende es que la base de datos prueba reciba más de una consulta SQL.

- Enlace utilizado:

  http://localhost:3434/bodegas.php?name=BODEGA1

- Consulta enviada a la base de datos:

```
SELECT * FROM gestion.ges_bodega
WHERE bod_nombre='BODEGA1';
```

- El ataque simulado fue realizado mediante el uso de la expresión:

  http://localhost:3434/bodegas.php?name=BODEGA1; drop table gestion.ges_usuario --+

- Nueva consulta a la base de datos:

```
SELECT * FROM gestion.ges_bodega
WHERE bod_nombre='BODEGA1';
DROP TABLE gestion.ges_usuario;
```

Los ataques hacen uso de la expresión ';' para añadir sentencias al final del enlace. Esta vulnerabilidad es el resultado de una falta de validación de entradas y de consultas parametrizadas.

e) Ataques de inyección de código SQL inferenciales basados en un booleano

Los ataques de inyección de código SQL inferenciales basados en un booleano hacen preguntas a la base de datos mediante una sentencia que le obliga a retornar un resultado diferente a la consulta establecida.

- Enlace utilizado:

  http://localhost:3434/bodegas.php?name=BODEGA1

- Consulta enviada a la base de datos:

```
SELECT * FROM gestion.ges_bodega
WHERE bod_nombre='BODEGA1';
```

- El ataque simulado fue realizado mediante el uso de la expresión:

  http://localhost:3434/bodegas.php?name=bodega1 and declare @s varchar (8000); select @s = prueba (); if (ascii (substring (@s, 1, 1)) & (power (2, 0))) > 0 waitfor delay '0:0:5'

- Nueva consulta a la base de datos:

```
SELECT * FROM gestion.ges_bodega
WHERE bod_nombre='BODEGA1' AND
DECLARE @s VARCHAR (8000);
SELECT @s = prueba ();
IF (ascii (SUBSTRING (@s, 1, 1)) & (power (2, 0))) > 0
waitfor delay '0:0:5'
```

Según los resultados de los ataques, el contenido de la respuesta HTTP cambiará o seguirá siendo el mismo. Esto permite que un atacante infiera si la carga útil utilizada devolvió verdadero o falso, aunque no se devuelvan datos de la base de datos. Estos ataques suelen ser lentos (especialmente en bases de datos grandes) ya que un atacante necesitaría enumerar una base de datos, carácter por carácter.

f) Ataques de inyección de código SQL inferenciales basado en el tiempo

El propósito de los ataques de inyección de código SQL inferenciales basados en el tiempo es detener la base de datos un período de tiempo específico y luego devolver los resultados, lo que indica que la consulta SQL se ejecutó de forma correcta.

- Enlace utilizado:

  http://localhost:3434/groups.php?catalogo=1?

- Consulta enviada a la base de datos:

```
SELECT * FROM gestion.ges_catalogo
WHERE cat_id=1;
```

- El ataque simulado fue realizado mediante el uso de la expresión:

  http://localhost:3434/users.php?name=gabriela pg_sleep(15) --+

- Nueva enviada a la base de datos:

```
SELECT * FROM gestion.ges_catalogo
WHERE cat_id=1-SLEEP(15);
```

Estos tipos de inyección de código SQL se pueden identificar mediante las siguientes expresiones: “SLEEP ()”, BENCHMARK (), “WAIT FOR DELAY”.

g) Ataques de inyección de código SQL basados en procedimientos almacenados

El propósito de los ataques de inyección de código SQL basados en procedimientos almacenados es integrar códigos de inyección de código SQL maliciosos en los procedimientos almacenados existentes en las bases de datos.

- Enlace utilizado:

  http://localhost:3434/groups.php?name=gabriela?

- Consulta enviada a la base de datos:

```
CREATE OR REPLACE FUNCTION ejemplo()
RETURNS integer AS $$
BEGIN
SELECT *FROM gestion.ges_usuario WHERE usr_nombre = ''
AND usr_password = ''; END;
$$ LANGUAGE plpgsql; SELECT ejemplo();
```

- El ataque simulado fue realizado mediante el uso de la expresión:

  *http://localhost:3434/users.php?name=gabriela and select *from gestion.ges_usuario where usr_nombre = '' and usr_password = ''; drop table gestion.ges_usuario --+*

- Nueva consulta enviada a la base de datos:

```
CREATE OR REPLACE FUNCTION ejemplo()
RETURNS integer AS $$
BEGIN
SELECT *FROM gestion.ges_usuario
WHERE usr_nombre = '' AND usr_password = ''; DROP TABLE
gestion.ges_usuario;--'
END;
$$ LANGUAGE plpgsql;
```

El atacante introduce la sentencia drop dentro de la consulta del procedimiento almacenado alterando la estructura inicial de la misma.

h) Ataques de inyección de código SQL basados en codificación alternativa

El atacante utiliza distintas codificaciones como: hexadecimal, ASCII o Unicode para confundir a la base de datos en las sentencias SQL evitando así la validación básica.

El propósito de estos ataques de inyección de código SQL es modificar la consulta mediante codificación alternativa como: hexadecimal, ASCII y Unicode, lo que permite al atacante escapar del filtro del desarrollador que escanea las consultas de entrada en busca de un "carácter malo" especial conocido.

- Enlace utilizado:

  http://localhost:3434/users.php?name=legalUser?

- Consulta enviada a la base de datos:

```
SELECT usr_id FROM gestion.ges_usuario
WHERE usr_nombre= 'legalUser';
```

- El ataque simulado fue realizado mediante el uso de la expresión:

  http://localhost:3434/users.php?name=legalUser?;

  exec(CHAR(0x73687574646f776e)) -- AND usr_password=" --+

- Nueva consulta enviada a la base de datos:

```
SELECT usr_id FROM gestion.ges_usuario
WHERE usr_nombre= 'legalUser';
exec(CHAR(0x73687574646f776e)) -- AND usr_password=''
```

Estos tipos de inyección de código SQL se pueden identificar mediante las siguientes expresiones: “exec()”, “char()”, “ascii()”.

Finalmente, en la Figura 25 se visualiza el primer resultado de análisis de texto que realiza el modelo. Aquí aseguramos que cada consulta de los registros de transacciones identificada como anomalías es analizada por el modelo. Los datos que se muestran son:

el identificador del registro de transacción confirmado como un ataque, el tipo de ataque detectado y la consulta asociada a este.

```
(base) MacBook-Pro% python CuartoPaso.py
-----ALERTA-----
LOG_ID = 62
INYECCIÓN TIPO 4 DETECTADA: Piggy Backed
 select * from gestion.ges_bodega where bod_nombre='BODEGA1'; AND bod_codigo='';
 DROP table gestion.ges_usuario;
-----ALERTA-----
LOG_ID = 178
INYECCIÓN TIPO 3 DETECTADA: Basado en el Tiempo
 select *from gestion.ges_catalogo where cat_id=1-SLEEP(15);
-----ALERTA-----
LOG_ID = 362
INYECCIÓN TIPO 5 DETECTADA: Inferencia
 Declare @s varchar (8000); select @s = prueba (); if (ascii (substring (@s, 1,
1)) & (power (2, 0))) > 0waitfor delay '0:0:5'
-----ALERTA-----
LOG-ID = 513
INYECCIÓN TIPO 2 DETECTADA: Tautologia
 select * from gestion.ges_usuario where usr_nombre = 'GABRIELA' and usr_passwor
d = '' OR '1=1';
-----ALERTA-----
LOG-ID = 522
INYECCIÓN TIPO 1 DETECTADA: Operador Union
 select gru_nombre, gru_cod, gru_nivel from activo.grupo where gru_id=5 union se
lect prov_ruc, prov_razon_social, 22 from activo.proveedor where 1=1;
```

Figura 25. Resultado de la ejecución de los ataques de inyección de código SQL

3.1.5.2. Rendimiento

La precisión de los modelos de aprendizaje automático se puede medir desde distintos enfoques teóricos; sin embargo, en la mayoría de las aplicaciones comerciales, se necesita asignar un valor comercial a 4 tipos de resultados: verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos. Al multiplicar el número de resultados en cada segmento con los valores comerciales asociados, se asegurará de utilizar el mejor modelo disponible.

- Verdadero positivo: la predicción es correcta y el valor real es positivo (es decir, este valor es uno de los valores que el modelo estaba tratando de identificar. En el caso de un modelo que busca identificar clientes que probablemente comprarán el producto, este punto de datos representa un comprador potencial)
- Falso positivo: la predicción es incorrecta y el valor real es positivo
- Verdadero negativo: la predicción es correcta y el valor real es negativo (es decir, este valor no es uno de los valores que el modelo intentaba identificar.
- Falso negativo: la predicción es incorrecta y el valor real es negativo (Rendon & Acosta, 2006).

Para terminar, después de establecer un modelo de clasificación con datos de prueba definidos, el siguiente paso fue averiguar qué tan efectivo es el algoritmo implementado para identificar correctamente los registros de transacciones que dejan rastros de una inyección de código SQL. Para esto, se simularon 50 ataques de inyección de código SQL sobre nuestro servidor con base PostgreSQL. El resultado fue la obtención de 100 registros de transacciones; 50 teniendo inyecciones de código SQL registradas en las consultas de cada registro de transacción y 50 registros de transacciones con comportamiento 'normal' sin ningún indicio de ataque.

Los resultados del rendimiento obtenidos se pueden visualizar en la Tabla 9.

Tabla 9. Resultados del Rendimiento del Modelo de Clasificación

	Predicción	
Resultado del modelo	Registro de Transacción de Ataque	Registro de Transacción Normal
Positivo	**VP** 84%	**FP** 10%
Negativo	**FN** 16%	**VN** 90%

Las ecuaciones 3 y 4 indican el porcentaje de la exactitud obtenida con los 100 registros de transacciones de prueba ingresados.

$$\boldsymbol{Exactitud}: \quad \frac{\boldsymbol{VP + VN}}{\boldsymbol{VP + FP + FN + VN}} * \mathbf{100\%} = \mathbf{87\%} \qquad (3)$$

$$\boldsymbol{Exactitud}: \quad \frac{\mathbf{84 + 90}}{\mathbf{84 + 10 + 16 + 90}} * \mathbf{100\%} = \mathbf{87\%} \qquad (4)$$

Las ecuaciones 5 y 6 indican el porcentaje de la precisión obtenida con los 100 registros de transacciones de prueba ingresados.

$$\boldsymbol{Precisión}: \quad \frac{\boldsymbol{VP}}{\boldsymbol{VP + FP}} * \mathbf{100\%} = \mathbf{89.36\%} \qquad (5)$$

$$\boldsymbol{Precisión}: \quad \frac{\mathbf{84}}{\mathbf{84 + 10}} * \mathbf{100\%} = \mathbf{89.36\%} \qquad (6)$$

Las ecuaciones 7 y 8 indican el porcentaje de la sensibilidad obtenida con los 100 registros de transacciones de prueba ingresados.

$$\boldsymbol{Sensibilidad}: \quad \frac{\boldsymbol{VP}}{\boldsymbol{VP\ + FN}} * \mathbf{100\%} = \mathbf{\ 84\%} \qquad (7)$$

$$\boldsymbol{Sensibilidad}: \quad \frac{\mathbf{84}}{\mathbf{84\ + 16}} * \mathbf{100\%} = \mathbf{\ 84\%} \qquad (8)$$

Los valores evaluados reflejan que:

- De 50 ataques conocidos en el conjunto de registros de transacciones, 42 fueron identificados por nuestro modelo, mientras que 8 fueron descartados como comportamiento normal. Aunque el modelo presenta una exactitud deseable, en un conjunto grande de datos podría dejar varios ataques sin ser identificados.
- Los resultados porcentuales de la exactitud, precisión y sensibilidad están casi parejos. Por lo que podemos reiterar que el modelo tiene un bajo número de Falsos Positivos, al superar valores del 80%, y que el modelo implementado tiene un rendimiento aceptable.

3.1.6. Despliegue

La última fase de la metodología CRISP-DM tiene como objetivo explicar al cliente como poner en funcionamiento el proyecto que se ha construido en las fases anteriores. En este proyecto hemos optado por utilizar la metodología en cascada, con el objetivo de entregar un prototipo de sistema para la visualización de los resultados del modelo establecido por la metodología CRISP-DM.

3.2. DESARROLLO DEL PROTOTIPO DE SISTEMA

Para finalizar con el despliegue del modelo establecido con la metodología previa, a continuación, se listan los hitos realizados para cumplir con el objetivo de la implementación del prototipo de sistema.

3.2.1. Requerimientos

En la actualidad el CSIRT-EPN administra las infraestructuras críticas de TI de la EPN, teniendo acceso a los servidores que alojan las aplicaciones y sus gestores de base de datos, en donde se han identificado varios registros de ataques de inyección de código SQL.

El proceso de identificación de ataques que se maneja en la actualidad es mediante el análisis de los registros de registros de transacciones generados por el DBMS PostgreSQL. A pesar de haber obtenido una solución para la identificación de ataques mediante la metodología previa, es importante entregar un sistema que tenga una funcionalidad y usabilidad acorde con los requerimientos de la CSIRT-EPN.

3.2.1.1. Levantamiento de requerimientos del sistema

El proceso de levantamiento de requerimientos del sistema se llevó a cabo de varias reuniones con el CSIRT-EPN, en donde se mantuvo una serie de entrevistas con el equipo de trabajo e investigación del área, y después de discutir las actividades que se llevarían a

cabo dentro del sistema, se logró establecer un conjunto de requerimientos necesarios para la implementación del sistema.

a) Funcionalidades del sistema

- **Autenticación de usuarios**

F1: El sistema contará con una autenticación para ingresar a la aplicación.

La autenticación de usuarios consiste en la verificación del usuario que ingresa al sistema por medio de las credenciales establecidas.

- **Registro de usuarios**

F2: El sistema será capaz de registrar usuarios.

El registro de usuarios se refiere a la creación de usuarios dentro del sistema.

- **Ingreso de registros de transacciones**

F3: El sistema será capaz de ingresar los archivos de registros de transacciones.

El ingreso de registros de transacciones se refiere a la acción del ingreso de un archivo .log por medio de la interfaz web.

- **Generación de resultados de prevención**

F4: El sistema será capaz de generar los resultados de la herramienta de prevención.
La generación de resultados servirá para: identificar las tablas, variables y/o campos que sean susceptibles a futuros ataques de inyección de código SQL.

b) Requerimientos funcionales

- **Diagrama general**

La Figura 26 presenta el Diagrama de Casos de Uso General e indica las 4 actividades que puede hacer un usuario en el sistema de visualización.

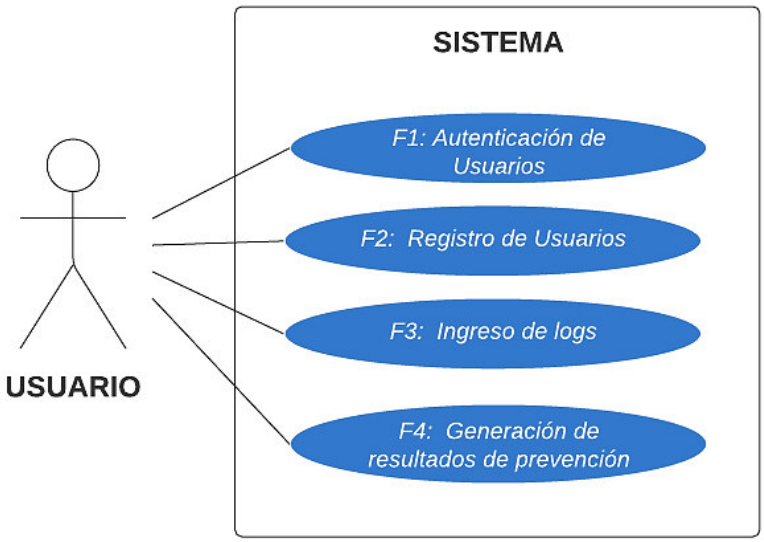


Figura 26. Diagrama de Casos de Uso General

- **Diagramas de casos de uso**

F1: Autenticación de usuarios

La Figura 27 muestra el Diagrama de Casos de Uso para autenticación de los usuarios en el sistema.

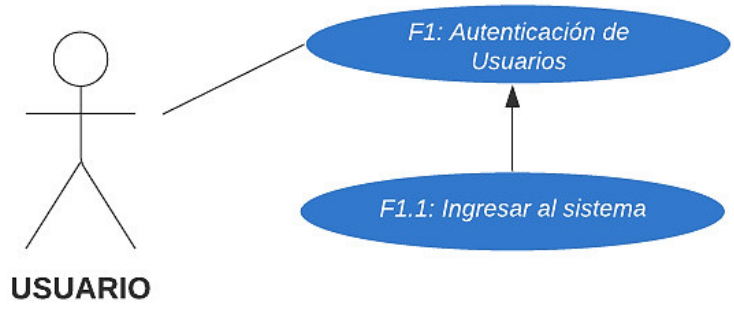


Figura 27. Diagrama de Casos de Uso de Autenticación de usuario

DETALLE:

F1.1: Ingresar al Sistema

Descripción: Se podrá ingresar al sistema por medio de un correo electrónico y una contraseña registrada.

Actores: Usuario del sistema

Flujo Principal:

La Tabla 10 detalla el proceso que debe realizar un usuario para ingresar al sistema.

Tabla 10. Flujo de caso de uso de Ingreso al sistema				
Paso	**Actor**	**Paso**	**Sistema**	**Excepción**
1	Inicia la aplicación	2	Presenta pantalla de registro de usuario y contraseña	
3	Ingresa el correo electrónico			
4	Ingresa contraseña del usuario			
5	Pulsa el botón Iniciar Sesión	6	Valida los datos ingresados	E1, E2, E3
		7	Presenta pantalla inicial de la aplicación	

Excepciones:

La Tabla 11 contiene las excepciones que pueden existir y no permitan al usuario el ingreso al sistema.

Tabla 11. Excepciones de caso de uso de Ingreso al sistema		
Código	**Mensaje**	**Alternativa**
E1	Datos obligatorios vacíos	Llenar los campos restantes
E2	Datos ingresados incorrectos	Corregir los datos incorrectos y volver a intentar
E3	Error en la base de datos	Cerrar la aplicación y volverla a abrir

F2: Registro de Usuarios

La Figura 28 muestra el Diagrama de Casos de Uso para Registros de Usuarios en el sistema la cual despliega dos opciones: Crear Usuario y Recuperar Contraseña.

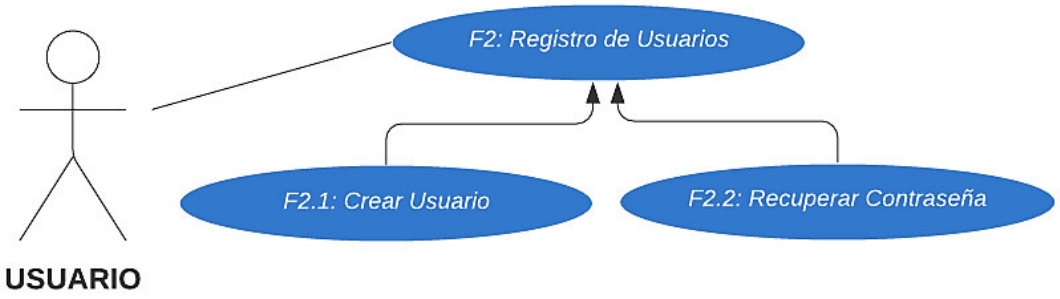


Figura 28. Diagrama de Casos de Uso Registro de Usuarios

DETALLE:

F2.1: Crear Usuario

Descripción: Se podrá crear los usuarios que tendrán acceso al sistema.

Autores: Usuario del Sistema

Flujo Principal:

La Tabla 12 detalla el proceso a realizar en el sistema para crear un nuevo usuario.

Tabla 12. Flujo de caso de uso de Creación de Usuario

Paso	Actor	Paso	Sistema	Excepción
1	Inicia la aplicación	2	Presenta pantalla de registro de usuario y contraseña	
3	Pulsa el botón Registrarse	4	Presenta pantalla de registro de usuario	
5	Ingresa correo electrónico			
6	Ingresa contraseña			
7	Ingresa confirmación de contraseña			
8	Pulsa el botón Registrarse	9	Verifica los datos ingresados	E1, E2, E3
		10	Presenta un mensaje que el usuario se ha creado exitosamente	

Excepciones:

La Tabla 13 indica las excepciones que pueden existir y no permitan la creación de un usuario nuevo.

Tabla 13. Excepciones de caso de uso de Creación de Usuario		
Código	**Mensaje**	**Alternativa**
E1	Datos obligatorios vacíos	Llenar los campos restantes
E2	Datos ingresados incorrectos	Corregir los datos incorrectos y volver a intentar
E3	Error en la base de datos	Cerrar la aplicación y volverla a abrir

F2.2: Recuperar Contraseña

Descripción: Se podrá modificar la contraseña del usuario registrado en el sistema.

Autores: Usuario del Sistema

Flujo Principal:

La Tabla 14 detalla el proceso a realizar en el sistema para la recuperación de contraseña.

Tabla 14. Flujo de caso de uso de Recuperación de Contraseña				
Paso	**Actor**	**Paso**	**Sistema**	**Excepción**
1	Inicia la aplicación	2	Presenta pantalla de registro de usuario y contraseña	
3	Pulsa el botón Recuperar Contraseña	4	Presenta pantalla de recuperación de contraseña	
5	Ingresa correo electrónico			
6	Pulsa botón Enviarme las instrucciones de reseteo de Contraseña	7	Envía el correo electrónico con el enlace de recuperación de contraseña	
8	Pulsa enlace de Cambio de Contraseña	9	Presenta pantalla de cambio de contraseña	
10	Ingresa nueva contraseña			
11	Ingresa confirmación de nueva contraseña			
12	Pulsa el botón Cambiar Contraseña	13	Verifica los datos Ingresados	E1, E2, E3
		14	Presenta un mensaje que la contraseña se ha cambiado con éxito	

Excepciones:

La Tabla 15 indica las excepciones existentes para la recuperación de contraseña.

Tabla 15. Excepciones de uso de Recuperación de Contraseña		
Código	**Mensaje**	**Alternativa**
E1	Datos obligatorios vacíos	Llenar los campos restantes
E2	Datos ingresados incorrectos	Corregir los datos incorrectos y volver a intentar
E3	Error en la base de datos	Cerrar la aplicación y volverla a abrir

F3: Ingreso de registros de transacciones

La Figura 29 muestra el Diagrama de Casos de Uso para ingresar el archivo de registros de transacciones.

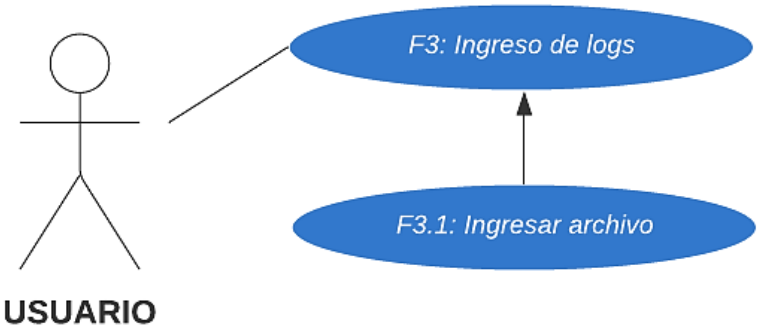


Figura 29. Diagrama de Casos de Uso Ingreso de Registros de Transacciones

DETALLE:

F3.1: Ingresar archivo

Descripción: Se podrá enviar un archivo de registros de transacciones al servidor que albergue al sistema.

Autores: Usuario del Sistema

Flujo Principal:

La Tabla 16 detalla el proceso a realizar en el sistema para el ingreso de un archivo.

Tabla 16. Flujo de caso de uso de Ingreso de Archivo				
Paso	**Actor**	**Paso**	**Sistema**	**Excepción**
1	Selecciona opción Ingresar Logs	2	Presenta pantalla de ingreso de archivo	
3	Pulsa el botón Examinar	4	Presenta la ventana de búsqueda de archivo	
5	Selecciona el archivo			
6	Pulsa el botón Abrir	7	Presenta la pantalla de ingreso de archivo con archivo escogido	
8	Pulsa el botón Cargar Archivo	9	Verifica el archivo ingresado	E1, E2, E3
		10	Presenta un mensaje que el archivo se ha ingresado con éxito	

Excepciones:

La Tabla 17 indica las excepciones que pueden existir y no permitan el ingreso de un archivo al sistema.

Tabla 17. Excepciones de caso de uso de Ingreso de Archivo		
Código	**Mensaje**	**Alternativa**
E1	Datos obligatorios vacíos	Llenar los campos restantes
E2	Datos ingresados incorrectos	Corregir los datos incorrectos y volver a intentar
E3	Error en la base de datos	Cerrar la aplicación y volverla a abrir

F4: Generación de resultados de prevención

La Figura 30 muestra el Diagrama de Casos de Uso para la Generación de resultados de prevención que permite identificar si un enlace puede ser o no utilizado en la herramienta de prevención que tiene el sistema de visualización.

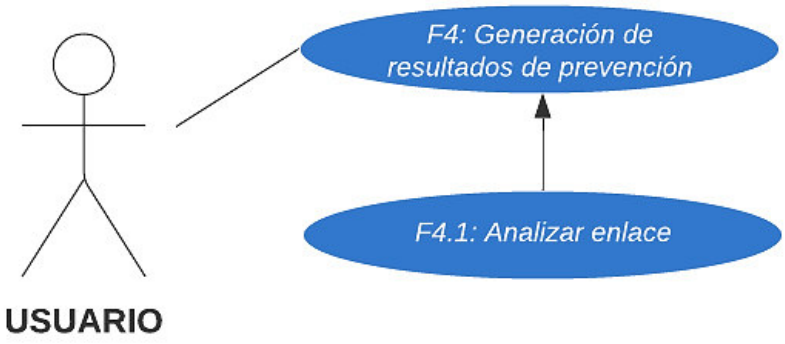


Figura 30. Diagrama de Casos de Uso Generación de resultados de prevención

DETALLE:

F4.1: Analizar enlace

Descripción: Se podrá analizar el enlace de un servidor web para que se analizado por la herramienta de prevención implementada en el sistema.

Autores: Usuario del Sistema

Flujo Principal:

La Tabla 18 detalla el proceso a realizar en el sistema para el análisis del enlace web.

Tabla 18. Flujo de caso de uso de Análisis del Enlace Web

Paso	Actor	Paso	Sistema	Excepción
1	Selecciona la pestaña Herramienta de Análisis	2	Presenta pantalla de Herramienta de Análisis	
3	Ingresa el enlace de la página web			
4	Pulsa el botón Analizar	5	Verifica el enlace ingresado	E1, E2, E3
		6	Presenta los resultados de análisis del enlace	

Excepciones:

La Tabla 19 indica las excepciones que pueden existir y no permitan el análisis del enlace web.

Tabla 19. Excepciones de uso de Análisis del Enlace Web		
Código	**Mensaje**	**Alternativa**
E1	Datos obligatorios vacíos	Llenar los campos restantes
E2	Datos ingresados incorrectos	Corregir los datos incorrectos y volver a intentar
E3	Error en la base de datos	Cerrar la aplicación y volverla a abrir

c) Requerimientos no funcionales

- **Restricciones de diseño e implementación**
  - Como lenguaje de programación se utilizará Python 3 y Ruby.
  - Para el desarrollo del producto se utilizará el ambiente de desarrollo de "Ruby on Rails".
  - Se utilizará el paradigma de Programación Orientada a Objetos.
  - Se utilizará como Motor de Base de Datos a PostgreSQL.
- **Requerimientos Especiales**

La Figura 31 muestra el diagrama de los requerimientos no funcionales del sistema y su clasificación.

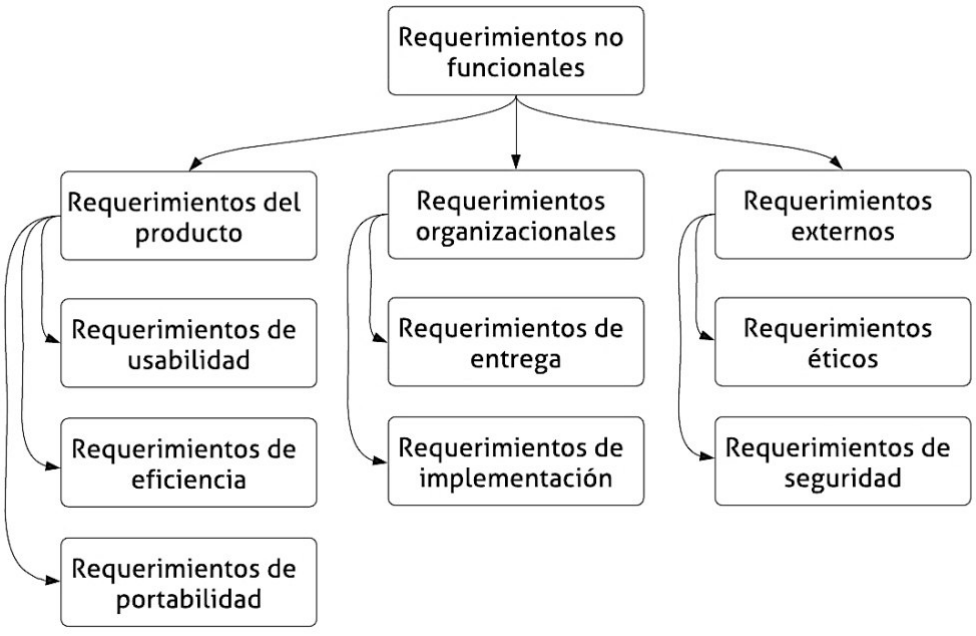


Figura 31. Cuadro de Requerimientos Especiales Implementados

Requerimientos del producto

- **Usabilidad**
  - En el sistema se puede visualizar que cada uno de sus módulos están bien descritos acordes a lo que se realiza en cada pantalla.
  - El formulario creado para el inicio de sesión, registro y recuperación de contraseña están apropiadamente controlados para que el tipo de dato ingresado vaya acorde con el formulario.
  - Los botones están claramente etiquetados acorde a la función que desempeñan.
  - Los mensajes de errores están definidos.
  - Todos estos elementos hacen que la interfaz sea intuitiva y fácil de usar por el usuario del sistema.
- **Eficiencia**
  - La principal fuente de medición del rendimiento del sistema recae en la carga de registros de transacciones para la ejecución del algoritmo de identificación de ataques. Estas pruebas serán llevadas a cabo en el último paso de la implementación de las metodologías CRISP-DM y cascada.
  - Los requerimientos de espacio necesario de disco duro para que el servidor funcione correctamente dependen de los programas necesarios, en este caso se recomienda un mínimo de 50MB libres para el correcto funcionamiento de la página web.
  - Es importante resaltar que, aparte de las pruebas de carga realizadas al algoritmo de identificación, la eficiencia del funcionamiento de la página web dependerá del servidor en el que este establecido. Si bien existen servidores que albergan páginas gratuitamente, su eficiencia frente al usuario final dependerá de los servicios que ese dispuesto a pagar la CSRIT-EPN.
- **Portabilidad**

El sistema al tener implementada una página web tiene una portabilidad variada, sin embargo, existe la herramienta de análisis (SQLMAP) la cual solo puede ser ejecutada desde el servidor en el que este implementada la página web.

Requerimientos organizacionales

- **Entrega**

El sistema al ser un prototipo funcional será presentado al CSIRT-EPN para definir su funcionamiento. En esta entrega también se entregarán manuales de usuario y

de instalación. Se procederá a realizar un ejemplo de todo o el proceso de ingreso y de análisis. Y se establecerá una prueba de usabilidad para documentación.

- **Implementación**

La unidad del CSIRT-EPN definió como estándar de implementación las maquetas de interfaces listadas en el literal 3.2.1.1.4.1. El ambiente de desarrollo y alojamiento del gestor de base de datos quedó a disposición de nosotros los desarrolladores de este proyecto.

Requerimientos externos

- **Éticos**

El desarrollo del sistema se lo realizó acoplándose a las necesidades impuestas por el CSRIT-EPN. También se estableció que los registros de transacciones entregados contienen información sensible de los usuarios de las aplicaciones administradas por el área, por lo que se firmó un acuerdo de confidencialidad por nosotros los autores de este proyecto,

- **Seguridad**

El sistema cuenta con una autenticación de usuarios con contraseña, dando acceso a las estadísticas desplegadas únicamente a usuarios autorizados.

3.2.1.2. Otros requerimientos

a) Interfaces graficas

Los bocetos de las interfaces del Prototipo de Sistema para la identificación y predicción de ataques a sistemas de bases de datos utilizando técnicas de minería de datos se muestran a continuación.Es importante mencionar que los bocetos han sido utilizados de manera referencial y no representan el diseño ni la funcionalidad final del prototipo.

La Figura 32 muestra el modelo de interfaz gráfica de registro de usuario que permite que una nueva persona se registre en el sistema.

Figura 32. Modelo de interfaz gráfica de registro de usuarios

El modelo de interfaz gráfica de inicio de sesión servirá de limitador para la visualización de los demás módulos mediante un control de credenciales como muestra la Figura 33. De igual manera, se habilitará la función de olvido de clave en caso de necesitarlo.

Figura 33. Modelo de interfaz gráfica de inicio de sesión

La Figura 34 muestra el modelo de interfaz para carga de un documento que permite seleccionar el archivo de texto que contiene un grupo de registros de transacciones.

Figura 34. Modelo de interfaz gráfica de carga de registros de transacciones

El modelo de interfaz resumen diario, desplegará información sobre los registros de ataques que hubo en el día. La información será desplegada mediante gráficos que permitirá el usuario visualizar la información correspondiente como se muestra en la Figura 35.

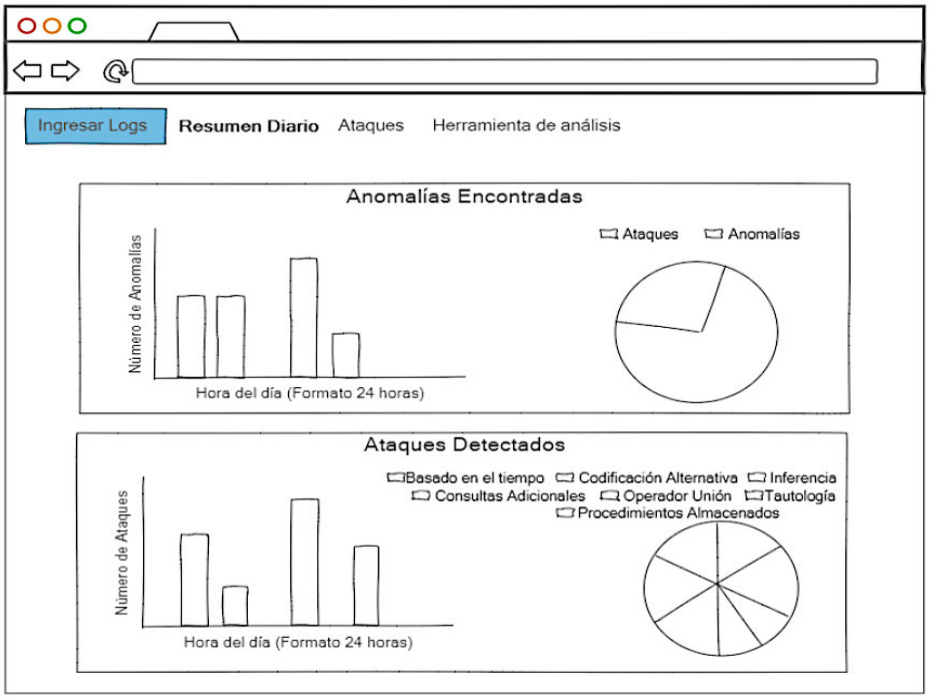


Figura 35. Modelo de interfaz gráfica de la pestaña Resumen Diario

El modelo de interfaz de ataques como se muestra en la Figura 36 desplegará una tabla de todos los registros que ha tenido el sistema, así como las alertas diarias en tiempo real que va detectando el algoritmo.

Figura 36. Modelo de interfaz gráfica de la pestaña Análisis y Predicción

La Figura 37 indica el modelo de interfaz de la herramienta de análisis que contendrá la herramienta SQLMAP, la cual podrá ser utilizada mediante el ingreso de una dirección web (*URL*) donde este el servidor de base de datos a analizar. La información desplegada del análisis de la dirección web se evidenciará mediante los resultados en consola que se podrán visualizar en la página web.

Figura 37. Modelo de interfaz gráfica de la pestaña Herramienta de análisis

b) Requisitos de Hardware y Software

- **Arquitectura del prototipo de sistema**

El prototipo de sistema para la detección y prevención de intrusos se realizó en base a la arquitectura cliente servidor. En la Figura 38 se detalla de la arquitectura del sistema.

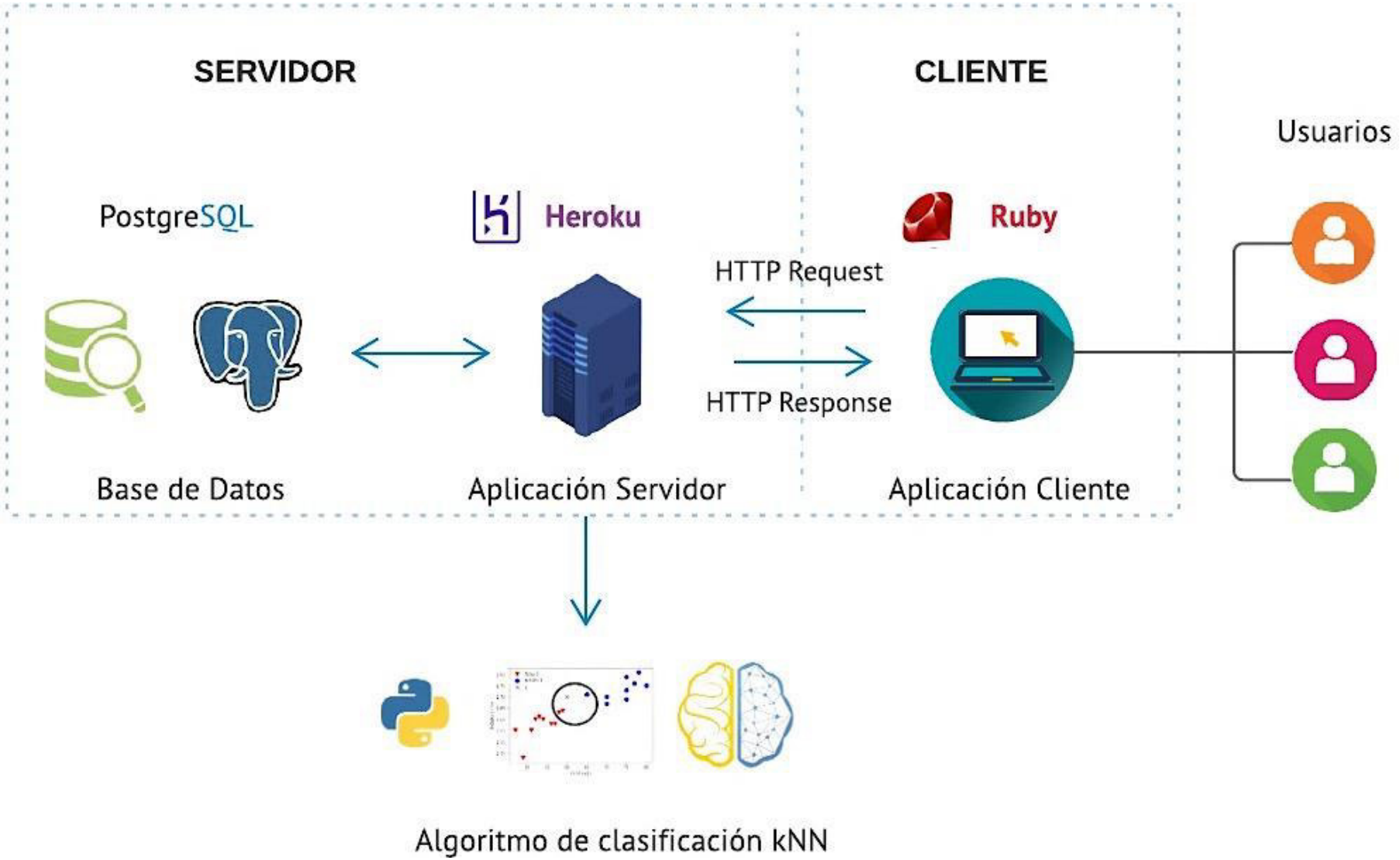


Figura 38. Arquitectura del Prototipo de Sistema

3.2.2. Diseño

3.2.2.1. Diagrama de clases

La Figura 39 muestra el Diagrama de clases utilizado previo a la creación del prototipo de sistema web.

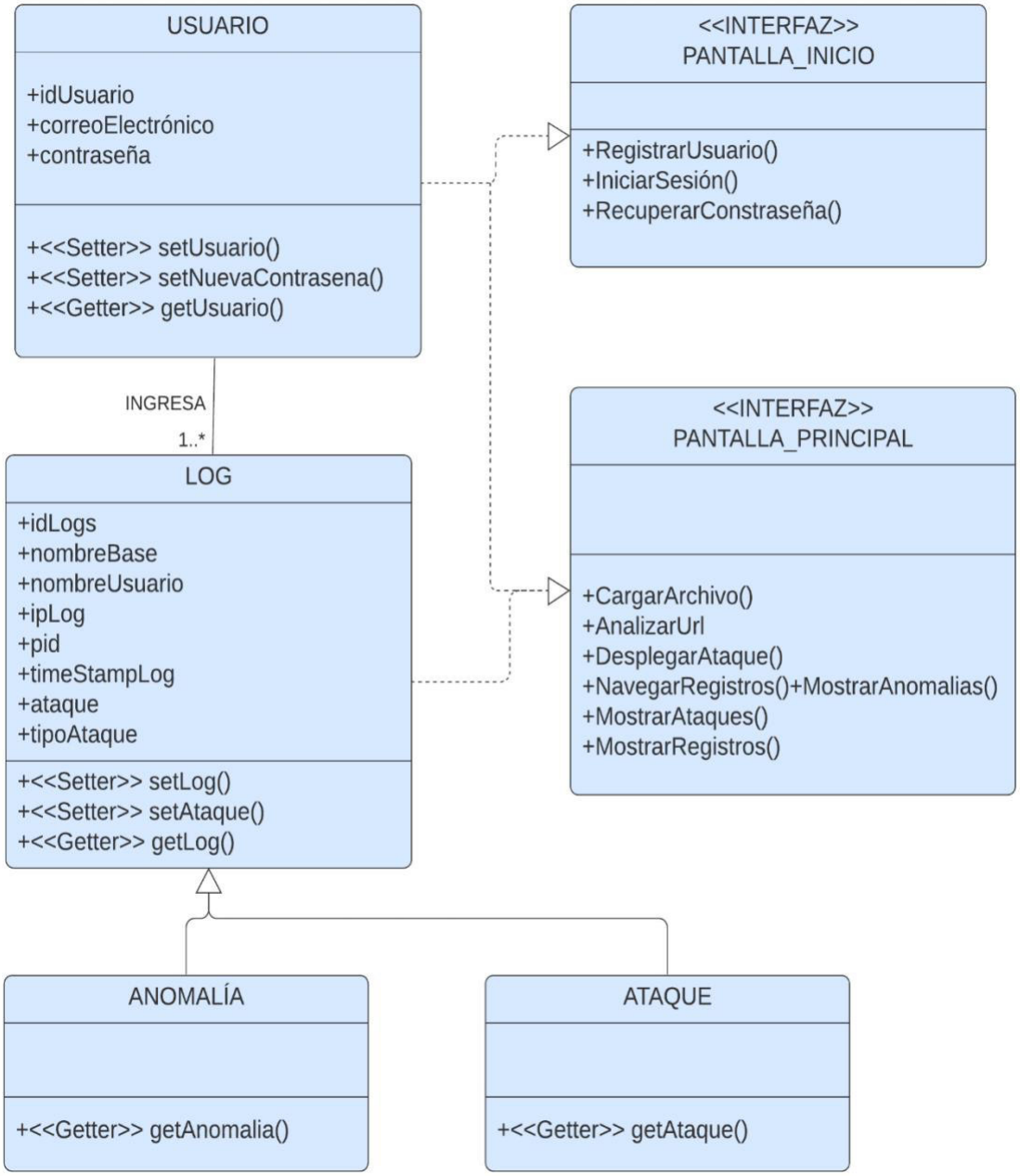


Figura 39. Diagrama de clases del Dominio del Problema

3.2.2.2. Diagrama de secuencia

F1: Autenticación de usuarios

F1.1: Ingresar al sistema

La Figura 40 muestra mediante un Diagrama de Secuencia el proceso que realiza el usuario para ingresar al sistema.

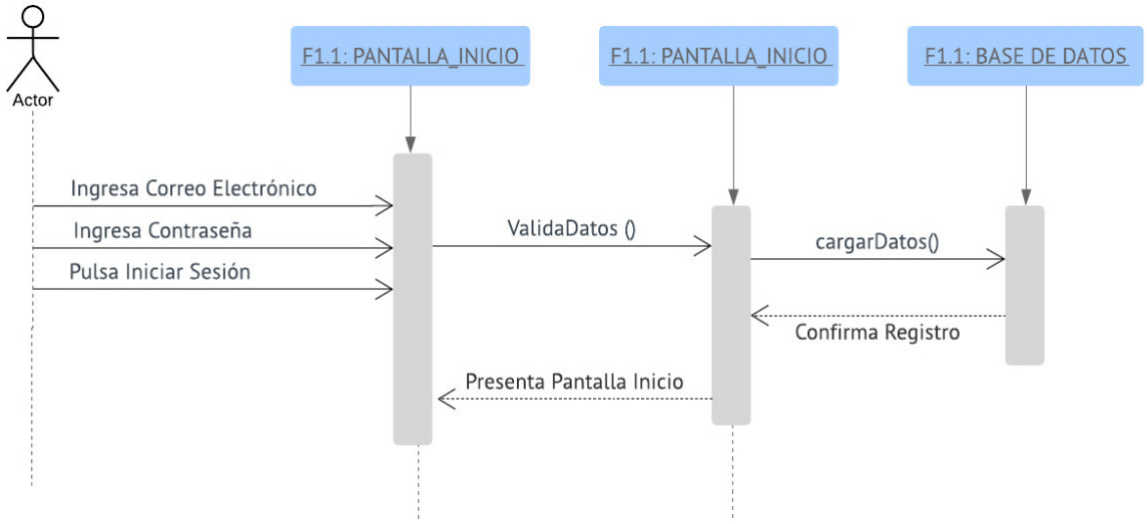


Figura 40. Diagrama de Secuencia de Ingreso al Sistema

F2: Registro de Usuarios

F2.1: Crear Usuario

La Figura 41 muestra el Diagrama de Secuencia para crear un nuevo usuario en el sistema.

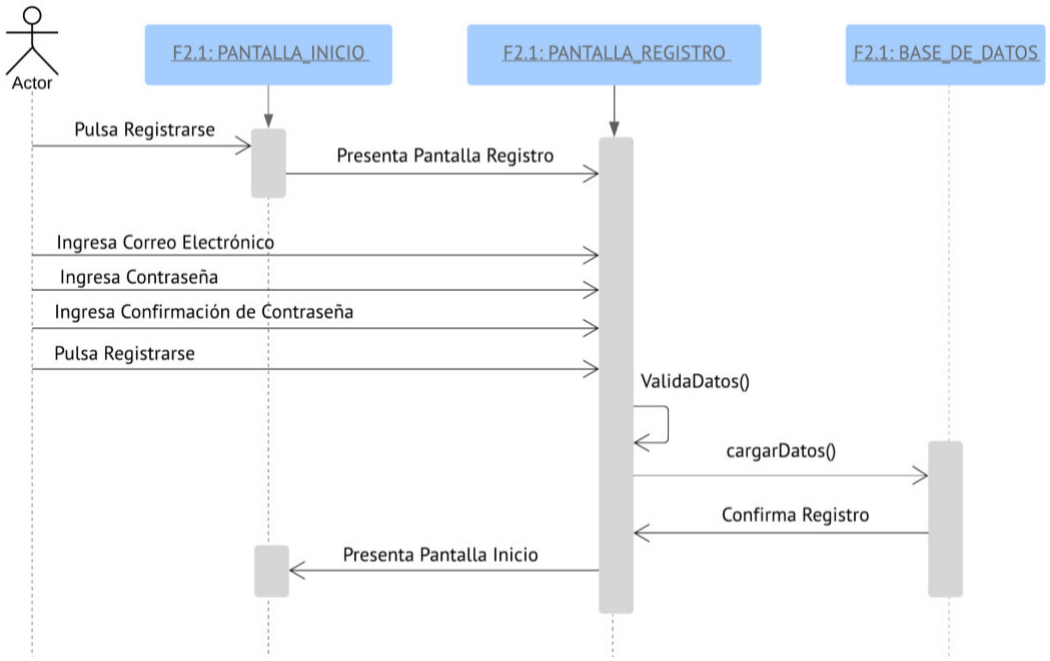


Figura 41. Diagrama de Secuencia Creación de Usuario

F2.2: Recuperar Contraseña

La Figura 42 muestra el Diagrama de Secuencia para la recuperación de contraseña.

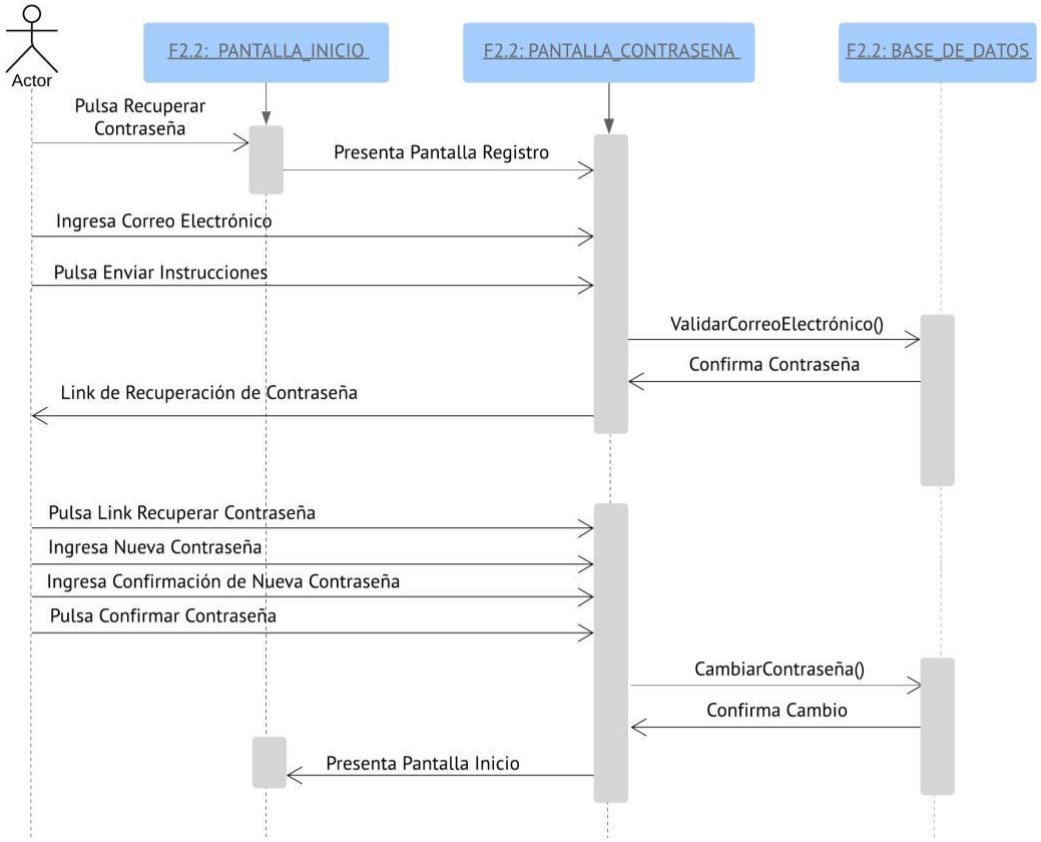


Figura 42. Diagrama de Secuencia Recuperación Contraseña

F3: Ingreso de registros de transacciones

F3.1: Ingresar archivo

La Figura 43 muestra el Diagrama de Secuencia para el ingreso de un archivo al sistema.

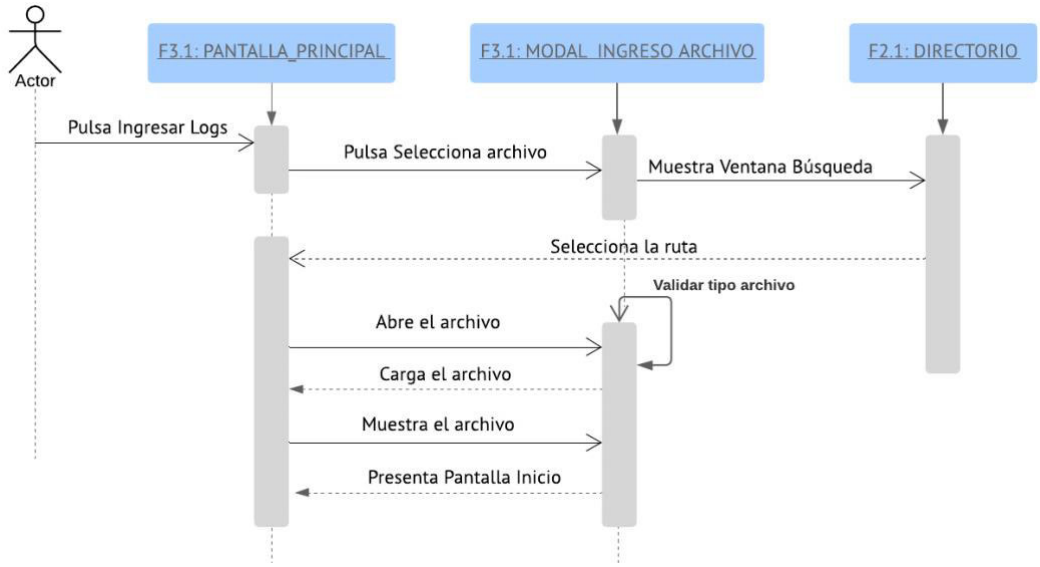


Figura 43. Diagrama de Secuencia Ingresar archivo

F4: Generación de resultados de prevención

F4.1: Analizar enlace

La Figura 44 muestra el Diagrama de Secuencia para la generación del análisis de un enlace web.

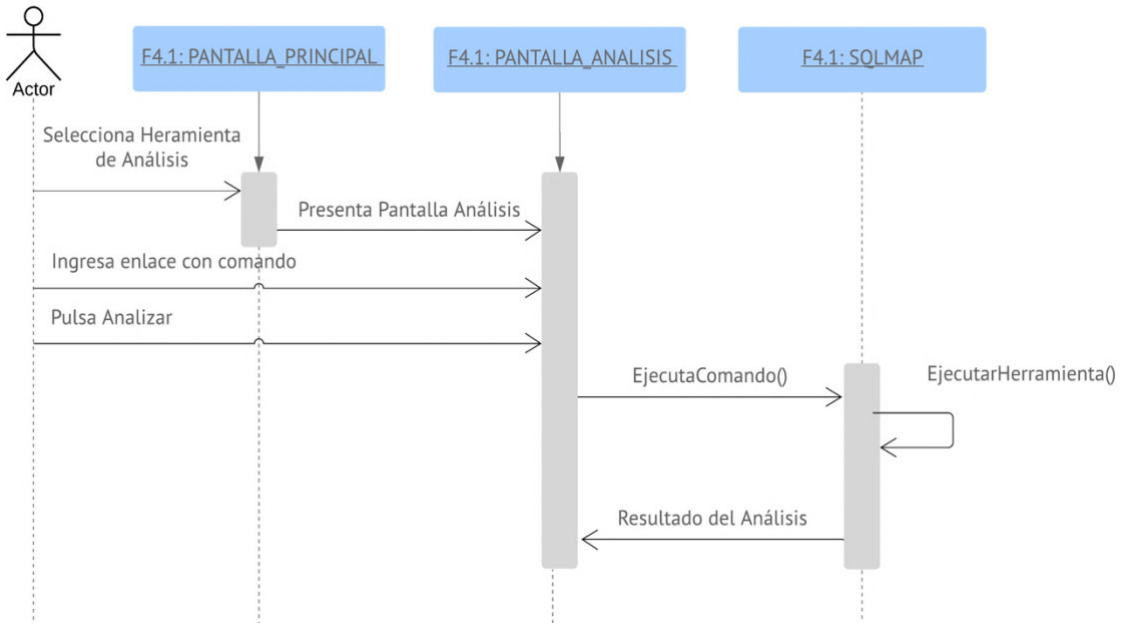


Figura 44. Diagrama de Secuencia Análisis de Enlace

3.2.2.3. Modelo Entidad Relación

La base de datos a implementar para el prototipo de la página web fue establecida de la siguiente manera, tal como se muestra en la Figura 45, teniendo 4 tablas que son las que se describen a continuación:

- USUARIO: La tabla “usuario” controla el inicio de sesión de la aplicación.
- ARCHIVO: La tabla “archivo” registra los datos asociados al archivo que se ingresa al servidor.
- REGISTRO_TRANSACCION: La tabla “registro_transaccion” registra los datos de las anomalías encontradas en los registros de transacciones.
- CONSULTA: La tabla “consulta” registra la secuencia SQL encontrada en cada registro de transacción.

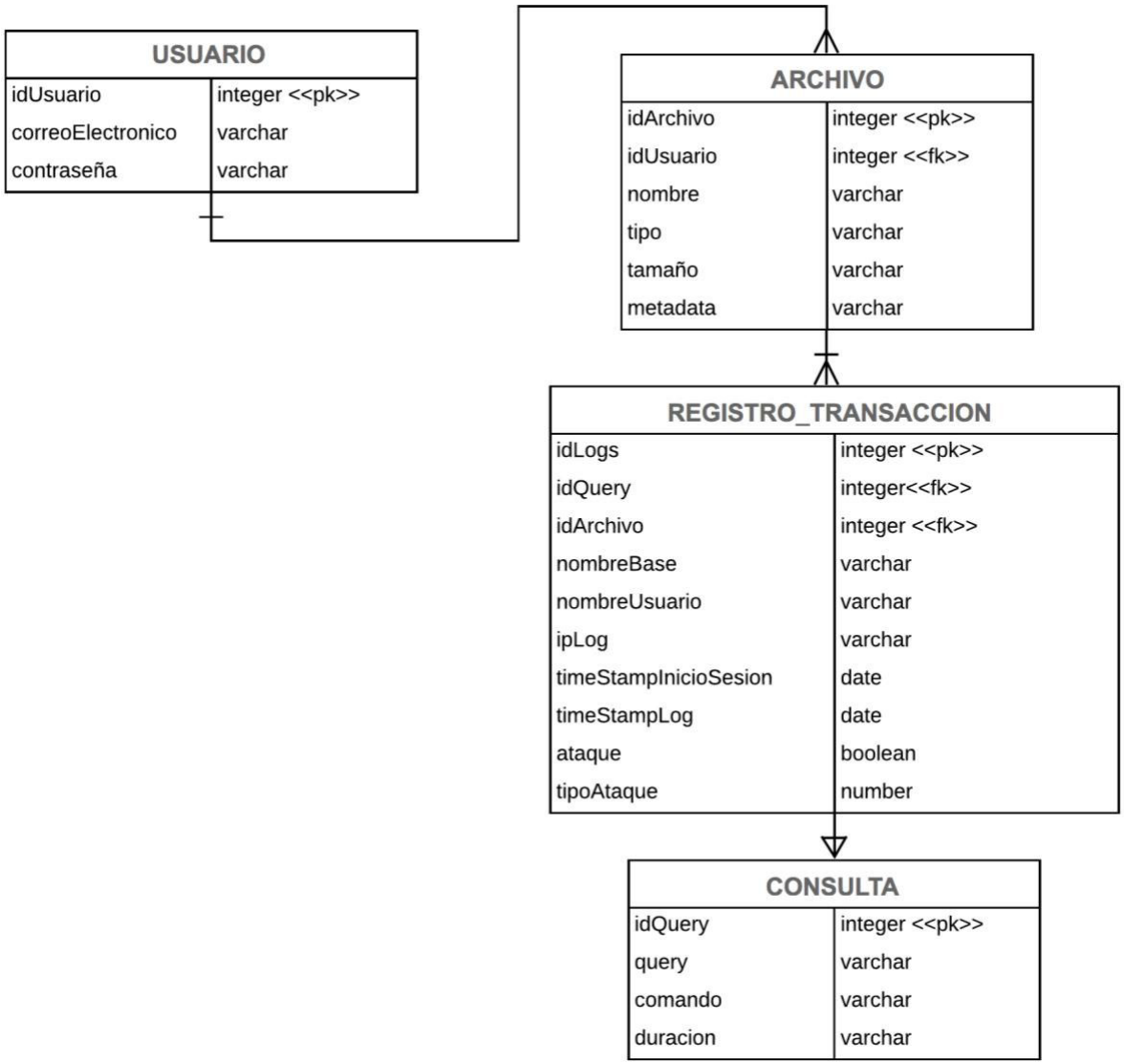


Figura 45. Diagrama de Entidad – Relación

3.2.2.4. Definición de estándares de diseño y desarrollo

Validación de formularios

- **Validación de campos obligatorios. -** todos los formularios constarán del ingreso de datos obligatorios y notificará al usuario en caso de no llenar alguno.
- **Validación de tipo de datos. -** todos los campos de texto no deberán permitir el ingreso de un dato que no concuerde con el tipo de dato de ingreso en el formulario.
- **Validación de ingreso de archivo. -** se deberá validar el tipo de archivo ingresado al formulario de ingreso de los registros de transacciones, así como también se validará que no se ingrese un archivo vacío o menor a 200kb.

Estándares de codificación

Se utilizarán los estándares definidos por 'Ruby on Rails':

- **Formato:**
  - Usar indentación o sangría de 2 espacios, sin el uso de Tabs.
  - Usar espacios alrededor de los operadores, después de las comas, después de dos puntos, después de punto y coma, alrededor de { y antes de }.
  - Mantener las líneas de código con menos de 80 caracteres.
- **Sintaxis:**
  - Usar def con paréntesis cuando tenga argumentos.
  - Usar && y || para expresiones booleanas.
  - Usar and y or para flujos de control.

Estándares interfaz gráfica

Se utilizarán componentes de Bootstrap para el despliegue de los resultados en las interfaces, así como también se seguirán los modelos establecidos en el literal 3.2.1.1.4.1.

Mensajes de error y mensajes

Se utilizarán dos tipos de cuadros de diálogo:

- **Cuadro de Diálogo de Error. -** reflejará todos los mensajes de error emitidos por el sistema,
- **Cuadro de Diálogo de Información.** - reflejará todos los mensajes con éxito y/o informativos orientados a la notificación de una acción al usuario.

3.2.3. Implementación

El propósito de esta fase es la creación de una interfaz web y del ambiente de desarrollo de acuerdo a los parámetros establecidos.

3.2.3.1. Inicio de sesión

La Figura 46 muestra la página de inicio de sesión del sistema web. Esta contiene los campos básicos de inicio de sesión, así como un botón para la recuperación de la clave del usuario en caso de necesitarlo.

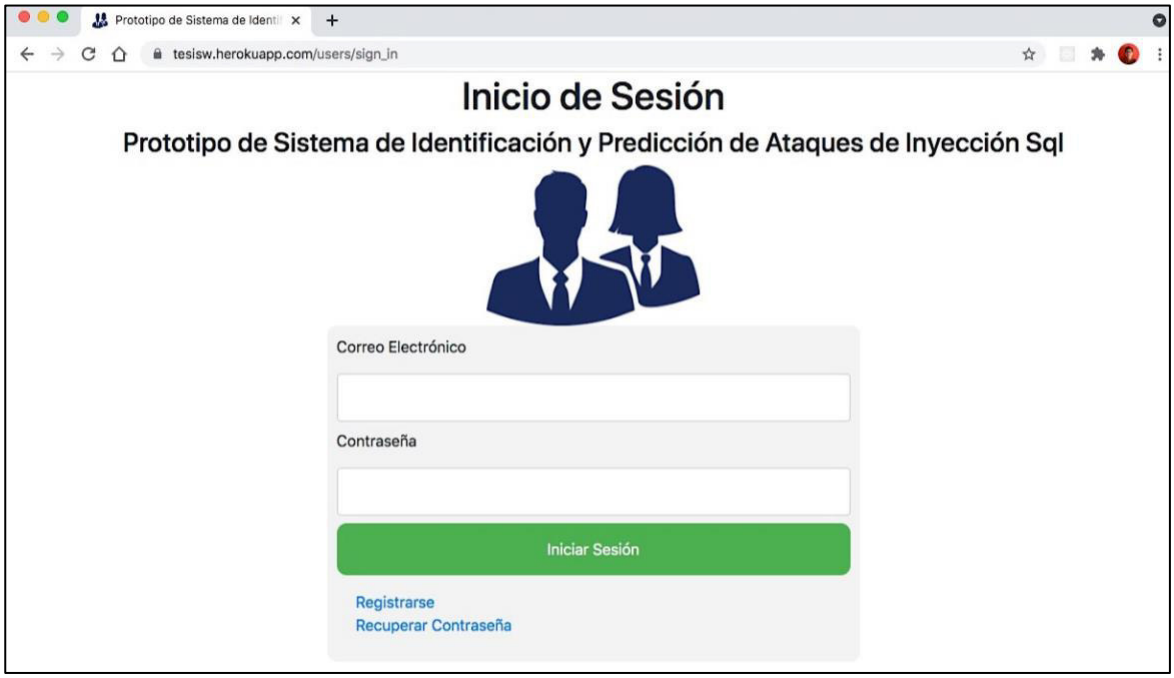


Figura 46. Interfaz Web del Inicio de Sesión

3.2.3.2. Pestaña: Resumen Diario

La Figura 47 muestra el menú que permite navegar entre cada uno de los módulos de la aplicación web. El primer módulo se encuentra divido en:

- Anomalías Detectadas: Visualización gráfica de los registros de transacciones que fueron identificados como anomalías. Estas gráficas explicarán la hora y la cantidad de anomalías detectadas, así como una división que diferencia las anomalías de los ataques identificados.
- Ataques Detectados: Visualización gráfica de las anomalías identificadas como ataques. Estas gráficas muestran los ataques por horas, así como una división de los tipos de ataques.

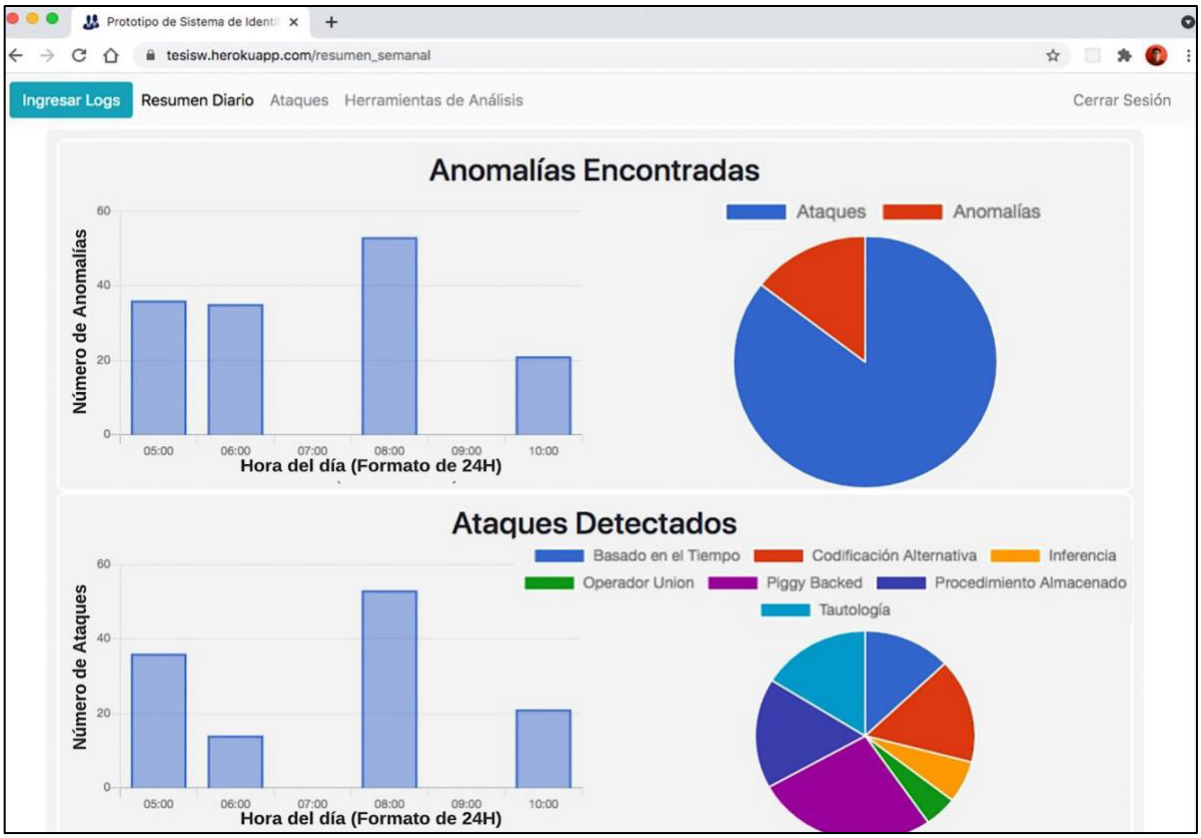


Figura 47. Interfaz Web de Resumen Diario

3.2.3.3. Pestaña: Registros de Ataques y Alertas

La Figura 48 muestra una tabla donde se visualiza los registros de todos los ataques identificados desde el inicio de funcionamiento de la aplicación. De igual manera, permite visualizar en tiempo real los ataques que van siendo detectados y datos asociados a los mismos.

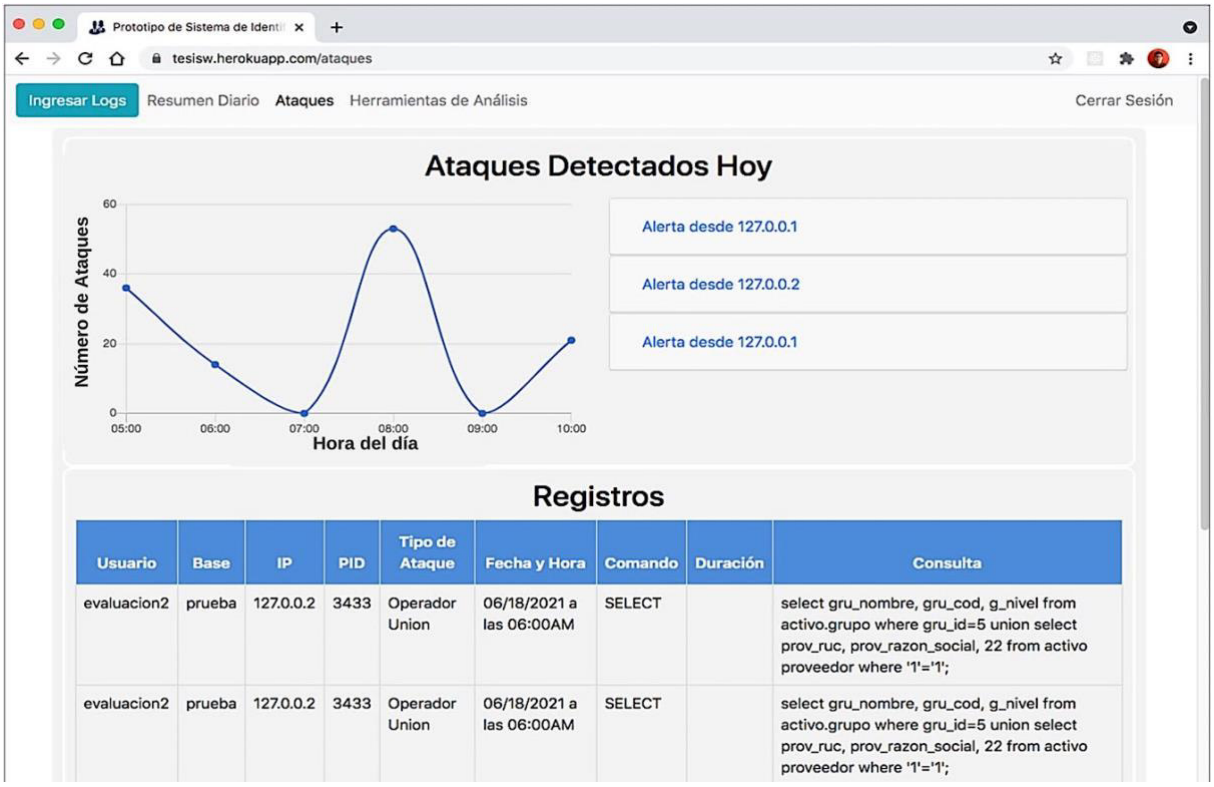


Figura 48. Interfaz Web de Ataques y Alertas

3.2.3.4. Pestaña: Herramientas de Prevención SQLMAP

Por último, la Figura 49 muestra la herramienta de análisis para la predicción de ataques en los que puede verse afectado un servidor. Los resultados de este análisis se muestran mediante la línea de comando implementada en la página donde se resaltará los posibles ataques detectados, así como datos de tablas, variables, etc.

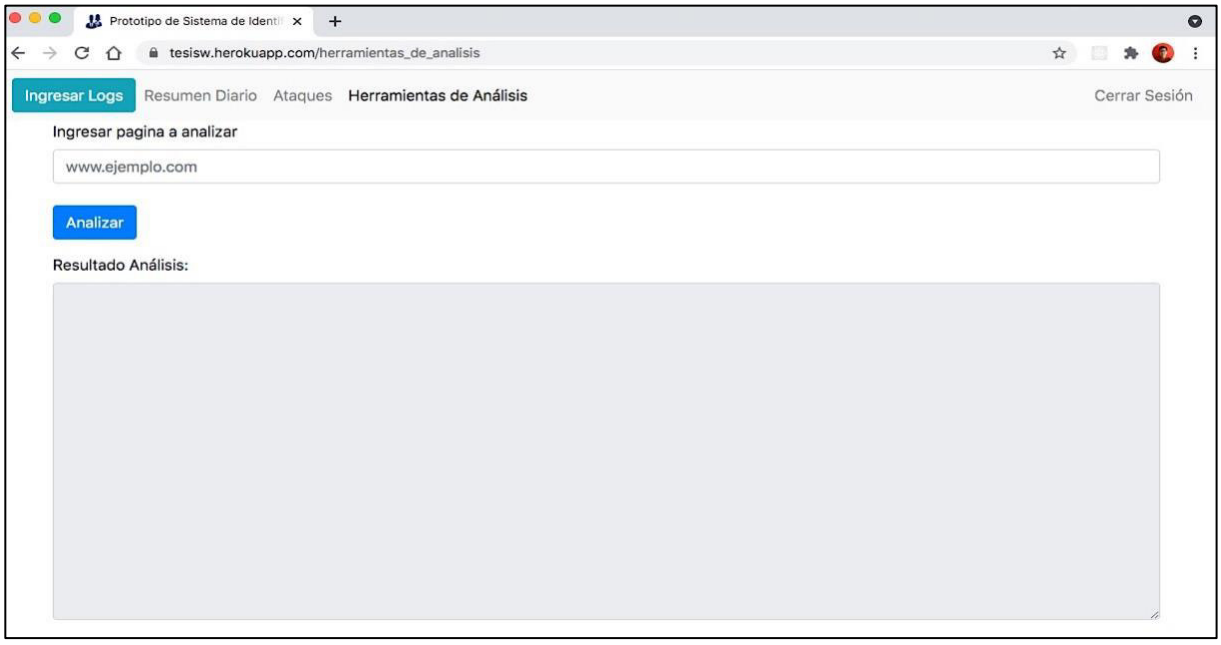


Figura 49. Interfaz Web de Ataques y Alertas

3.2.4. Verificación y Mantenimiento

Esta fase se complementa mediante el uso de pruebas funcionales, de carga y de usabilidad. La cuales estaremos nombrando en el literal 3.3.

3.3. PRUEBAS

3.3.1. Pruebas Funcionales

De forma paralela a la implementación, se prepararon una serie de casos pruebas unitarias para evaluar que el prototipo refleje los diferentes tipos de ataques en la interfaz web. Estas pruebas son evaluaciones basadas en la ejecución de una fracción del código y como los resultados enviados por la ejecución del mismo son reflejados en la interfaz.

Esta evaluación permite obtener una visión general de cómo se están reflejando los ataques identificados por el algoritmo en la interfaz.

3.3.1.1. Planificación de pruebas funcionales

Consiste en la ejecución de 25 a 60 ataques continuos, por cada tipo de inyección de código SQL, en el servidor de base de datos de prueba. Los logs generados por este servidor serán ingresados al prototipo desarrollado y se analizarán los resultados dependiendo del número de ataques identificados. Las pruebas unitarias serán ejecutadas durante un periodo controlado de tiempo de 60 minutos por el tipo de ataque.

3.3.1.2. Descripción del método

Consiste en una ejecución controlada de una secuencia de ataques continuos durante el periodo de cincuenta minutos por cada tipo. Los tipos de ataques a simular son:

- Tautología
- Consultas Ilegales / Lógicamente Incorrectas
- Consulta de Unión
- Consultas Adicionales
- Basado en Inferencia
- Basado en Procedimientos Almacenadas
- Codificación Alternativa
- Basado en el Tiempo

Finalmente, después de la ejecución de los ataques por 7 horas, se somete a un conteo general e interpretación del número de ataques realizados en comparación a los mostrados por la interfaz.

3.3.1.3. Realización de las pruebas unitarias

a) Tautología

En la Figura 50 se puede apreciar los resultados de los primeros ataques implementados. El número total de ataques detectados fueron 38 de 41, y se muestra en la Figura 51.

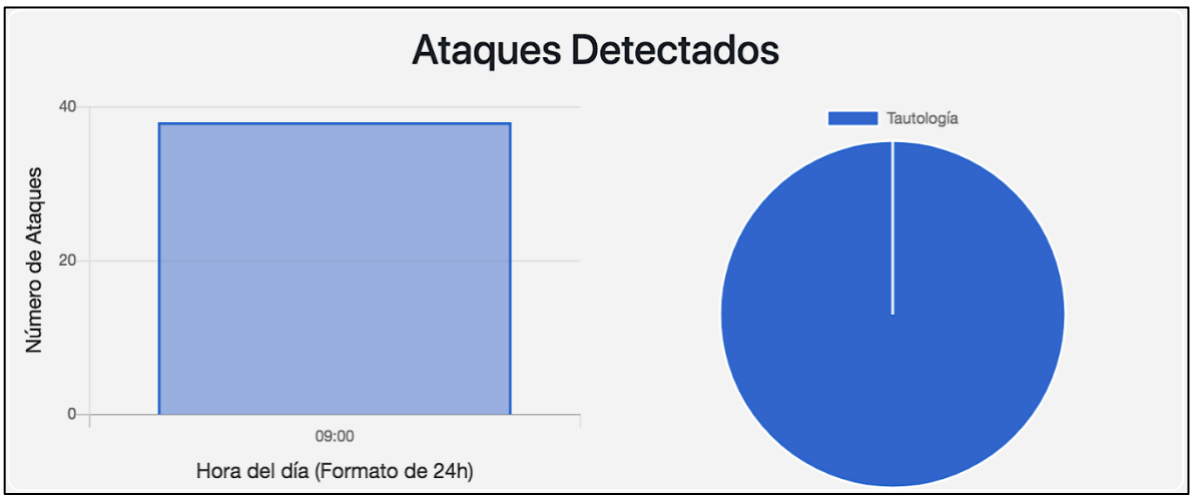


Figura 50. Resultado Inyección de código SQL Tautología en pestaña de Resumen Diario

Figura 51. Resultado Inyección de código SQL Tautología en pestaña Ataques

b) Consultas Ilegales

Este tipo de inyección de código SQL no puede ser identificado apropiadamente por el prototipo implementado ya que este tipo de ataque, al estar basado en mensajes de error, no tiene una cadena de texto especifica en la consulta que puede ser analizado en busca de un indicio de ataque de inyección de código SQL.

c) Consulta de Unión

En la Figura 52 se puede apreciar los resultados mostrados de los ataques de unión implementados. El número total de ataques detectados fueron 23 de 26, y se muestra en la Figura 53.

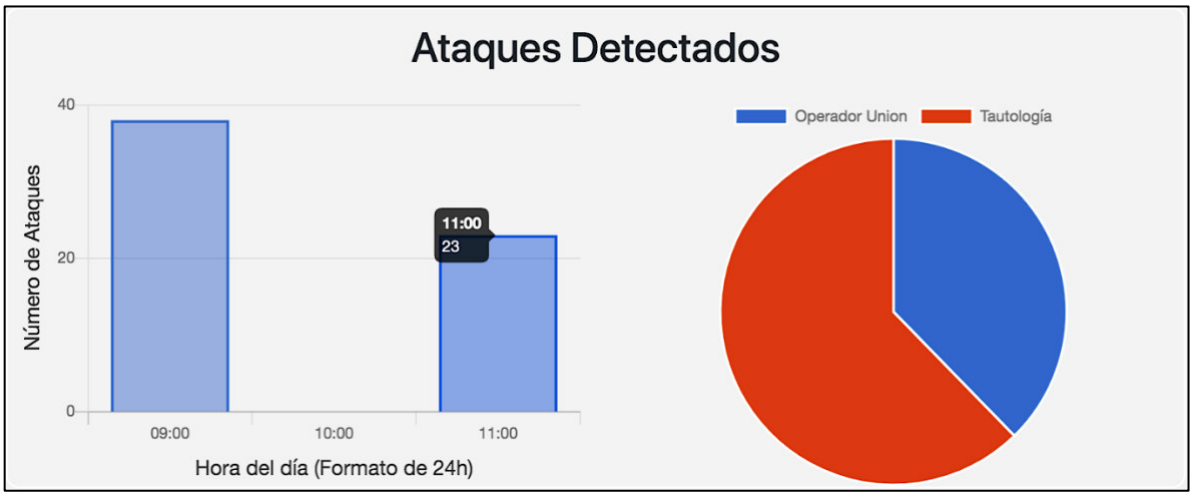


Figura 52. Resultado Inyección de código SQL Operador Unión en pestaña de Resumen Diario

Figura 53. Resultado Inyección de código Operador Unión en pestaña de Ataques

d) Consultas Adicionales

En la Figura 54 se puede apreciar los resultados mostrados de los ataques de consultas adicionales implementados. El número total de ataques detectados fueron 45 de 57, y este resultado se puede apreciar en la Figura 55.

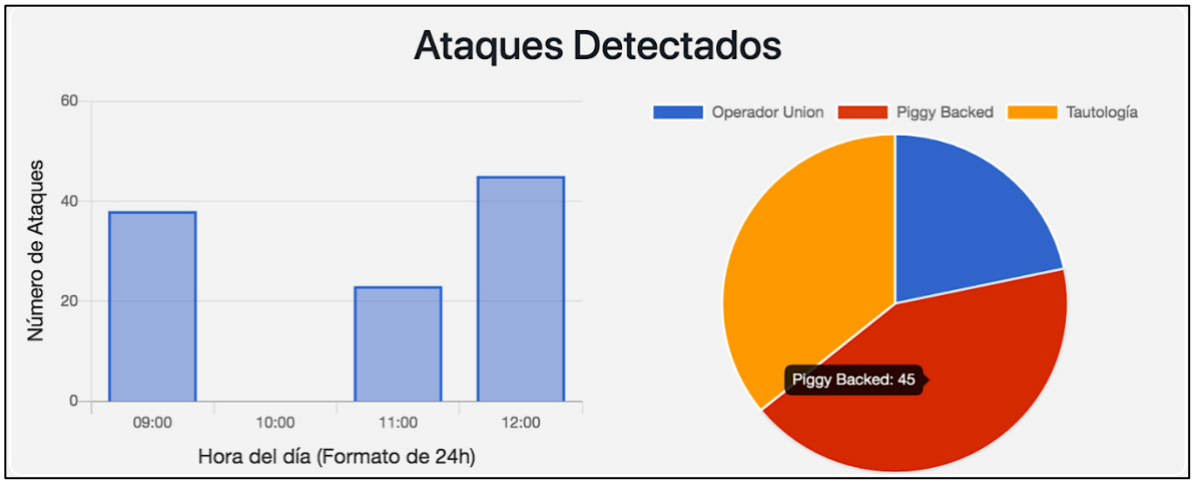


Figura 54. Resultado Inyección de código SQL Consultas Adicionales en la pestaña Resumen Diario

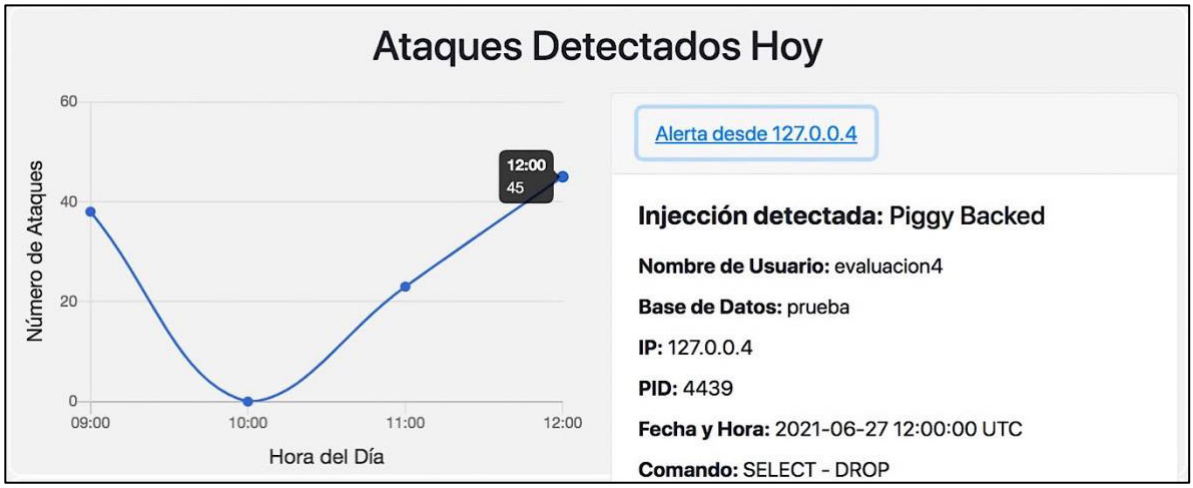


Figura 55. Resultado Inyección de código Consultas Adicionales en pestaña de Ataques

e) Inferencia – Booleano

En la Figura 56 se puede apreciar los resultados mostrados de los ataques de inferencia implementados. El número total de ataques detectados fueron 20 de 26, como se muestra en la Figura 57.

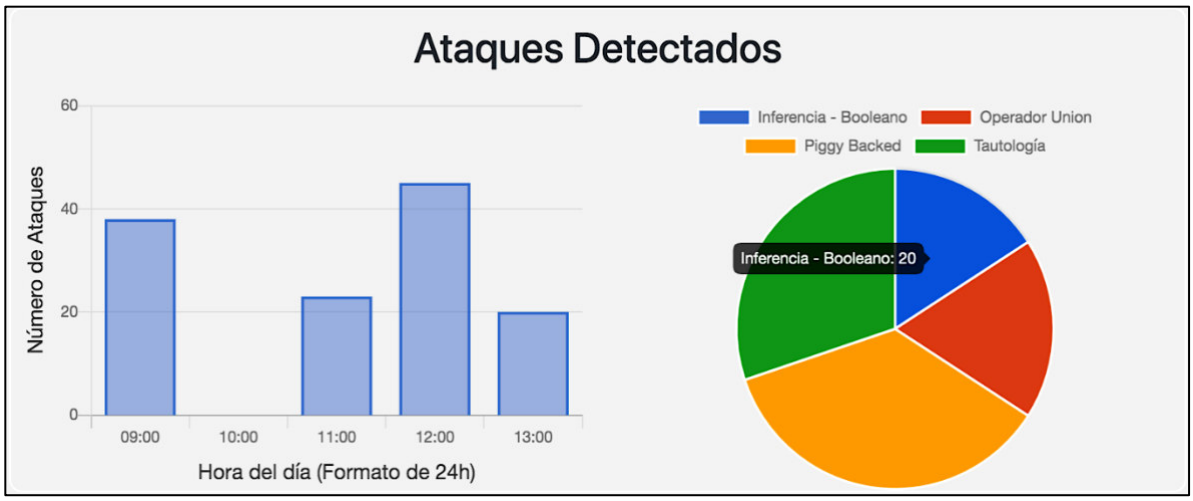


Figura 56. Resultado Inyección Inferencia – Booleano en pestaña de Resumen Diario

Figura 57. Resultado Inyección de código SQL Inferencia – Booleano en pestaña de Ataques

f) Inferencia – Basado en el Tiempo

En la Figura 58 se puede apreciar los resultados mostrados de los ataques de basado en el tiempo implementados. El número total de ataques detectados fueron 53 de 57, y este resultado se refleja en la figura 59.

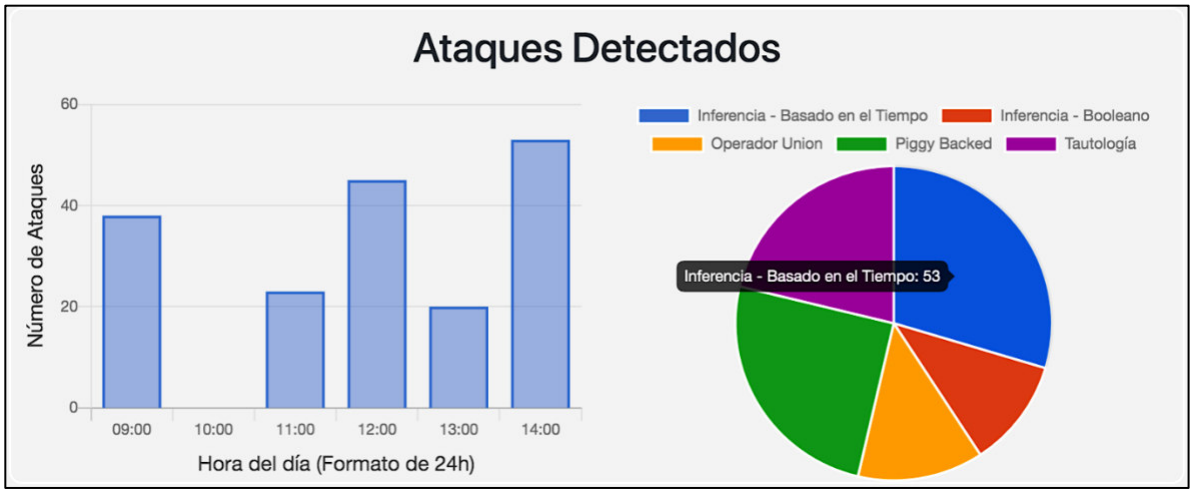


Figura 58. Resultado Inyección de código SQL Inferencia – Basada en Tiempo en pestaña Resumen Diario

Figura 59. Resultado Inyección de código SQL Inferencia – Basada en Tiempo en pestaña de Ataques

g) Procedimientos Almacenados

En la Figura 60 se puede apreciar los resultados mostrados de los ataques de procedimientos almacenados implementados. El número total de ataques detectados fueron 45 de 51, como se puede visualizar en la Figura 61.

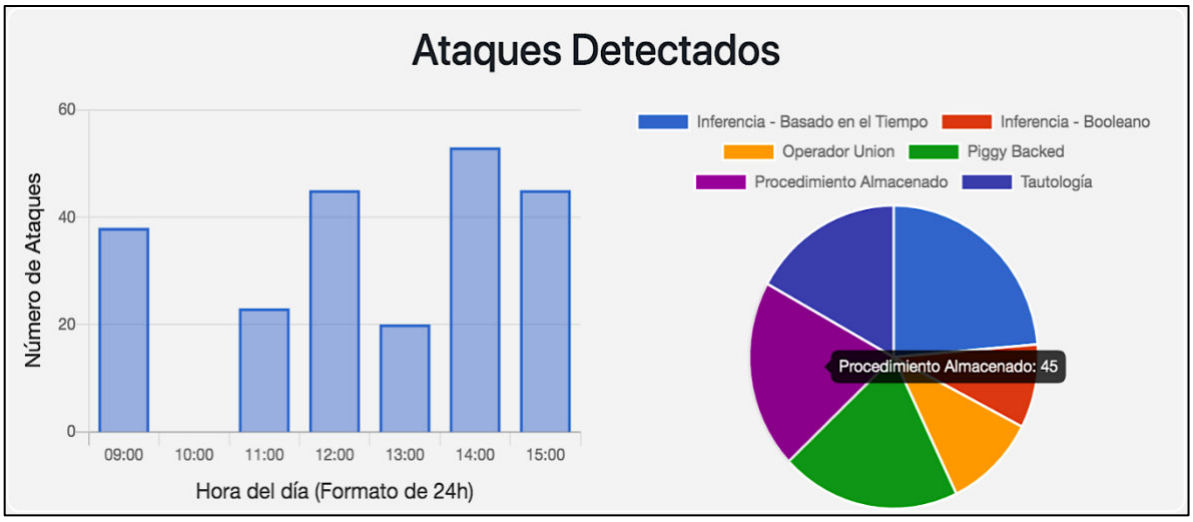


Figura 60. Resultado Inyección de código SQL Procedimiento Almacenado en pestaña de Resumen Diario

Figura 61. Resultado Inyección de código SQL Procedimiento Almacenado en pestaña de Ataques

h) Codificación Alternativa

En la Figura 62 se puede apreciar los resultados mostrados de los ataques de procedimientos almacenados implementados. El número total de ataques detectados fueron 47 de 55, y se puede ver en la Figura 63.

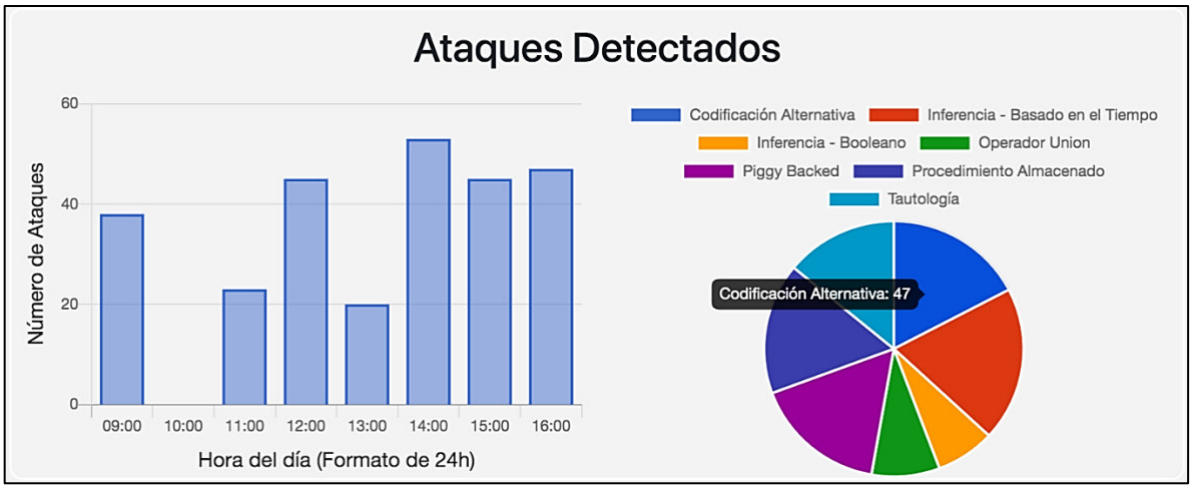


Figura 62. Resultado Inyección de código SQL Codificación Alternativa en pestaña de Resumen Diario

Figura 63. Resultado Inyección de código SQL Codificación Alternativa en pestaña de Ataques

i) Prevención con SQLMAP

La principal vía de los ataques que utilizan la inyección de código SQL son las aplicaciones de páginas web. Por lo que el análisis de causas y prevención de estos ataques fue realizado necesariamente desde la web. La herramienta utilizada, SQL MAP, fue ejecutada bajo un ambiente controlado ya que de ejecutarle en servidores privados estaríamos violando leyes de seguridad de tales proveedores. SQLMAP admite seis técnicas de inyección de código SQL: ciego basado en booleano, ciego basado en tiempo, basado en errores, basado en consultas UNION, consultas apiladas y fuera de banda. Es también capaz de reconocer el formato hash de las contraseñas (Charania & Vyas, 2016).

Los siguientes pasos fueron los empleados para probar la herramienta SQLMAP sobre un servidor con base de datos MySQL. En la Figura 64 se puede observar el sitio web de prueba.

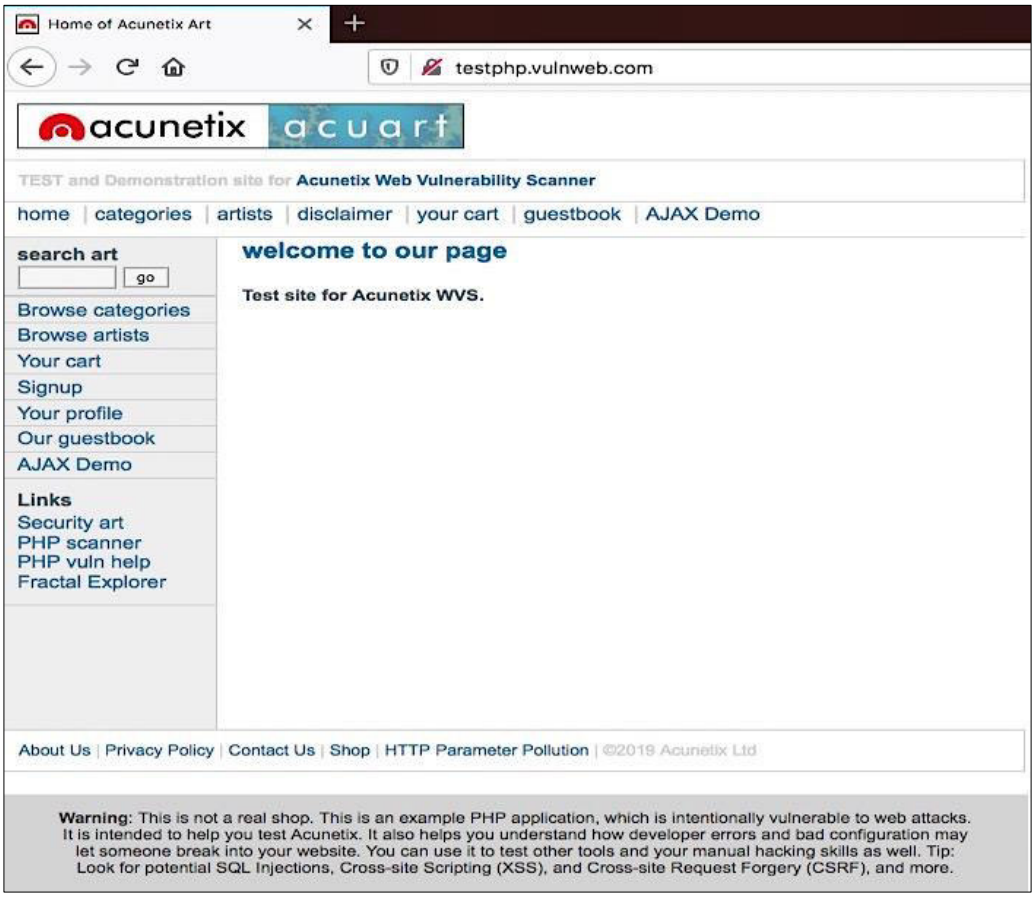


Figura 64. Servidor web de prueba

Usando un sistema operativo basado en Unix, o de una ventana de comando que pueda ejecutar los comandos de Unix, utilizamos la siguiente sentencia en la ventana de comandos:

```
SQLMAP -u http://testphp.vulnweb.com/listproducts.php?cat=1 --dbs
```

En las Figuras 65 y 66 podemos observar los resultados con información importante; el número de base de datos, servidor y puntos de inyección de código SQL posibles con el parámetro ingresado.

Figura 65. Resultado Ejecución SQLMAP 1

Figura 66. Resultado Ejecución SQLMAP 2

Ahora usando el comando -D podemos obtener especificaciones del nombre de la base de datos a la que deseamos acceder, y una vez que tengamos acceso a la base de datos, podemos verificar si hay acceso a las tablas. Para eso se utilizó el comando:

```
SQLMAP -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart --
table
```

Obteniendo como resultado lo que se muestra en la Figura 67 a continuación:

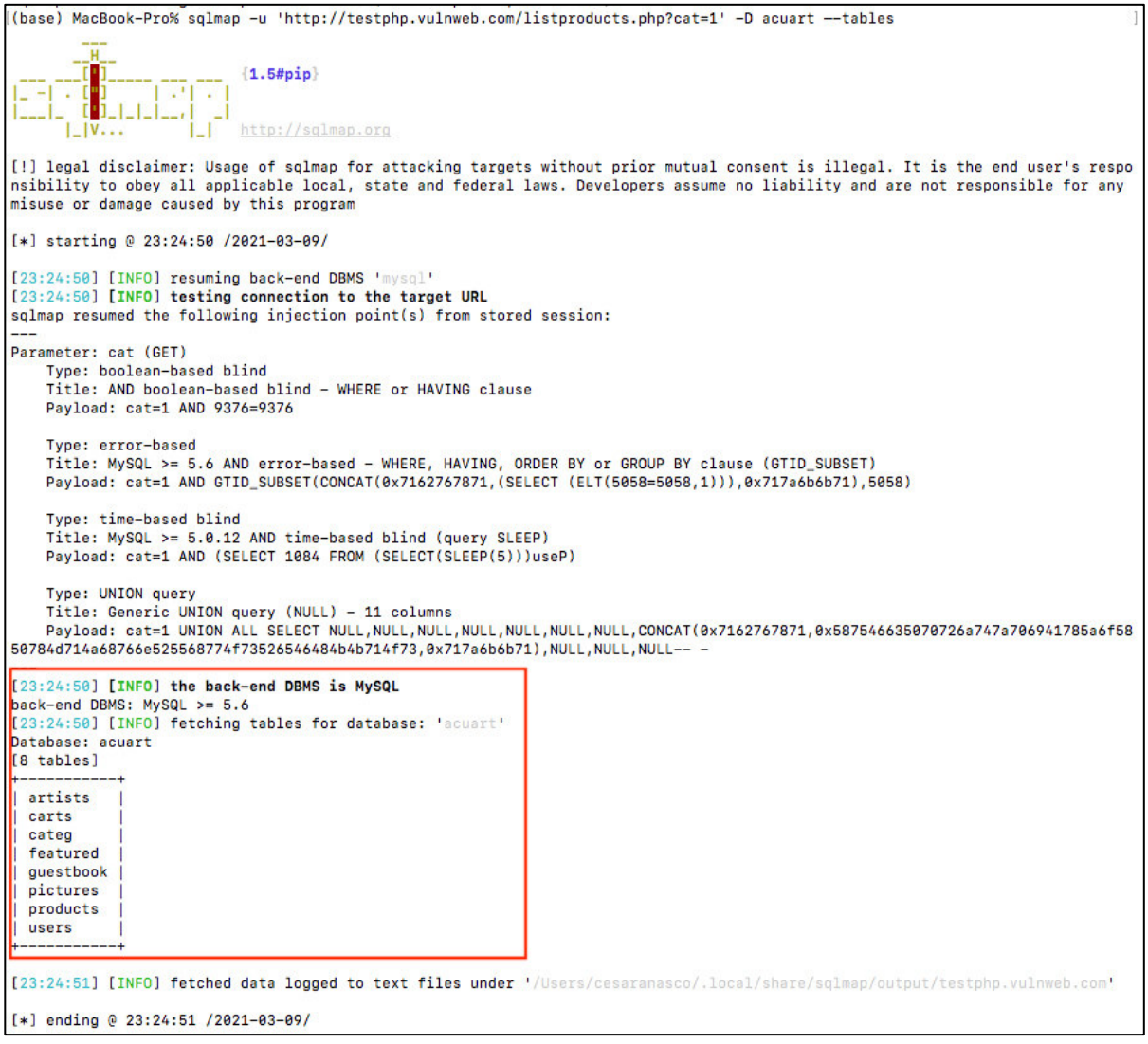


Figura 67. Resultado Ejecución SQLMAP 3

Con las evaluaciones expuestas, es claro que la herramienta SQLMAP tiene una variedad amplia de opciones diferentes que se pueden usar en conjunto para atacar una dirección web en particular y configurar diferentes perfiles de ataques. Por lo que la prevención fue basada en esta herramienta.

3.3.1.4 Resultados

Los ataques detectados en el prototipo están reflejados gráficamente en la primera pestaña. En el primer gráfico de anomalías encontradas en la Figura 68 observamos la presencia de registros de transacciones que el algoritmo encontró como irregulares, pero no fueron catalogados como ataques. Estos usualmente estas conformados por registros de transacciones que reflejan algún comportamiento irregular de un usuario o también registros de transacciones de "consultas ilegales", estas son un tipo de inyección de código SQL para la cual el algoritmo de identificación tiene que mejorase ya que su sentencia de consulta y comportamiento es distinta a los otros tipos de inyecciones de código SQL.

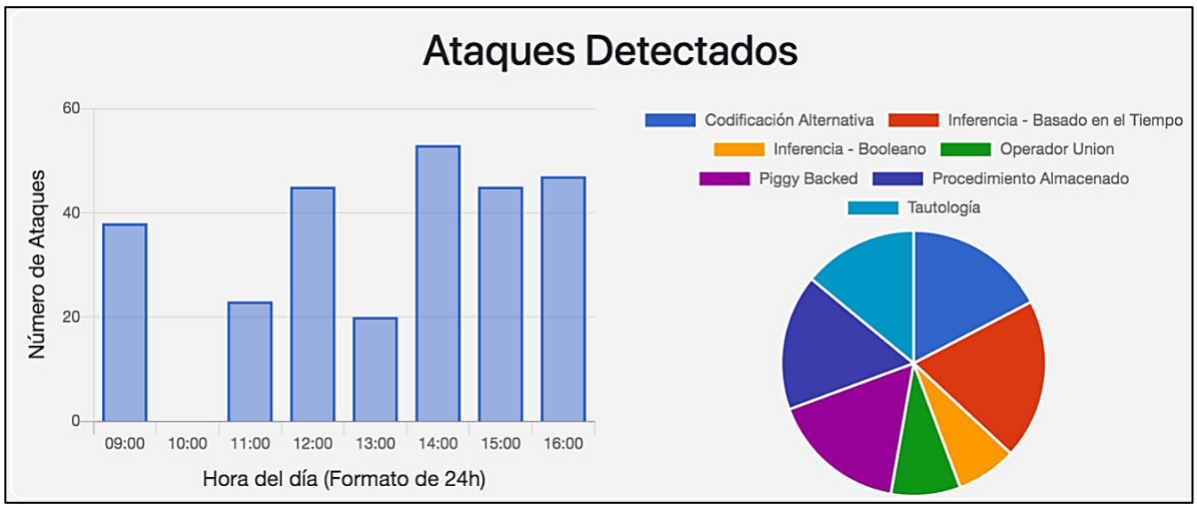
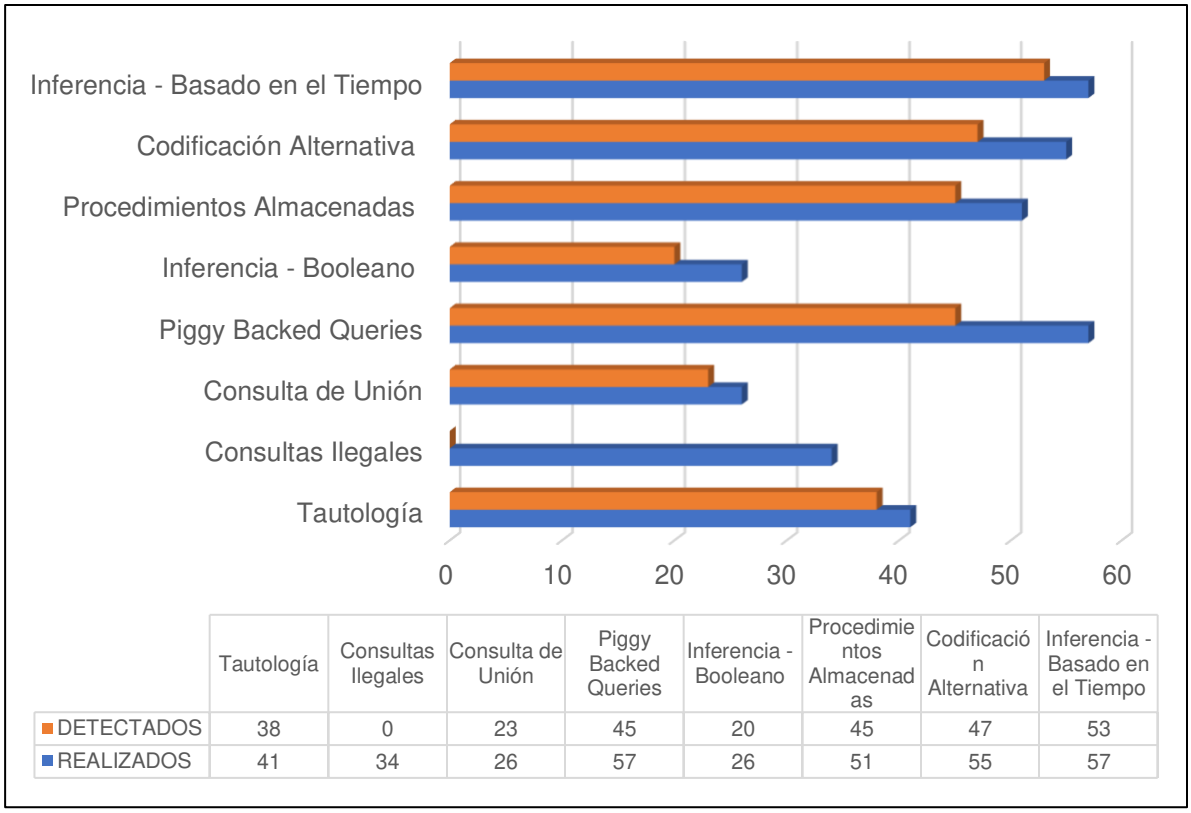


Figura 68. Resultado Global Evaluaciones de Pruebas Unitarias

La primera pestaña Resume Diario proporciona un resumen visual para tener un primer acercamiento de los ataques que están sucediendo en el servidor de base de datos.

En la Figura 69 podemos observar un resumen del número de ataques realizado por cada tipo de inyección de código SQL y el número de los mismos que fue identificado por el prototipo de sistema. El gráfico refleja que en varios tipos de inyección de código SQL el algoritmo no acierta el cien por ciento de los casos, pero su rendimiento es efectivo para el objetivo de este proyecto.



	Tautología	Consultas Ilegales	Consulta de Unión	Piggy Backed Queries	Inferencia - Booleano	Procedimientos Almacenadas	Codificación Alternativa	Inferencia - Basado en el Tiempo
DETECTADOS	38	0	23	45	20	45	47	53
REALIZADOS	41	34	26	57	26	51	55	57

Figura 69. Comparación entre Ataques Realizados y Detectados

En la Figura 70 podemos observar que el porcentaje de identificación de cada tipo de inyección de código SQL sobrepasa el 76% de los casos, exceptuando al tipo de ataque de "consultas ilegales". Estos porcentajes concuerdan con los resultados del cálculo del rendimiento, obtenido de la implementación del algoritmo de clasificación base kNN, expuestos en el literal 3.1.5.2.

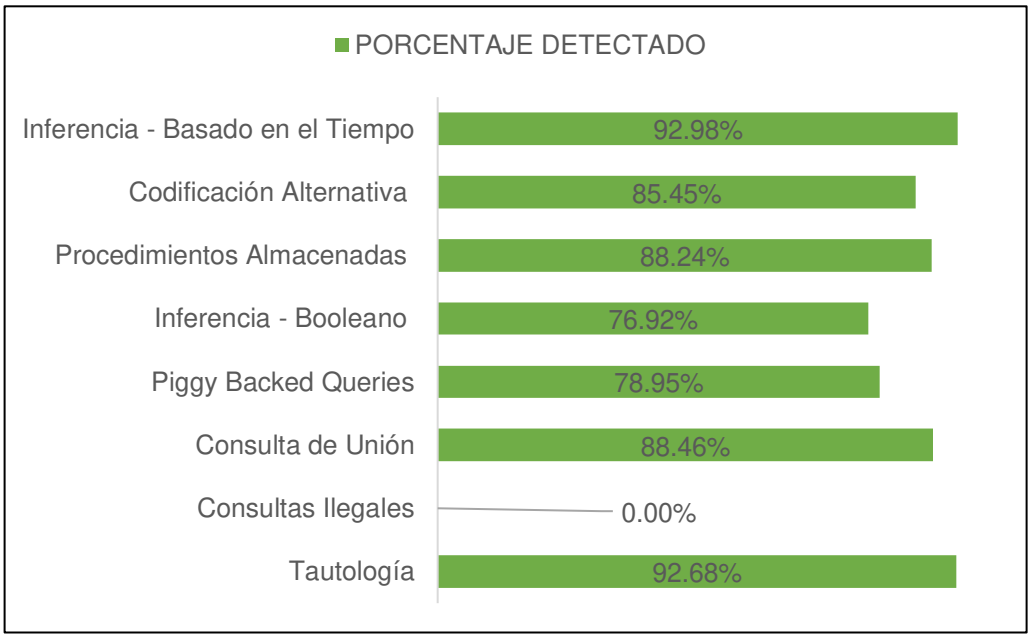


Figura 70. Porcentaje de Ataques Detectados por Tipo

3.3.2. Pruebas de Carga

Las pruebas de carga serán realizadas a la parte más crítica del sistema, la cual es la carga de archivos de registros de transacciones y la ejecución del algoritmo de identificación implementado en la metodología previa CRISP-DM. El objetivo de esta prueba es establecer un límite del tamaño del archivo que es ingresado al sistema y entender el tiempo que se demora en ejecutar el algoritmo con diferentes tamaños de archivos.

3.3.2.1. Planificación de pruebas de carga

Las pruebas de carga del prototipo se realizaron mediante el uso de los archivos en el directorio app>>engine>>CsirtLogs>>, donde se encuentra el motor de análisis de registros de transacciones. Este directorio cuenta con los siguientes cuatro programas escritos en el lenguaje de programación Python:

- **PrimerPaso.py**
  En este paso se realiza la recolección de datos desde el archivo ingresado, lectura de registros de transacciones, extracción de atributos y generación de nuevos archivos. Estos archivos contendrán los datos de los registros de transacciones extraídos en un formato establecido para el siguiente paso.
- **SegundoPaso.py**
  Se realiza una división de conjuntos de registros de transacciones, donde se generan archivos con un máximo de 5000 registros de transacciones cada uno.
- **TercerPaso.py**
  Se realiza la lectura de cada conjunto de registros de transacciones generados en al paso anterior, se realiza la limpieza, transformación, normalización de datos e implementación del algoritmo kNN. El resultado de este paso es la generación de nuevos archivos en donde se encuentren solo los registros de transacciones identificados como anomalías.
- **CuartoPaso.py**
  Se realiza un análisis de texto de la consulta hecha a la base de datos asociado al registro de transacción que fue identificado como anomalía. En caso de registrarse alguna expresión que revele la existencia de un ataque o de su intento, se realiza el registro de anomalía al servidor del sistema.

3.3.2.2. Descripción del método

Consiste en medir el tiempo de ejecución de cada archivo explicado en el numeral 3.3.2.1, mediante la carga de diferentes tamaños de archivos de registros de transacciones. La medición del tiempo de ejecución de los archivos se realizó mediante el uso del comando 'time', el cual muestra los siguientes resultados:

- **USER:** es la cantidad de tiempo de CPU invertido en código de modo de usuario (fuera del kernel) dentro del proceso. Este es solo el tiempo real de la CPU utilizado para ejecutar el proceso.
- **SYS:** es la cantidad de tiempo de CPU invertido en el kernel dentro del proceso. Esto significa ejecutar el tiempo de CPU invertido en llamadas al sistema dentro del kernel, a diferencia del código de la biblioteca, que todavía se ejecuta en el espacio del usuario.
- **CPU:** el porcentaje de la CPU que se asignó al comando.
- **TOTAL:** es la suma de los tres atributos señalados al terminar, el resultado se muestra en segundos.

3.3.2.3. Resultados

Las pruebas fueron realizadas en un ambiente con las siguientes características:

MARCA:	MacBook Pro 2012
SISTEMA OPERATIVO:	macOS HighSierra
PROCESADOR:	2.9 GHz Intel Core i7
MEMORIA RAM:	4 GB 1333 MHz DDR3
TARJETA GRÁFICA:	Intel HD Graphics 4000 1536 MB
DISCO DURO:	500GB

Los resultados se pueden encontrar en Tabla 20.

LONGITUD PROMEDIO DE REGISTROS DE TRANSACCIONES: 934 caracteres

Tabla 20. Resultados de las Pruebas de Carga

ARCHIVOS	TIEMPO (S)																										
TAMAÑO (mb)	1.3			2.6			5.2			10.4			20.7			41.7			83			166			331.6		
NUMERO DE LOGS	5000			10000			20000			40000			80000			160000			320000			640000			1280000		
ACTORES	USER	SYS	CPU	USER	SYS	CPU	USER	SYS	CPU	USER	SYS	CPU	USER	SYS	CPU	USER	SYS	CPU	USER	SYS	CPU	USER	SYS	CPU	USER	SYS	CPU
PrimerPaso.py	0.67	0.17	79%	0.82	0.13	94%	1.26	0.25	82%	2.06	0.32	81%	3.71	0.53	76%	7.43	0.96	89%	15.73	1.92	84%	30.24	4.70	89%	64.06	12.47	61%
	1.062			1.001			1.830			2.915			5.511			9.406			20.968			39.143			124.280		
SegundoPaso.py	1.44	0.43	68%	1.37	0.39	75%	1.42	0.40	78%	1.56	0.47	65%	1.57	0.48	77%	1.81	0.57	73%	2.27	0.71	83%	3.35	1.22	64%	5.42	2.16	77%
	2.743			2.352			2.326			3.097			2.650			3.247			3.581			7.058			9.838		
TercerPaso.py	6.84	0.34	98%	12.12	0.52	92%	22.70	0.66	96%	48.07	0.87	97%	86.57	1.22	97%	178.06	1.81	98%	364.94	3.66	98%	724.67	9.15	97%	1548.94	19.86	97%
	7.283			13.587			24.249			50.516			99.700			92.010			373.910			756.030			1611.580		
CuartoPaso.py	1.76	0.26	27%	3.84	0.38	33%	10.30	0.69	41%	36.02	1.56	50%	127.62	3.47	65%	464.79	7.18	77%	1784.57	16.39	85%	7164.10	51.13	91%	16123.24	78.3	93%
	7.255			12.460			26.433			73.900			199.430			507.220			2102.290			7870.950			17933.230		
TOTAL (s)	18.343			29.400			54.838			130.428			307.291			611.883			2500.749			8673.181			19678.928		

En las figuras 71 y 72, se muestra el comportamiento que tiene el uso de cada archivo cuando se usan con una cantidad distinta de registros de transacciones. Los archivos PrimerPaso.py y SegundoPaso.py tiene un comportamiento casi lineal, por lo que se infiere que el tamaño de los registros de transacciones que se usa sobre estos no afecta su rendimiento. A diferencia de los archivos SegundoPaso.py y TercerPaso.py, el comportamiento de estos tiene mucha más demora cuando el tamaño de los conjuntos de registros de transacciones es más grande.

El comportamiento del archivo TercerPaso.py parece tener más complejidad y tiempo de demora, como se muestra en la figura 71. Sin embargo, en la figura 72 podemos observar que el archivo CuartoPaso.py tiene un crecimiento de demora exponencial cuando el conjunto de registros de transacciones es más grande. Aunque existe aumento del tiempo de demora del tercer archivo, donde se obtuvo una demora de 27 minutos con 1 millón de registros de transacciones, esto es entendible ya que el algoritmo de identificación esta implementado allí. A diferencia del cuarto archivo, cuyo crecimiento descontrolado se debe a la función de registro que es enviada al servidor cada vez que una anomalía es encontrada.

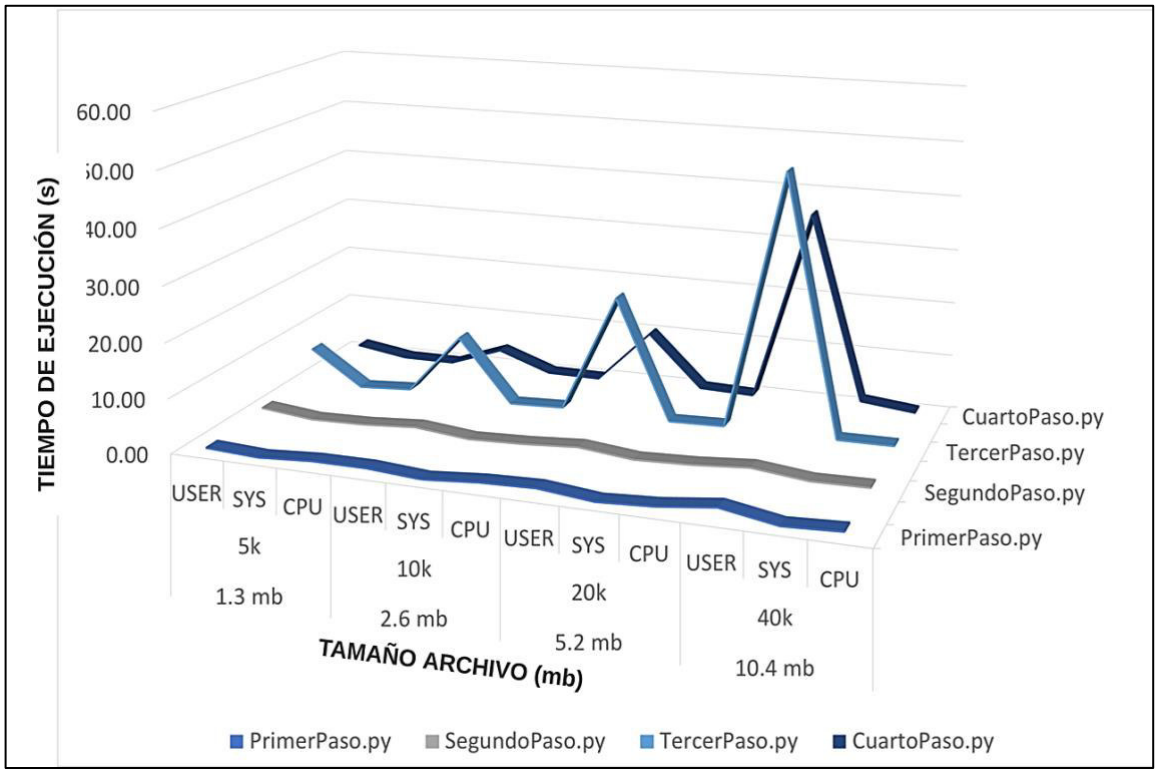


Figura 71. Resultado de Pruebas de Carga de los primeros Cuatro Archivos de Registros de Transacciones

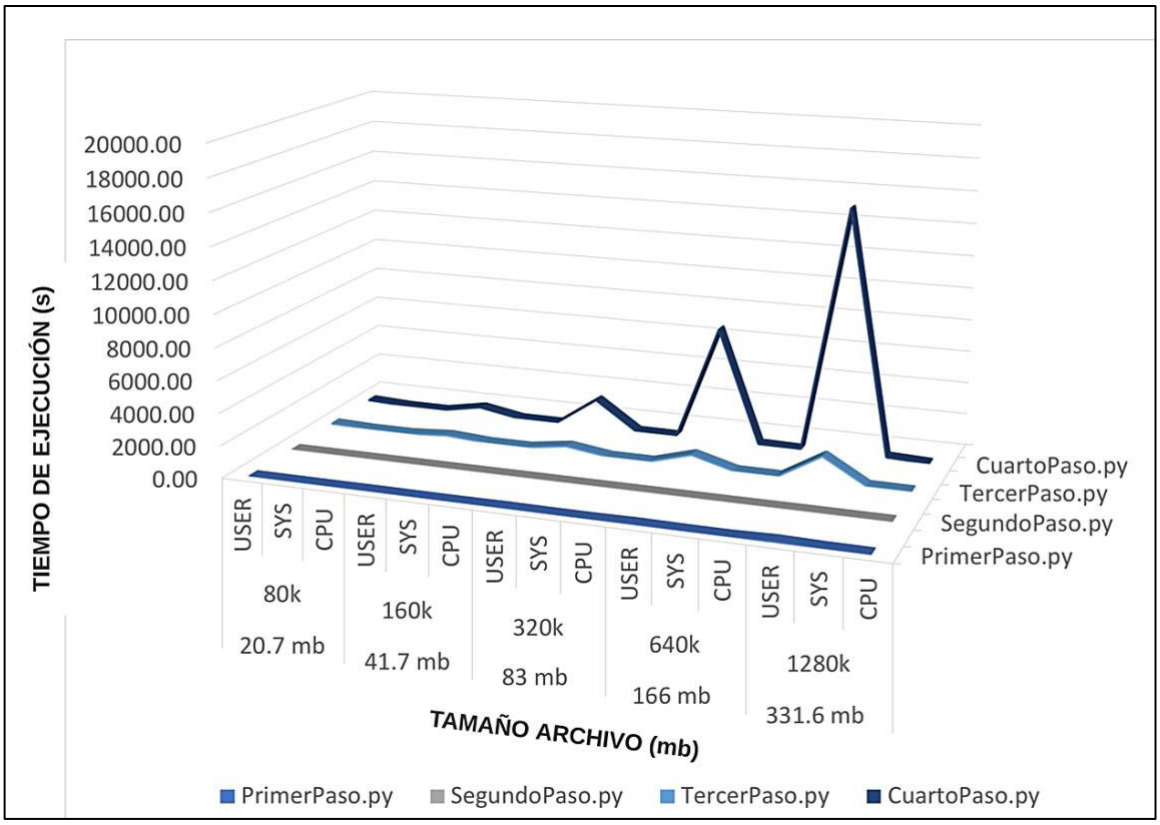


Figura 72. Resultados de Pruebas de Carga de los segundos Cinco Archivos de Registros de Transacciones

En las figuras 73 y 74 podemos observar que el archivo que hace más uso de la unidad central de procesamiento (CPU) es el archivo TercerPaso.py, lo cual se debe a la separación de conjuntos de registros de transacciones y la identificación de anomalías que se realiza en este paso. Los demás archivos tienen un uso del CPU uniforme y normal para las tareas que realizan.

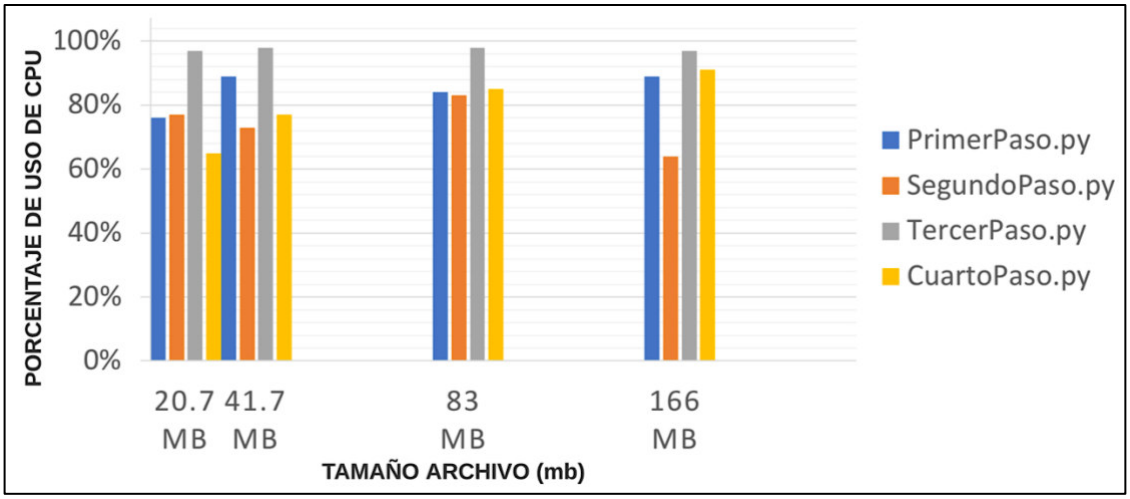


Figura 73. Resultados de uso de CPU de los primeros Cuatro Archivos de Registros de Transacciones

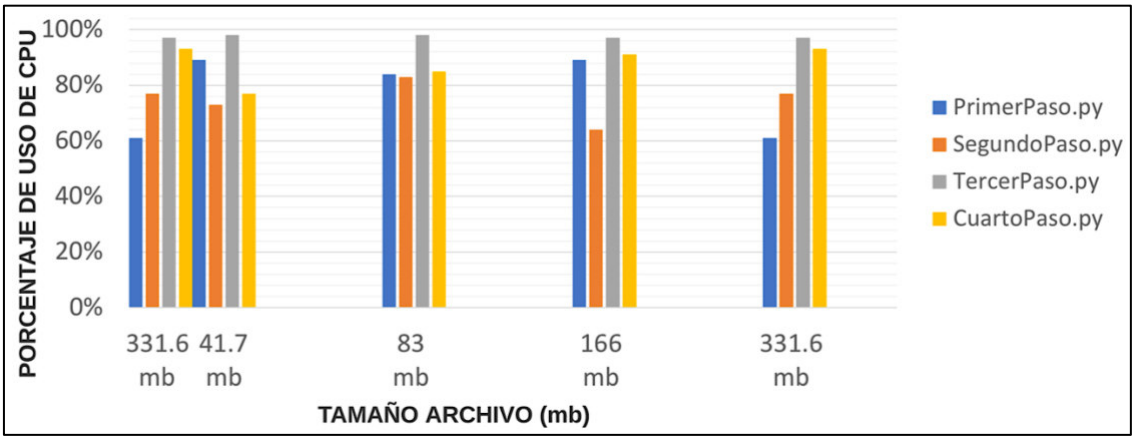


Figura 74. Resultados de uso de CPU de los segundos Cinco Archivos de Registros de Transacciones

3.3.3. Pruebas de Usabilidad

El concepto usabilidad de un sistema software tiene dos componentes principales: uno hace referencia al aspecto funcional del sistema y otro a cómo los usuarios pueden usar dicha funcionalidad (Brooke, 2018). En esta última hemos basada las pruebas finales realizadas al prototipo web de sistema. Los factores principales establecidos fueron; la efectividad, eficiencia, satisfacción del sistema frente a los usuarios.

3.3.3.1. Planificación de pruebas de usabilidad

Las pruebas de usabilidad del prototipo se realizaron mediante la creación de un formulario en “Google Forms”. El formulario contiene un listado de siete preguntas que se describen en el numeral 3.3.3.2 y fueron respondidas por 10 estudiantes egresados de la Facultad de Sistemas quienes navegaron y utilizaron el sistema de visualización guiándose en el manual de usuario.

3.3.3.2. Descripción del método

Consiste en un cuestionario de 8 ítems, con cinco opciones de respuesta del 1 al 5 que va desde “Total desacuerdo” y “Total acuerdo”. Dos de estos ítems son de identificación (correo institucional) y de retroalimentación (comentario).

Los ítems con calificación tienen las siguientes preguntas:

- ¿Utilizaría usted este prototipo de sistema para la detección de ataques a PostgreSQL?
- ¿Considera usted que el prototipo de sistema es intuitivo?
- ¿Cada módulo realiza correctamente su función acorde al nombre que lleva?
- ¿Considera usted que los módulos del prototipo tienen relación?
- ¿Cree usted que las gráficas desplegadas en el prototipo son claras?

- ¿Cree usted que los manuales explican claramente los requisitos necesarios para la instalación y uso del prototipo?
- ¿Cree usted que el manual de instalación explica claramente la ruta de ubicación de registros de transacciones y la forma en la que deben estar estructurados para que funcionen correctamente con el algoritmo implementado en el prototipo web?

3.3.3.3. Realización de la encuesta

A todos los encuestados se les envió los manuales de instalación y de usuario, así como también el link del prototipo, para que hagan uso del sistema. Tras el registro y uso del aplicativo, se les pidió responder la encuesta creada.

3.3.3.4. Resultados individuales

Se obtuvieron los siguientes resultados:

La Figura 75 muestra como resultado que si se utilizaría el prototipo de sistema para detectar ataques a PostgreSQL porque cumple con las expectativas y el motivo para el cual fue creado.

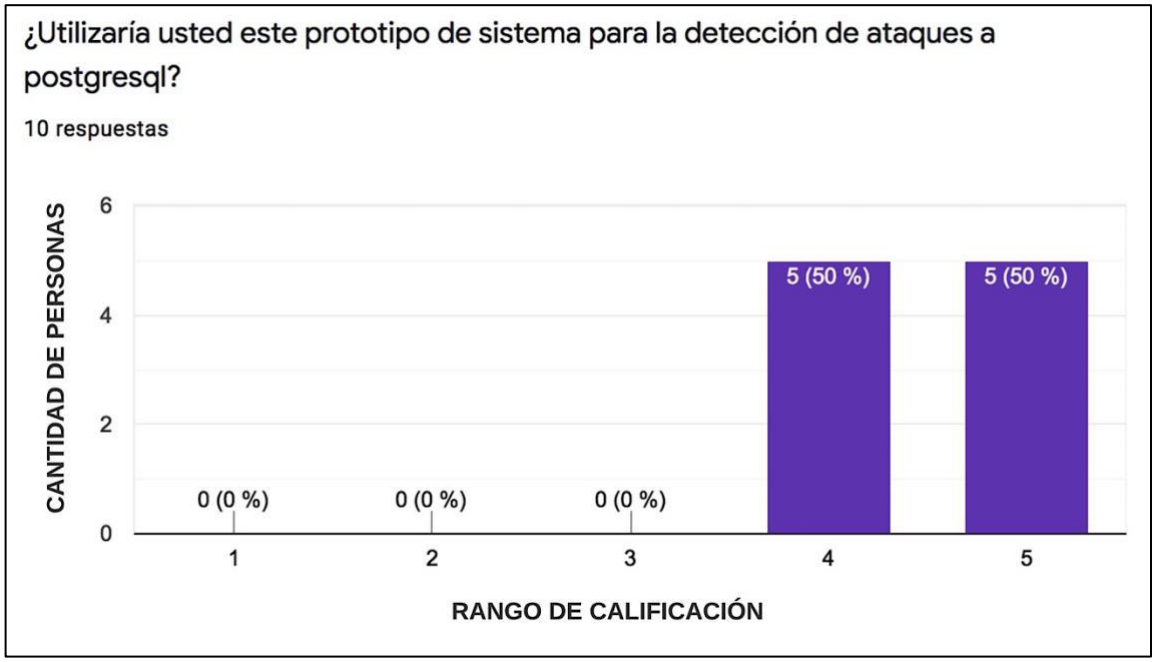


Figura 75. Resultado de la pregunta 1

La Figura 76 muestra que el aplicativo es intuitivo al contar con pocos módulos y poder navegar por el sistema sin complicación, además que cumple con el objetivo para quienes lo utilizan.

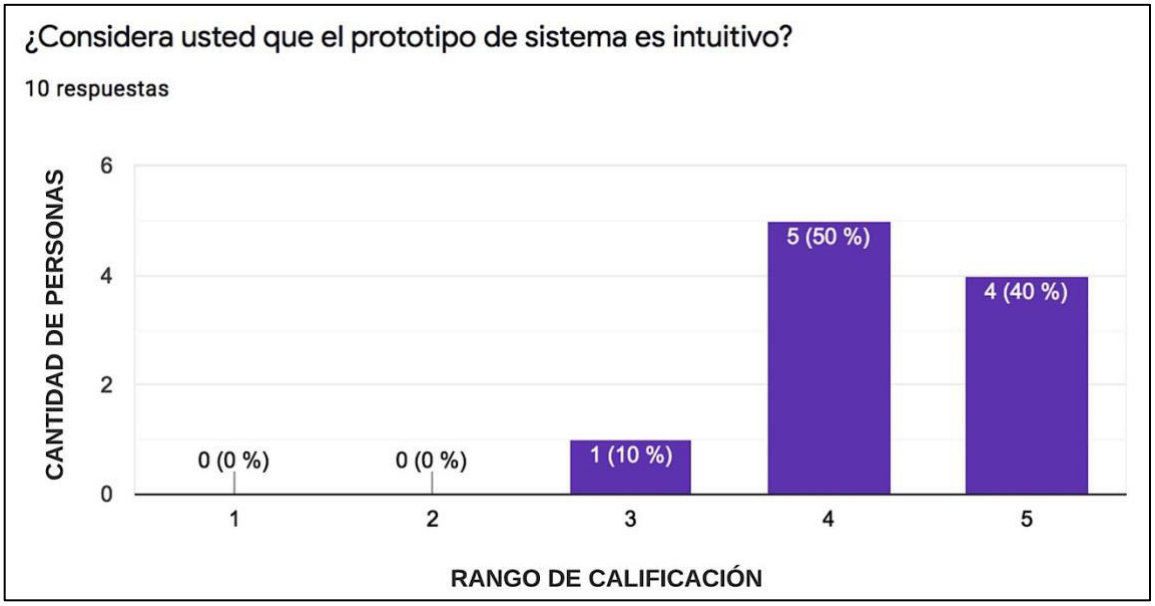


Figura 76. Resultado de la pregunta 2

La Figura 77 refleja como resultado de la pregunta que cada módulo del sistema funciona correctamente en base al nombre y el despliegue del prototipo es bueno. No se encontró mayor dificultad al navegar por cada módulo.

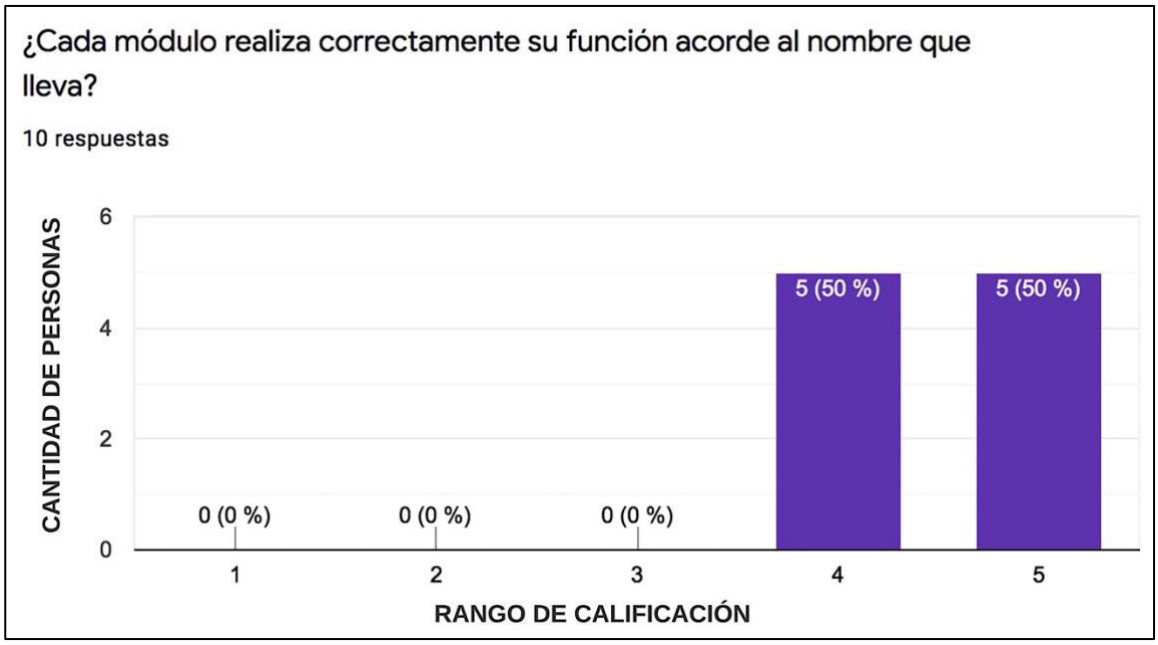


Figura 77. Resultado de la pregunta 3

En la Figura 78 se puede entender que la mayoría de usuarios encontró que los módulos del prototipo son intuitivos y que tiene relación entre sí.

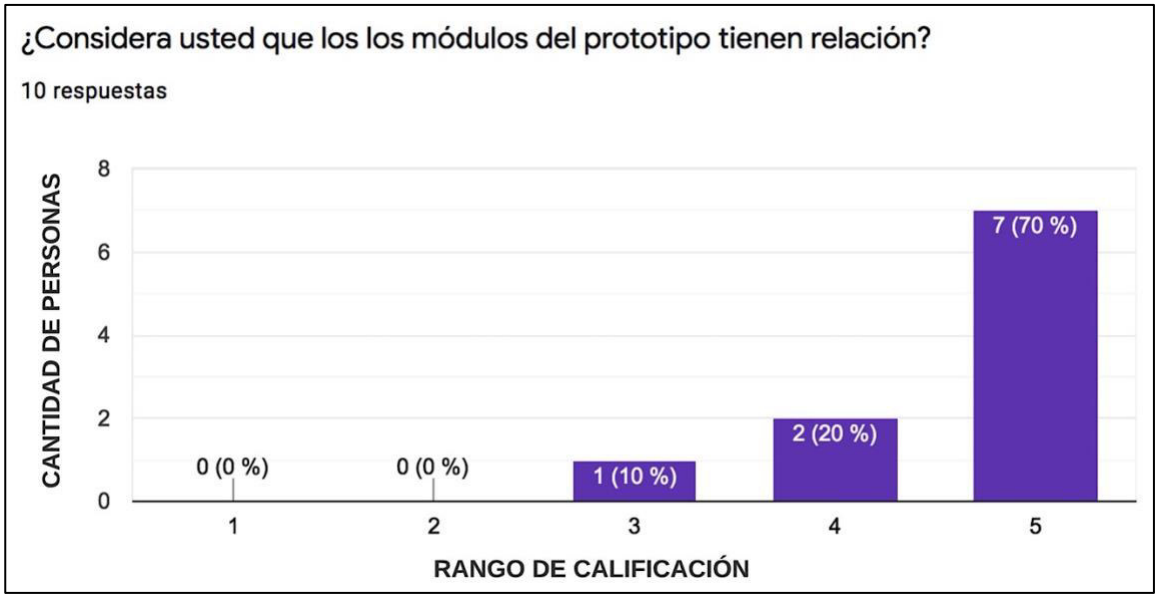


Figura 78. Resultado de la pregunta 4

En la Figura 79 se mira obtuvo un gráfico con más disparidad dentro de los resultados de la encuesta, los usuarios consideran que el despliegue de las gráficas puede mejorar.

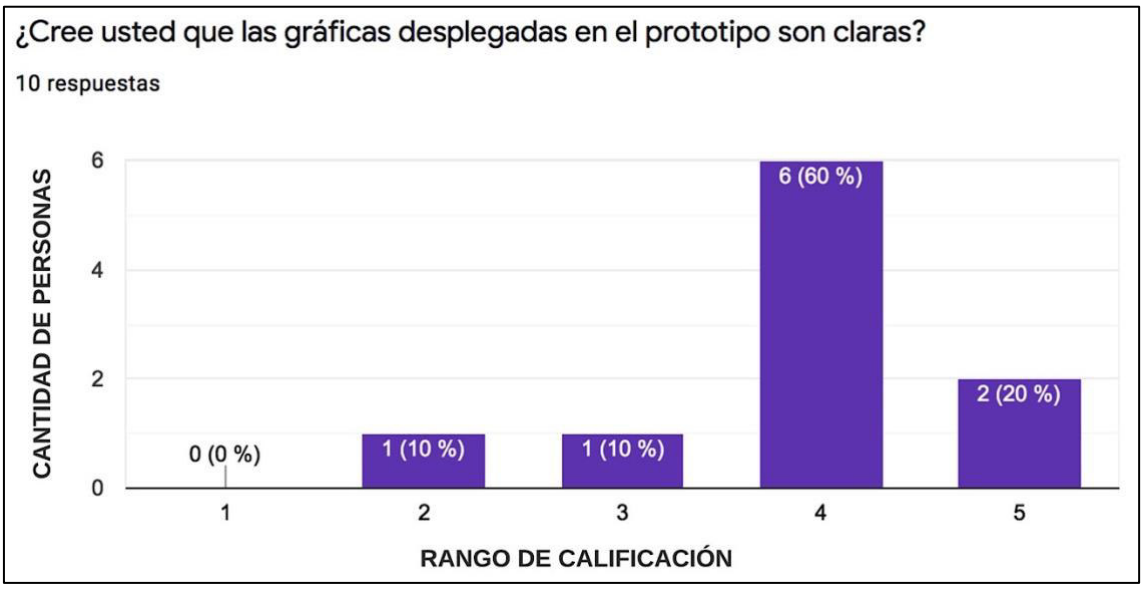


Figura 79. Resultado de la pregunta 5

La Figura 80 muestra que los manuales revisados y utilizados por los usuarios son fáciles e intuitivos de usar con el aplicativo web.

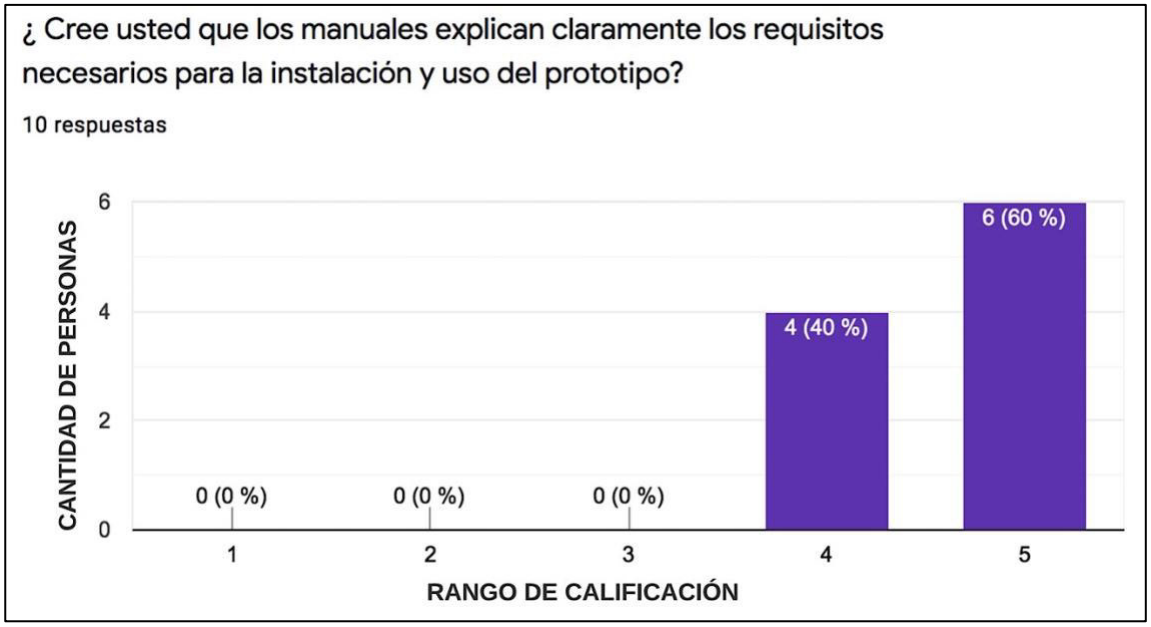


Figura 80. Resultado de la pregunta 6

En la Figura 81, se indica que la mayoría de usuarios considera que la localización de registros de transacciones para el uso e instalación del prototipo son claramente explicadas en los manuales, lo cual es de suma importancia ya que son parte vital del funcionamiento del prototipo.

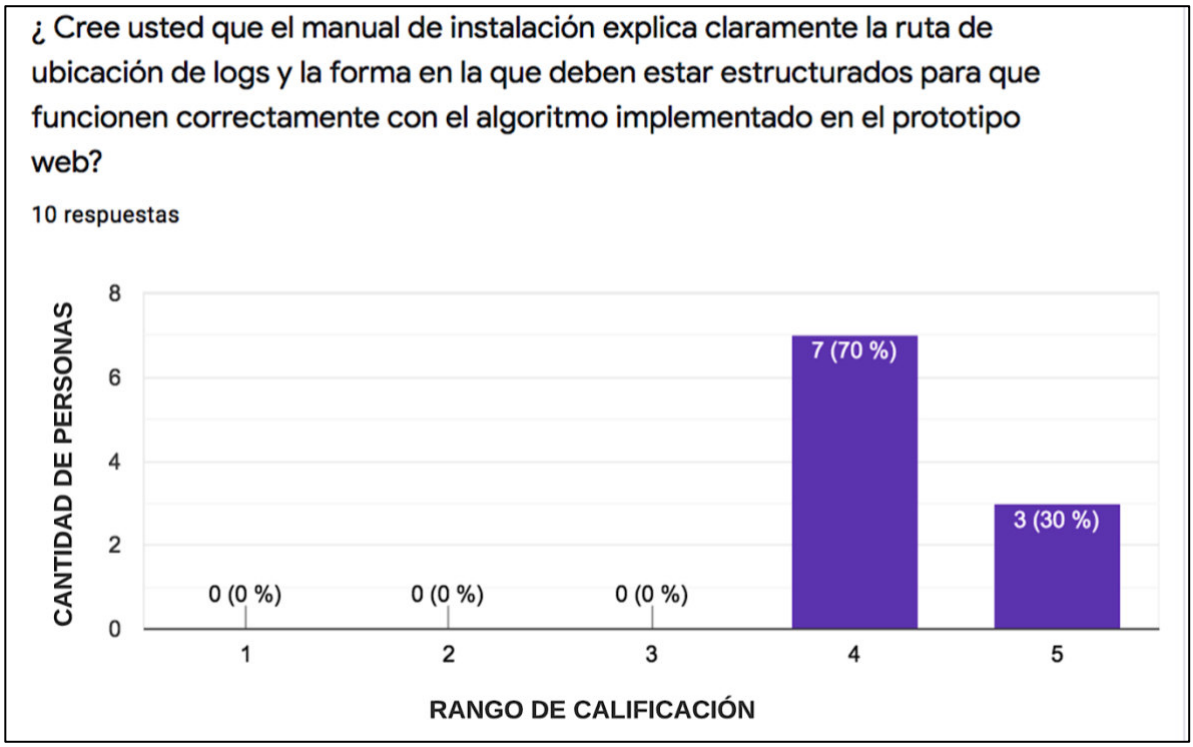


Figura 81. Resultado de la pregunta 7

3.3.3.5. Resultados globales

Los resultados de la Tabla 21 indican que la aplicación tiene una buena acogida por los usuarios y se ve reflejado en cada respuesta y en el puntaje obtenido que se acercan a los 10 puntos. El resultado más bajo se refleja en la pregunta de la Figura 76, en la cual se hace referencia a las gráficas desplegadas en el prototipo web, por lo cual se considerará un cambio en la mismas. También existe una retroalimentación muy importante en el último ítem de la encuesta, el cual no es porcentual y se puede encontrar en el Anexo 5.

Tabla *21*. Resultados Numéricos de la Evaluación

PREGUNTA	**PUNTAJE (/50)**	**PUNTAJE (/10)**
Figura 75	45	9.00
Figura 76	43	8.60
Figura 77	45	9.00
Figura 78	46	9.20
Figura 79	39	7.80
Figura 80	46	9.20
Figura 81	43	8.60
TOTAL	**307/350**	**87.71%**

CAPÍTULO IV: CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS

4.1. CONCLUSIONES

Se ha desarrollado con éxito una herramienta para identificar anomalías, así como de identificación de ataques para los registros de transacciones generados por los servidores de base de datos implementados por el CSIRT-EPN. Las pruebas de funcionalidad en la etapa de evaluación reflejaron que el algoritmo implementado en el sistema es práctico, y tiene cabida para ser implementado no solo en bases PostgreSQL, sino también en otras que pueden generar datos similares utilizados en este proyecto.

El desarrollo del modelo de minería de datos bajo la metodología CRISP-DM, facilitó el proceso pues la secuencia que sigue sus fases no es rígida y permitió avanzar o retroceder entre sus distintas etapas. Esto se ve reflejado principalmente al efectuar la etapa de modelamiento e identificar que existían datos innecesarios del conjunto de registros de transacciones, por lo que fue necesario regresar a la etapa de preparación de datos, realizar una nueva limpieza y borrado de datos ineficaces y luego continuar con el desarrollo de los siguientes pasos de la metodología.

Se logró integrar una gran cantidad de registros de transacciones utilizando los parámetros más relevantes generados en PostgreSQL. Siendo este un factor importante para realizar varias pruebas con las técnicas de minería de datos implementadas y de gran ayuda al escoger el algoritmo kNN como la técnica principal para el proyecto.

Utilizar la metodología en cascada permitió agilizar el desarrollo del sistema de visualización gracias a que se tenía una cantidad pequeña de requerimientos claros y detallados. La manera secuencial de desarrolló implicó pasar a la siguiente etapa únicamente cuando la anterior haya finalizado siguiendo de manera fácil el proyecto.

El prototipo de sistema desarrollado permitió incorporar los resultados del modelo y visualizar la estadística diaria del comportamiento del modelo de análisis. Al momento de que la herramienta identifique alguna anomalía, el sistema emite alertas para notificar un comportamiento extraño o una posible inyección. Sin embargo, las predicciones realizadas emitidas en las alertas del sistema no siempre pueden ser correctas.

De acuerdo con las pruebas realizadas, tanto al sistema como al modelo, el sistema cumple su propósito. Sin embargo, pueden presentar demoras en el análisis de registros de transacciones. El algoritmo de clasificación basado en proximidad kNN resultó ser el más adecuado en la detección de ataques de inyección de código SQL, y su eficacia se basa en el límite de 5000 registros de transacciones por análisis. Es decir que cuantos más registros

de registros de transacciones sean ingresados la complejidad computacional será mayor, por lo que el uso de recursos computacionales crece en cada análisis.

4.2. RECOMENDACIONES Y TRABAJOS FUTUROS

Se recomienda usar el algoritmo kNN para conjuntos de datos moderados y pequeños, para tener un buen rendimiento del modelo.

A partir del proyecto realizado, se propone analizar la integración del algoritmo implementado para la identificación de anomalías con una red neuronal, ya que se ha demostrado en varios estudios actuales que se puede obtener un identificador de intrusos más robusto y acertado, mediante el uso de una red de aprendizaje constante.

Se recomienda dar cabida a más algoritmos de aprendizaje no supervisado para el modelamiento de usuarios, para así poder identificar no solo ataques externos a los servidores, sino también los ataques que pueden ocurrir desde el interior del C-SIRT.
Se recomienda indagar en profundidad sobre métodos de identificación de otros tipos de ataque como: exceso de privilegios, denegación de servicios (DoS), datos sensibles sin manejo, etc. Pues existe una gama amplia de ataques internos y externos, a los que pueden ser vulnerables los servidores de bases de datos, diferentes a los identificados en este proyecto.

REFERENCIAS BIBLIOGRÁFICAS

Al-Sayid, N. A., & Aldlaeen, D. (2013). Database security threats: A survey study. *2013 5th International Conference on Computer Science and Information Technology, CSIT 2013 - Proceedings*. https://doi.org/10.1109/CSIT.2013.6588759

Ali, A. B. M., Shakhatreh, A. Y. I., Abdullah, M. S., & Alostad, J. (2011). SQL-injection vulnerability scanning tool for automatic creation of SQL-injection attacks. *Procedia Computer Science*. https://doi.org/10.1016/j.procs.2010.12.076

Anitha, V., Supha Lakshmi, A., Revathi, M., & Selvi, K. (2015). Detecting various SQL Injection vulnerabilities using String Matching and LCS method. *6th International Conference on Advanced Computing, ICoAC 2014*. https://doi.org/10.1109/ICoAC.2014.7229717

Bace, R., & Mell, P. (2001). NIST special publication on intrusion detection systems. *Nist Special Publication*, 1–51.

Bächle, M., & Kirchberg, P. (2007). Ruby on rails. *IEEE Software*. https://doi.org/10.1109/MS.2007.176

BID, & OEA. (2020). CIBERSEGURIDAD, Riesgos, Avances y el camino a seguir en America Latina y El Caribe. *Bid- Oea*, *1*, 204.

Bootstrap. (2019). *Bootstrap · La biblioteca de HTML, CSS y JS más popular del mundo.*

Bossi, L., Bertino, E., & Hussain, S. R. (2017). A system for profiling and monitoring database access patterns by application programs for anomaly detection. *IEEE Transactions on Software Engineering*. https://doi.org/10.1109/TSE.2016.2598336

Brahma, A., & Panigrahi, S. (2020). Role of Soft Outlier Analysis in Database Intrusion Detection. *Advances in Intelligent Systems and Computing*. https://doi.org/10.1007/978-981-15-1081-6_41

Brooke, J. (2018). System usability scale (SUS). *Iron and Steel Technology*.

Chandrababu, A., & Muddangula, A. (2019). *Adoption of Hybrid Methodology in projects* [Uppsala Universitet]. https://www.diva-

portal.org/smash/get/diva2:1393923/FULLTEXT01.pdf

Charania, S., & Vyas, V. (2016). SQL Injection Attack :Detection and Prevention. *International Research Journal of Engineering and Technology*. https://docplayer.net/49705074-Sql-injection-attack-detection-and-prevention.html

Chetan, R., & Ashoka, D. V. (2012). Data mining based network intrusion detection system: A database centric approach. *2012 International Conference on Computer Communication and Informatics, ICCCI 2012*. https://doi.org/10.1109/ICCCI.2012.6158816

Gong, X., Zhou, Y., Bi, Y., He, M., Sheng, S., Qiu, H., He, R., & Lu, J. (2019). Estimating Web Attack Detection via Model Uncertainty from Inaccurate Annotation. *Proceedings - 6th IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2019 and 5th IEEE International Conference on Edge Computing and Scalable Cloud, EdgeCom 2019*. https://doi.org/10.1109/CSCloud/EdgeCom.2019.00019

Grupo Global de PostgreSQL. (2020). *PostgreSQL: la base de datos de código abierto más avanzada del mundo*.

Gupta, S. (2020). Using SQLMAP tool. In *SQL Injection Attacks*. https://doi.org/10.1007/978-1-4842-6505-5_3

John, A., Moreno, E., & Caldas, D. (2019). *en Servicios Web con SQL Injection Modelo base de conocimiento para auditorías de seguridad en Servicios Web con SQL Injection*.

Kemp, C., & Gyger, B. (2013). Professional Heroku Programming. *Professional Heroku Programming*.

Malik, M., & Patel, T. (2016). Database Security - Attacks and Control Methods. *International Journal of Information Sciences and Techniques*. https://doi.org/10.5121/ijist.2016.6218

Mattsson, U. T. (2011). How to Prevent Internal and External Attacks on Data - Securing the Enterprise Data Flow Against Advanced Attacks. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.1144290

Montañana Pérez, R., Monserrat Coll, F., & Mira, E. (2003). Implantación de un sistema de detección de intrusos en un entorno universitario. *RedIRIS*, *64*, 21–32.

Morales, C. X. S., & Castaño, S. A. L. (2018). *Modelo de Detección de Intrusos Usando Técnicas de Aprendizaje de Máquina*. Tecnológico de Antioquia.

Nadali, A., Kakhky, E. N., & Nosratabadi, H. E. (2011). Evaluating the success level of data mining projects based on CRISP-DM methodology by a Fuzzy expert system. *ICECT 2011 - 2011 3rd International Conference on Electronics Computer Technology*, *6*, 161–165. https://doi.org/10.1109/ICECTECH.2011.5942073

Nandasana, D., & Barot, V. (2017). A framework for database intrusion detection system. *Proceedings - International Conference on Global Trends in Signal Processing, Information Computing and Communication, ICGTSPICC 2016*. https://doi.org/10.1109/ICGTSPICC.2016.7955272

Ntagwabira, L., & Kang, S. L. (2010). Use of query tokenization to detect and prevent SQL injection attacks. *Proceedings - 2010 3rd IEEE International Conference on Computer Science and Information Technology, ICCSIT 2010*. https://doi.org/10.1109/ICCSIT.2010.5565202

Ojagbule, O., Wimmer, H., & Haddad, R. J. (2018). Vulnerability Analysis of Content Management Systems to SQL Injection Using SQLMAP. *Conference Proceedings - IEEE SOUTHEASTCON*. https://doi.org/10.1109/SECON.2018.8479130

Omotunde, H., & Ibrahim, R. (2016). Mitigating SQL injection attacks via hybrid threat modelling. *2015 IEEE 2nd International Conference on InformationScience and Security, ICISS 2015*. https://doi.org/10.1109/ICISSEC.2015.7371019

Párrizas, A. A. (2005). *Propuesta de una Arquitectura de Sistemas de Detección de Intrusos con Correlación*.

Python Software Foundation. (2020). *El tutorial de Python - documentación de Python 3.9.1*.

Racciatti, H. (2002). *Técnicas de SQL Injection: Un Repaso (Versión 1.5)*.

RailsGuides. (n.d.). *Webpacker - Guías de Ruby on Rails*.

Raj, S. N., & Sherly, E. (2018). An SQL injection defensive mechanism using reverse insertion technique. *Communications in Computer and Information Science*. https://doi.org/10.1007/978-981-10-8660-1_25

Rendon, M., & Acosta, J. (2006). *Estudio sobre el estado de las soluciones ICT y de los casos prácticos de aplicación de minería de datos en almenos 5 casos representativos.*

Sadeghian, A., Zamani, M., & Abdullah, S. M. (2013). A taxonomy of SQL injection attacks. *Proceedings - 2013 International Conference on Informatics and Creative Multimedia, ICICM 2013*, 269–273. https://doi.org/10.1109/ICICM.2013.53

Scrum, A., & Meade, B. M. (2010). Waterfall - Software Development Model. In *Library*.

Solarte Martinez, G. R., Ocampo, C. A., & Castro Bermúdez, Y. V. (2017). Sistema de detección de intrusos en redes corporativas. *Scientia et Technica*, *22*(1), 60. https://doi.org/10.22517/23447214.9105

Telefónica Company. (2015). *Bases De Datos Y Sus Vulnerabilidades Más Comunes*. https://www.acens.com/wp-content/images/2015/03/vulnerabilidades-bbdd-wp-acens.pdf

Varshney, K., & Ujjwal, R. L. (2019). LsSQLIDP : Literature survey on SQL injection detection and prevention techniques. *Journal of Statistics and Management Systems*. https://doi.org/10.1080/09720510.2019.1580904

Vartouni, A. M., Kashi, S. S., & Teshnehlab, M. (2018). An anomaly detection method to detect web attacks using Stacked Auto-Encoder. *2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems, CFIS 2018*. https://doi.org/10.1109/CFIS.2018.8336654

Verizon Business. (2019a). 2019 Data Breach Investigations Report. *Verizon Business Journal*.

Verizon Business. (2019b). Verizon: 2019 Data Breach Investigations Report. *Computer Fraud & Security*, *2019*(6), 4. https://doi.org/10.1016/s1361-

3723(19)30060-0

VMware Inc. (2020). *Descargar VMware Workstation Pro | LATAM*.

Wang, B., Ying, S., & Yang, Z. (2020). A Log-Based Anomaly Detection Method with Efficient Neighbor Searching and Automatic K Neighbor Selection. *Scientific Programming*. https://doi.org/10.1155/2020/4365356

Zhao, Y., Nasrullah, Z., & Li, Z. (2019). PyOD: A python toolbox for scalable outlier detection. In *arXiv*.

ANEXOS

Anexo 1. Servidor de Prueba del Sistema

Enlace al servidor web: https://tesisw.herokuapp.com/

Anexo 2. Repositorio Web

Enlace al repositorio web: https://github.com/wayiok/TesisEPN

Anexo 3. Manual de Usuario

Anexo 4. Manual de Instalación

Anexo 5. Resultados de Pruebas de Usabilidad