Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas

DISEÑO DE LA INVESTIGACIÓN DE SEGURIDAD EN LA INFORMACIÓN AL UTILIZAR APIS IMPLEMENTADAS MEDIANTE EL PROTOCOLO ODATA PARA SERVICIOS DE INTEGRACIÓN EN UNA EMPRESA COMERCIALIZADORA DE CALZADO

Julio Antonio Gordiano Carranza

Asesorado por M.A. Ing. Mario Enrique López Herrarte

Guatemala, noviembre 2024

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DISEÑO DE LA INVESTIGACIÓN DE SEGURIDAD EN LA INFORMACIÓN AL UTILIZAR APIS IMPLEMENTADAS MEDIANTE EL PROTOCOLO ODATA PARA SERVICIOS DE INTEGRACIÓN EN UNA EMPRESA COMERCIALIZADORA DE CALZADO

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA
DE LA FACULTAD DE INGENIERÍA
POR

JULIO ANTONIO GORDIANO CARRANZA
ASESORADO POR M.A. ING. MARIO ENRIQUE LÓPEZ HERRARTE

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, NOVIEMBRE 2024

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. José Francisco Gómez Rivera (a. i.)
VOCAL II	Ing. Mario Renato Escobedo Martinez
VOCAL III	Ing. José Milton De León Bran
VOCAL IV	Ing. Kevin Vladimir Cruz Lorente
VOCAL V	Ing. Fernando José Paz González
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. César Rolando Batz Saquimux
EXAMINADOR	Ing. Oscar Alejandro Paz Campos
EXAMINADOR	Ing. Pedro Pablo Hernández Ramírez
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO DE LA INVESTIGACIÓN DE SEGURIDAD EN LA INFORMACIÓN AL UTILIZAR APIS IMPLEMENTADAS MEDIANTE EL PROTOCOLO ODATA PARA SERVICIOS DE INTEGRACIÓN EN UNA EMPRESA COMERCIALIZADORA DE CALZADO

Tema que me fuera asignado por la Dirección de la Escuela de Estudios de Posgrado, con fecha 28 de septiembre 2024.

Julio Antonio Gordiano Carranza

EEPFI-PP-5176-2024
Guatemala, 28 de septiembre de 2024

Director
Carlos Gustavo Alonzo
Escuela De Ingenieria En Sistemas
Presente.

Estimado Carlos Gustavo Alonzo

Reciba un cordial saludo de la Escuela de Estudios de Postgrado de la Facultad de Ingeniería.

El propósito de la presente es para informarle que se ha revisado y aprobado el Diseño de Investigación titulado: **DISEÑO DE LA INVESTIGACIÓN DE SEGURIDAD EN LA INFORMACIÓN AL UTILIZAR APIS IMPLEMENTADAS MEDIANTE EL PROTOCOLO ODATA PARA SERVICIOS DE INTEGRACIÓN EN UNA EMPRESA COMERCIALIZADORA DE CALZADO**, el cual se enmarca en la línea de investigación: **Área de Innovación - Dispositivos y sistemas para incrementar la seguridad al utilizar tecnología de la información y comunicaciones**, presentado por el estudiante **Julio Antonio Gordiano Carranza** carné número **200312674**, quien optó por la modalidad del "PROCESO DE GRADUACIÓN DE LOS ESTUDIANTES DE LA FACULTAD DE INGENIERÍA OPCIÓN ESTUDIOS DE POSTGRADO". Previo a culminar sus estudios en la Maestría en Artes en Tecnologias De La Inf. Y La Comunicacion.

Y habiendo cumplido y aprobado con los requisitos establecidos en el normativo de este Proceso de Graduación en el Punto 6.2, aprobado por la Junta Directiva de la Facultad de Ingeniería en el Punto Décimo, Inciso 10.2 del Acta 28-2011 de fecha 19 de septiembre de 2011, firmo y sello la presente para el trámite correspondiente de graduación de Pregrado.

Atentamente,

"Id y Enseñad a Todos"

Mtro. Mario Enrique López Herrarte
Asesor(a)

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
ESCUELA DE POSTGRADO
FACULTAD DE INGENIERIA

Mtro. Marlon Antonio Perez Turk
Coordinador(a) de Maestría

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECTORA
ESCUELA DE ESTUDIOS DE POSTGRADO
FACULTAD DE INGENIERÍA

Mtra. Aurelia Anabela Cordova Estrada
Directora
Escuela de Estudios de Postgrado
Facultad de Ingeniería

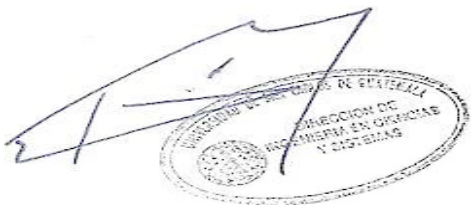
UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÍA

EEP-EICS-5038-2024

El Director de la Escuela De Ingenieria En Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del Asesor, el visto bueno del Coordinador y Director de la Escuela de Estudios de Postgrado, del Diseño de Investigación en la modalidad Estudios de Pregrado y Postgrado titulado: **DISEÑO DE LA INVESTIGACIÓN DE SEGURIDAD EN LA INFORMACIÓN AL UTILIZAR APIS IMPLEMENTADAS MEDIANTE EL PROTOCOLO ODATA PARA SERVICIOS DE INTEGRACIÓN EN UNA EMPRESA COMERCIALIZADORA DE CALZADO**, presentado por el estudiante universitario **Julio Antonio Gordiano Carranza**, procedo con el Aval del mismo, ya que cumple con los requisitos normados por la Facultad de Ingeniería en esta modalidad.

ID Y ENSEÑAD A TODOS



Mtro. Carlos Gustavo Alonzo
Director
Escuela De Ingenieria En Sistemas

Guatemala, septiembre de 2024

Decanato
Facultad e Ingeniería
24189101- 24189102

LNG.DECANATO.OIE.758.2024

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al Trabajo de Graduación titulado: **DISEÑO DE LA INVESTIGACIÓN DE SEGURIDAD EN LA INFORMACIÓN AL UTILIZAR APIS IMPLEMENTADAS MEDIANTE EL PROTOCOLO ODATA PARA SERVICIOS DE INTEGRACIÓN EN UNA EMPRESA COMERCIALIZADORA DE CALZADO**, presentado por: **Julio Antonio Gordiano Carranza** después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

Firmado electrónicamente por: José Francisco Gómez Rivera
Motivo: Informe final PREGRADOPOSTGRADO
Fecha: 21/11/2024 17:49:26
Lugar: Facultad de Ingeniería, USAC.

Ing. José Francisco Gómez Rivera
Decano a.i.

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO a.i.
Facultad de Ingeniería

Guatemala, noviembre de 2024

ACTO QUE DEDICO A:

Dios	Por ser mi guía y fortaleza, y por darme sabiduría, fe y esperanza cada día.
Mi esposa	Por su amor incondicional y apoyo constante, que hacen cada día mejor.
Mis hijos	Por ser mi mayor motivación y alegría, inspirándome a seguir adelante.
Mi mamá	Por su amor inagotable, enseñanzas y sacrificio, siendo mi refugio y guía. (q. e. p. d.)
Mi papá	Por ser mi ejemplo de vida y valores, y por su apoyo fundamental. (q. e. p. d.)
Mi hermano	Por ser una fuente constante de ánimo, fortaleza y apoyo toda la vida.
Mi familia	Por su amor, apoyo y motivación, siendo una fuente de fuerza y estimación.
Mis amigos	Por su amistad y compañía, motivándome a seguir superándome en la vida.

AGRADECIMIENTOS A:

Dios	Por su guía, protección y fortaleza, que me han ayudado a superar desafíos y alcanzar mis metas.
Mi esposa	Ingrid Reinoza por su amor, paciencia y apoyo inquebrantable, siendo un pilar fundamental en mi vida.
Mis hijos	Sofía y Sebastián Gordiano Reinoza por ser una fuente constante de alegría e inspiración, impulsándome a ser mejor cada día.
Mis padres	Ramiro Gordiano y Ma. Del Carmen Carranza por su amor, enseñanzas y apoyo incondicional, y por ser un ejemplo de esfuerzo y dedicación.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

INTRODUCCIÓN

En la era digital, la seguridad de la información se ha convertido en un factor crítico para las empresas, especialmente aquellas que dependen del uso intensivo de servicios *web* para sus operaciones diarias. El creciente uso de protocolos como OData, diseñados para facilitar la manipulación de datos a través de interfaces RESTful, ha conseguido desarrollar nuevas oportunidades para optimizar procesos, pero también nuevos desafíos en cuanto a la protección de la información.

En este contexto, el riesgo asociado a las vulnerabilidades inherentes de OData adquiere particular relevancia para la empresa comercializadora de calzado que es objeto de este estudio. El presente trabajo aborda las principales problemáticas de seguridad que enfrenta la empresa al implementar servicios basados en OData, y busca proponer soluciones orientadas a mejorar la protección de la información sensible. La capacidad de asegurar la integridad, confidencialidad y disponibilidad de los datos resulta esencial para asegurar la continuidad del negocio y mantener su competitividad en el mercado.

En el capítulo 1 se abordan los antecedentes del estudio, presentando investigaciones y trabajos previos que presenta desafíos significativos en la implementación de seguridad de las APIs REST y servicios OData. A continuación, en el capítulo 2, se justifica la importancia del proyecto, detallando las razones que sustentan su ejecución y destacando la línea de investigación relacionada con el área de estudio.

El capítulo 3 detalla los alcances investigativos y técnicos, así como los resultados esperados del proyecto, brindando una visión clara de los logros que se esperan obtener.

En el capítulo 4, se presenta el marco teórico, proporcionando una base sobre la seguridad del protocolo OData. En primer lugar, se abordan los conceptos fundamentales sobre OData, su arquitectura y su relevancia en la interoperabilidad de servicios y sistemas empresariales. Se exploran los principios del intercambio de datos mediante RESTful APIs y cómo OData se ha convertido en un estándar clave para integrar aplicaciones. Este marco proporciona una comprensión integral de las bases tecnológicas y de seguridad que son críticas para garantizar la integridad y la confidencialidad.

En el capítulo 5 se describen los resultados del desarrollo del prototipo de servicios OData, desde el diseño del flujo de los servicios RESTful y la implementación de autenticación multifactor, hasta la configuración de un API *gateway* con Azure y la integración con herramientas de seguridad como Azure Active Directory. Se abordan la monitorización, pruebas automatizadas y una evaluación de seguridad con simulaciones de ataques. El capítulo finaliza con una síntesis de los resultados y una comparativa de rendimiento del prototipo en entornos controlados.

El capítulo 6 discute los resultados obtenidos en la arquitectura basada en servicios OData, con énfasis en el rendimiento de la autenticación multifactor y el control de acceso. Se evalúa el impacto de las herramientas de Azure en la optimización del sistema, se identifican mejores prácticas para la implementación segura de servicios RESTful con OData, y se exploran posibles mejoras, como la escalabilidad y la incorporación de nuevas funcionalidades basadas en inteligencia artificial.

ANTECEDENTES

En el contexto de la seguridad de las APIs REST utilizadas en servicios en la nube y *web*, según Atlidakis y colaboradores (2020), se destaca la falta de herramientas maduras para probar automáticamente la seguridad, así como la escasez de orientación sobre su uso seguro. Para abordar este vacío, se presentan reglas de seguridad que describen propiedades deseables de las APIs REST y los servicios asociados. Para probar y detectar violaciones de estas reglas, se propone ejecutar *fuzzing* o enviar datos aleatorios, inválidos y no esperados mediante los formularios de entrada de una aplicación buscando alguna vulnerabilidad en una API REST con verificadores de propiedades activas.

Sin embargo, se reconoce la necesidad de expandir estas pruebas a más servicios y verificar más propiedades para detectar una variedad más amplia de vulnerabilidades. La contribución de este trabajo radica en proporcionar un marco inicial para evaluar la seguridad de las APIs REST y promover buenas prácticas en su diseño y uso. Este enfoque también aumenta la confianza en la infraestructura digital en general al abordar los riesgos asociados con la programación de forma remota y automatizada a través de APIs REST.

Los servicios basados en protocolo OData ayudan a satisfacer la necesidad de integrar y compartir información de manera eficiente. Facilita la publicación y consumo de servicios en línea orientados a datos mediante APIs RESTful y un lenguaje de consulta basado en URL. OData es popular por su madurez y simplicidad de uso. Ed-Douibi y colaboradores (2018), proponen un enfoque basado en modelos para semiautomatizar la representación de datos, transformación de solicitudes en sentencias SQL y deserialización de mensajes,

utilizando diagramas de clases UML (*Unified Modeling Language*) para generar los artefactos para un servicio OData sobre una base de datos relacional.

Es importante mencionar que, como trabajo futuro, se sugiere extender el enfoque generativo para agregar soporte predefinido para características básicas en cualquier infraestructura *web*, como seguridad (es decir, autenticación y encriptación). La autenticación multifactor, la prevención de inyección de consultas y el cifrado de datos son algunas de las estrategias que pueden fortalecer la seguridad de los servicios basados en OData.

Los servicios *web* RESTful proporcionan datos a través de APIs usando HTTP, lo que plantea desafíos para las pruebas debido a las secuencias de solicitudes y respuestas HTTP. La mayoría de los enfoques existentes realizan pruebas de caja negra como menciona Laranjeiro y colaboradores (2021). En este caso se realizan pruebas basadas únicamente en la información mínima expresada en las descripciones de sus interfaces. Se evaluaron un conjunto heterogéneo de 52 servicios REST que comprenden 1,351 operaciones y se ajustan a distintas categorías (por ejemplo, públicas, privadas, internas). Se revelaron diferentes tipos de problemas, incluidos la fiabilidad y también algunas vulnerabilidades de seguridad.

Sin embargo, Arcuri (2017), propone un enfoque de pruebas de caja blanca totalmente automatizado, generando casos de prueba mediante un algoritmo evolutivo. Plantea un enfoque para maximizar la cobertura de código y encontrar fallos utilizando los estados de retorno HTTP. Permite una evaluación exhaustiva de la lógica interna y la estructura del código, identificando errores que podrían no ser detectados por métodos de caja negra. Al tener acceso al código fuente, las pruebas pueden ser diseñadas para cubrir todas las rutas posibles, mejorando la eficacia de las pruebas.

Ambos enfoques, son complementarios y pueden ser utilizados en conjunto para una evaluación más exhaustiva de los servicios REST. Mientras que el enfoque de caja blanca permite una prueba detallada y orientada a la lógica del código, el enfoque de caja negra ofrece una evaluación práctica y centrada en la robustez y seguridad del servicio en escenarios reales.

Los servicios *web* RESTful se utilizan hoy en día de manera extensa para facilitar la comunicación en múltiples escenarios. Según Agnelo (2020), este tipo de sistema está especialmente expuesto a problemas de robustez (por ejemplo, falta o debilidad en la verificación de entradas) dado su diseño con restricciones relajadas y la falta de estándares. Los resultados de su trabajo muestran la capacidad de herramientas basadas en pruebas de caja negra para evaluar diferentes tipos de servicios y revelar problemas de robustez, así como de seguridad en casi la mitad de los servicios probados.

En conclusión, la seguridad de las APIs REST presenta desafíos significativos debido a la falta de herramientas maduras y directrices claras para su uso seguro. Atlidakis et al. (2020) destacan la importancia de las pruebas automáticas mediante *fuzzing* para detectar vulnerabilidades, mientras que Ed-Douibi et al. (2018) promueve la integración de características de seguridad en servicios OData. Los enfoques de pruebas de caja negra y caja blanca, propuestos por Laranjeiro et al. (2021), Agnelo (2020) y Arcuri (2017), se complementan y son necesarios para una evaluación efectiva de estos servicios. La adopción de buenas prácticas y herramientas efectivas no solo mejora la seguridad de los servicios *web*, sino que también aumenta la confianza en la infraestructura digital en general.

PLANTEAMIENTO DEL PROBLEMA

- Contexto general

La seguridad de la información es un pilar fundamental en el mundo digital actual. Con la creciente adopción de servicios *web* y la necesidad de compartir datos de manera eficiente, protocolos como OData (*Open Data Protocol*) han ganado popularidad por su capacidad para facilitar el acceso y la manipulación de datos a través de interfaces RESTful. Sin embargo, la implementación de servicios mediante OData presenta deficiencias importantes de seguridad que comprometen la integridad, confidencialidad y disponibilidad de la información, poniendo en riesgo tanto la seguridad de los datos como la eficiencia operativa. Este problema se vuelve crítico en el contexto de una empresa comercializadora de calzado, donde la protección de la información sobre clientes, proveedores y transacciones es vital para mantener la confianza y la competitividad en el mercado.

- Descripción del problema

El protocolo OData, aunque potente y flexible, tiene vulnerabilidades inherentes que pueden ser explotadas si no se implementan adecuadamente medidas de seguridad. Una de las principales deficiencias es la falta de mecanismos robustos de autenticación y autorización. En muchos casos, las implementaciones de OData no utilizan autenticación multifactor, lo que deja las puertas abiertas para que actores malintencionados accedan a información sensible con relativa facilidad. La autenticación básica no es suficiente para

proteger datos críticos en un entorno donde las amenazas cibernéticas son cada vez más sofisticadas.

Además, la transmisión de datos sin un cifrado adecuado incrementa el problema de seguridad. Aunque el cifrado de datos en tránsito es una práctica comúnmente recomendada, la realidad es que muchas implementaciones utilizan algoritmos débiles o desactualizados que no ofrecen la protección necesaria. Esto permite que los datos sean interceptados y leídos por terceros durante su transmisión, comprometiendo así la confidencialidad de la información. Estas vulnerabilidades técnicas se ven aumentadas por deficiencias en la implementación de medidas de seguridad en los servicios basados en OData.

Muchas organizaciones desconocen las buenas prácticas de seguridad o no siguen estándares reconocidos. Esto se traduce en configuraciones por defecto que no son seguras y en políticas de acceso mal configuradas, lo que facilita los accesos no autorizados y la manipulación indebida de los datos. La falta de una cultura de seguridad sólida y la ausencia de formación adecuada en seguridad informática contribuyen a que estas deficiencias persistan.

No contar con suficientes herramientas maduras para realizar pruebas de seguridad y evaluaciones de penetración añade otra capa de vulnerabilidad. Las pruebas de seguridad son esenciales para identificar y corregir fallos antes de que sean explotados por atacantes. Para la empresa comercializadora de calzado, esto significa que las vulnerabilidades pueden permanecer ocultas hasta que sean explotadas, causando daños que podrían haber sido evitados. Sin embargo, no cuentan con el personal especializado ni con las herramientas necesarias para realizar estas evaluaciones de manera efectiva. Esto deja a los sistemas expuestos y sin una defensa adecuada contra posibles ataques.

En el ámbito operativo, los problemas de seguridad pueden resultar en una reducción significativa de la productividad. Los incidentes de seguridad requieren tiempo y recursos para ser gestionados y solucionados, lo que desvía la atención de las actividades productivas. Las interrupciones en el servicio debido a ataques o fallos de seguridad pueden causar tiempos de inactividad prolongados, afectando la continuidad del negocio y aumentando los costos operativos. Estos costos incluyen no solo la reparación y mitigación de los daños, sino también la implementación de medidas adicionales de seguridad para prevenir futuros incidentes.

- Formulación del problema

La seguridad de la información al utilizar servicios implementados mediante el protocolo OData es un problema complejo y que se puede abordar desde diferentes puntos de vista. Es vital que la empresa comercializadora de calzado adopte un enfoque proactivo en la implementación de medidas de seguridad robustas, incluidas autenticación multifactor, prevención de inyección de consultas y cifrado adecuado de datos en tránsito. Además, es esencial fomentar una cultura de seguridad, seguir estándares reconocidos y utilizar herramientas de pruebas de seguridad para identificar y corregir vulnerabilidades. Solo a través de un esfuerzo coordinado y continuo se puede garantizar la seguridad de la información y la eficiencia operativa en el uso de servicios basados en OData.

- Pregunta central

¿Cómo mejorar la seguridad de la información utilizando servicios implementados mediante el protocolo OData para proteger la integridad,

confidencialidad y disponibilidad de los datos en una empresa comercializadora de calzado?

- Preguntas Auxiliares

    - ¿Qué medidas de autenticación y autorización son más efectivas para proteger los servicios OData contra accesos no autorizados?

    - ¿Qué herramienta se puede implementar para realizar evaluaciones de seguridad automatizadas en el ciclo de desarrollo de servicios OData?

    - ¿Qué tecnologías y arquitecturas pueden ser adoptadas para mitigar los riesgos de interrupciones y disponibilidad en los servicios OData?

1. JUSTIFICACIÓN

La creciente digitalización en el sector comercial ha incrementado la necesidad de integrar y compartir información de manera eficiente y segura. En este contexto, el protocolo OData se ha convertido en una herramienta muy utilizada para implementar servicios de integración gracias a su flexibilidad y facilidad de uso. Sin embargo, esta popularidad no está exenta de riesgos, y la falta de medidas de seguridad adecuadas en las implementaciones de OData puede tener consecuencias graves para las organizaciones.

En este marco, la línea de investigación se enfoca en dispositivos y sistemas para incrementar la seguridad al utilizar tecnologías de la información y comunicaciones. Por esta razón, es necesario asegurar la información en una empresa comercializadora de calzado, donde la confidencialidad, integridad y disponibilidad de los datos son fundamentales. La seguridad de la información no solo protege los datos sensibles de la organización, sino que también es crucial para mantener la confianza y la competitividad en el mercado.

La empresa enfrenta varias vulnerabilidades al utilizar APIs basadas en OData, entre las que se incluyen la falta de mecanismos robustos de autenticación y autorización, la susceptibilidad a la inyección de consultas, y la transmisión de datos sin cifrado adecuado. Estas deficiencias no solo exponen la información a accesos no autorizados y posibles manipulaciones, sino que también pueden llevar a interrupciones operativas significativas y a pérdidas financieras.

Las consecuencias del problema van más allá de la exposición de información sensible y los riesgos financieros. Otras implicaciones de no abordar adecuadamente las deficiencias de seguridad en las implementaciones son los bloqueos en la cadena de suministro. Los ataques a los sistemas de información pueden causar interrupciones en este proceso, afectando la disponibilidad de productos y la capacidad de cumplir con los pedidos de los clientes. Adicionalmente pueden causar demoras, pérdidas de ventas y daños a las relaciones con los proveedores.

A través de esta investigación, se busca no solo identificar los riesgos presentes en las implementaciones actuales de OData, sino también proponer soluciones prácticas y efectivas que la empresa pueda adoptar. La seguridad de la información es un desafío continuo que requiere un enfoque proactivo y una mejora constante.

Este trabajo no solo identificará las debilidades actuales, sino que también proporcionará un marco para la evaluación continua y la actualización de las medidas de seguridad. Las mejores prácticas de seguridad serán elementos clave para asegurar que las soluciones implementadas sean sostenibles y efectivas a largo plazo.

Realizar evaluaciones continuas de seguridad y pruebas automatizadas ayudarán a identificar y corregir vulnerabilidades antes de que puedan ser explotadas, asegurando así que las medidas de seguridad se mantengan efectivas y actualizadas frente a las nuevas amenazas. Por lo tanto, este trabajo contribuirá significativamente a proteger los activos digitales de la empresa, garantizar la continuidad del negocio y mantener la confianza de los clientes y socios comerciales.

Implementar herramientas y técnicas para detectar y mitigar inyecciones de consultas, tales como validaciones estrictas de entrada y el uso de ORM (*Object Relational Mapping*), puede prevenir la ejecución de consultas maliciosas, asegurando así la integridad y confidencialidad de los datos. Abordar las deficiencias de seguridad en las implementaciones de OData no solo es una necesidad técnica, sino una prioridad para asegurar la operatividad, la continuidad y la competitividad de la empresa comercializadora de calzado en el mundo digital actual, lo que tiene un impacto directo en su sostenibilidad, crecimiento y éxito a largo plazo.

La implementación de MFA (*Multi Factor Authentication*) añade una capa adicional de seguridad al requerir múltiples formas de verificación antes de conceder acceso a los datos. Esto reduce significativamente el riesgo de accesos no autorizados, ya que incluso si un atacante compromete una credencial, aún necesitaría superar las barreras adicionales.

Proteger la información crítica asegura que la empresa pueda operar de manera continua y eficiente, sin interrupciones causadas por incidentes de seguridad. La empresa fortalecerá sus relaciones comerciales y podrá crecer en el mercado como una entidad segura y confiable. Finalmente, las mejoras en seguridad permitirán a la organización adaptarse rápidamente a cambios en el entorno tecnológico y a nuevas amenazas, asegurando así su competitividad y éxito en el futuro.

2. OBJETIVOS

2.1. General

Mejorar la seguridad de la información en una empresa comercializadora de calzado utilizando servicios implementados mediante el protocolo OData para proteger la integridad, confidencialidad y disponibilidad de los datos.

2.2. Específicos

- Identificar y aplicar medidas de autenticación y autorización efectivas para proteger los servicios OData contra accesos no autorizados.

- Implementar una herramienta de evaluación de seguridad automatizada para garantizar la detección temprana de vulnerabilidades en el ciclo de desarrollo de servicios OData.

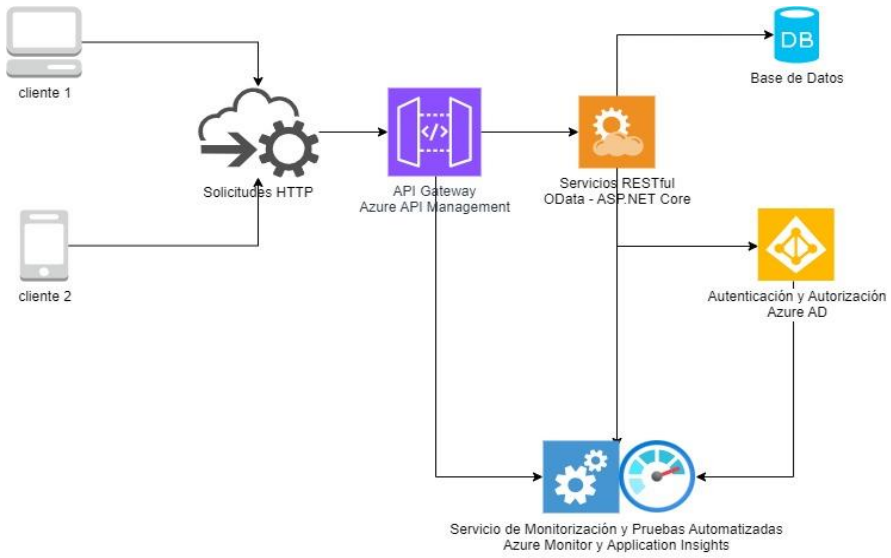
- Adoptar tecnologías y arquitecturas para mitigar los riesgos de interrupciones y garantizar la disponibilidad de los servicios OData.

3. NECESIDADES POR CUBRIR Y ESQUEMA DE SOLUCIÓN

En la era de la digitalización y el acceso instantáneo a la información, las aplicaciones *web* y móviles requieren de arquitecturas de servicios robustas, escalables y seguras. Por ello, resulta indispensable garantizar la protección de la información, donde la confidencialidad, integridad y disponibilidad de los datos son esenciales. Abordar estas áreas críticas con soluciones específicas y bien definidas permitirá a la empresa comercializadora de calzado mitigar los riesgos asociados con el uso de OData, protegiendo sus datos y asegurando la continuidad de la operación.

Figura 1.

Esquema de solución



Nota. Arquitectura de implementación de servicios Restful utilizando protocolo OData. Elaboración propia, realizado con Drawio.

Como se puede observar en la Figura 1, la arquitectura propuesta se compone de un API *gateway*, los servicios RESTful con OData, la autenticación y autorización, la base de datos y el servicio de monitorización y pruebas automatizadas. El API *gateway* actúa como el punto de entrada único para todas las solicitudes, manejando el enrutamiento, la autenticación, el balanceo de carga y la limitación de velocidad. Los servicios RESTful con OData exponen datos y funcionalidades utilizando el protocolo OData, que permite la consulta y manipulación de datos mediante URLs. La autenticación y autorización gestionan la autenticación multifactor y la autorización basada en roles, garantizando que solo los usuarios legítimos puedan acceder al sistema. La base de datos almacena los datos que se exponen a través de los servicios RESTful. Finalmente, el servicio de monitorización y pruebas automatizadas mide el rendimiento y la seguridad del servicio.

Para la implementación del API *gateway*, Azure API Management es la herramienta propuesta, proporcionando capacidades avanzadas de gestión de API, incluyendo autenticación, limitación de velocidad, monitoreo y análisis. Azure API Management facilita la creación, publicación, mantenimiento, monitoreo y protección de APIs RESTful a cualquier escala.

La autenticación multifactor (MFA) añade una capa adicional de seguridad al proceso de inicio de sesión, asegurando que solo los usuarios legítimos puedan acceder al sistema. En este proceso, el usuario primero ingresa su nombre de usuario y contraseña. Luego, el sistema envía un código de verificación a un segundo factor. El usuario ingresa el código recibido y el sistema lo verifica para autenticar al usuario. La autorización, por otro lado, controla el acceso a recursos y funcionalidades basándose en roles y permisos definidos en la base de datos. Los tokens de autenticación incluyen los roles

del usuario y se verifican en cada solicitud, asegurando que solo los usuarios con los permisos adecuados puedan acceder a ciertos recursos.

Para implementar estas funcionalidades, Microsoft proporciona varias herramientas y tecnologías. Azure Active Directory (Azure AD) se utiliza para la gestión de identidades y el control de acceso, soportando la autenticación multifactor (MFA) y la autorización basada en roles. Microsoft Identity es una biblioteca para la integración de la autenticación y autorización en aplicaciones .NET.

Los servicios RESTful con OData se implementarán utilizando ASP.NET Core con OData. ASP.NET Core proporciona un *framework* robusto y escalable para construir aplicaciones *web* y APIs, mientras que OData añade capacidades avanzadas de consulta y manipulación de datos. En ASP.NET Core, se configuran las rutas OData y se crean controladores y modelos para exponer datos a través de OData.

Las pruebas automatizadas son fundamentales para asegurar la calidad y disponibilidad del servicio. Las métricas para evaluar incluyen el porcentaje de tiempo de disponibilidad del servicio (*uptime*), el número de usuarios autenticados, el número de intentos de acceso no autorizado bloqueados, el número de vulnerabilidades detectadas en las pruebas, el tiempo de inactividad del servicio y el tiempo medio de recuperación ante desastres.

El servicio de monitorización y pruebas automatizadas se puede implementar utilizando Azure Monitor, que proporciona una plataforma para la monitorización de aplicaciones, incluyendo la recolección de métricas, *logs* y alertas. Azure Application Insights para el monitoreo de infraestructura y

aplicaciones, se integra con Azure Monitor para ofrecer un análisis detallado del rendimiento de las aplicaciones. Azure DevOps proporciona herramientas para la integración y la entrega continuas (CI/CD), incluyendo la ejecución de pruebas automatizadas con Selenium y JMeter.

4. ALCANCES

4.1. Alcance investigativo

Esta investigación se centra en analizar y proponer medidas para mejorar la seguridad de la información en servicios implementados mediante el protocolo OData. Se identifican y evalúan diversas estrategias de autenticación, autorización y cifrado, con el objetivo de proteger la integridad, confidencialidad y disponibilidad de los datos. Se considera la implementación de autenticación multifactor, la prevención de inyección de consultas y el cifrado adecuado de datos en tránsito como soluciones clave para abordar las vulnerabilidades inherentes a OData. También se analizan las deficiencias en las prácticas de seguridad dentro de la organización y se propone un enfoque integral que incluya tanto medidas técnicas como una cultura organizacional de seguridad sólida.

4.2. Alcance técnico

El proyecto define el diseño de un prototipo que incorpora diversas herramientas y tecnologías para fortalecer la seguridad en servicios basados en OData, considerando las siguientes características:

- Procedimiento para implementar autenticación multifactor: Establecer mecanismos robustos de autenticación para proteger contra accesos no autorizados.

- Diseño de una arquitectura segura: Incluir medidas preventivas como la detección y mitigación de inyección de consultas, asegurando que las configuraciones por defecto no sean inseguras.

- Evaluaciones de seguridad automatizadas: Integrar herramientas de pruebas de seguridad automatizadas en el ciclo de desarrollo de servicios OData para identificar y corregir vulnerabilidades de manera proactiva.

4.3. Resultados esperados

El desarrollo de esta investigación debe proporcionar una solución integral para mejorar la seguridad de los servicios implementados mediante OData en la empresa comercializadora de calzado, incluyendo:

- Un enfoque de autenticación y autorización efectivo: Propuestas concretas para implementar autenticación multifactor y políticas de acceso seguras.

- Arquitectura robusta y segura: Un diseño arquitectónico que prevenga vulnerabilidades comunes en implementaciones OData y mejore la disponibilidad del servicio.

- Definición de medidas de seguridad: Iniciativas para fortalecer la formación en seguridad informática y la adopción de estándares reconocidos dentro de la empresa.

5. MARCO TEÓRICO

5.1. Servicios de Transferencia de Estado Representacional (REST)

REST, en la arquitectura de *software*, es un estilo que define condiciones y restricciones, por ejemplo, una interfaz uniforme, que, al utilizarse para servicios *web*, debe de implementar propiedades como rendimiento, escalabilidad y capacidad de modificación, optimizando así el funcionamiento de los servicios en la *web* (Cupek y Huczala, 2015). En este estilo, los datos y las funcionalidades se tratan como recursos a los que se accede mediante Identificadores Uniformes de Recursos (URIs), y que comúnmente están vinculados en la *web*. La forma de gestionar los recursos se, es mediante operaciones simples y bien definidas.

La arquitectura REST es una arquitectura cliente/servidor que utiliza un protocolo de comunicación sin estado, como HTTP. Los clientes y servidores interactúan a través de representaciones de recursos. En lugar de comunicarse directamente sobre los datos, se envían versiones de esos datos, como JSON o XML utilizando una interfaz y un protocolo estándar, tal como proponen Cupek et al. (2015).

El diseño y la adaptabilidad de las API REST son esenciales para la comunicación en arquitecturas distribuidas. Debido a que los servicios experimentan cambios y actualizaciones frecuentes, las API REST deben ser flexibles y capaces de evolucionar sin afectar a los clientes. Li y colaboradores (2016) demuestran que una API REST puede ser diseñada para facilitar la

navegación por hipertexto, lo que permite enfrentar cambios en la estructura de la API sin depender excesivamente de su diseño inicial.

El objetivo principal de la navegación por hipertexto es manejar los cambios en la API REST al reducir la dependencia entre el cliente y la API, permitiendo que el cliente navegue hacia los recursos objetivo según el hipertexto (Li et al., 2016). Este enfoque se basa en cinco capas dentro de una API REST:

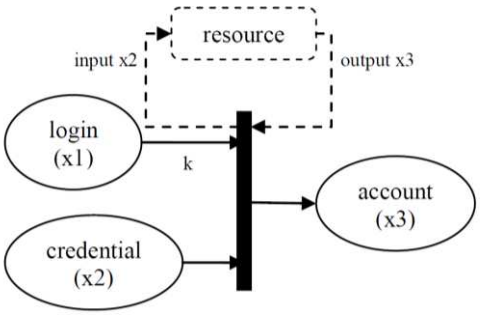
- Conexión: enlaces implementados independiente del lenguaje de programación y ejecutados en cualquier dispositivo.
- Interacciones: métodos o protocolos, como HTTP (*Hypertext Transfer Protocol*), para interactuar con los recursos.
- Identificación: identificadores URI (*Uniform Resource Identifier*) para los recursos.
- Representación: hipertextos XML (*eXtensible Markup Language*) que se envían y reciben en las interacciones.
- Descripción: información sobre las posibles representaciones de recursos.

La accesibilidad de la red de Petri determina la navegación por hipertexto en una API REST (Li y colaboradores, 2016). Se modelan los estados del cliente como la distribución de tokens en dicha red. Por ejemplo, en la Figura 2 se muestra que el estado inicial del cliente es (*x1*, 0, 0), lo que indica que tiene el *token x1* en el lugar de *login*, pero no en los lugares de *credential* y *account*. Desde este estado inicial, el REST Chart muestra que el estado final (0, 0, *x3*) se puede alcanzar de la siguiente forma:

$$(x1, 0, 0) \rightarrow (x1, x2, 0) \rightarrow (0, 0, x3).$$

Figura 2.

Ilustración de REST Chart



Nota. Ilustración de un REST Chart básico. Obtenido de Li, L., Chou, W., Zhou, W., & Luo, M. (2016). *Design patterns and extensibility of REST API for networking applications.* (https://ieeexplore.ieee.org/document/7378522), consultado 14 de septiembre de 2024. De dominio público.

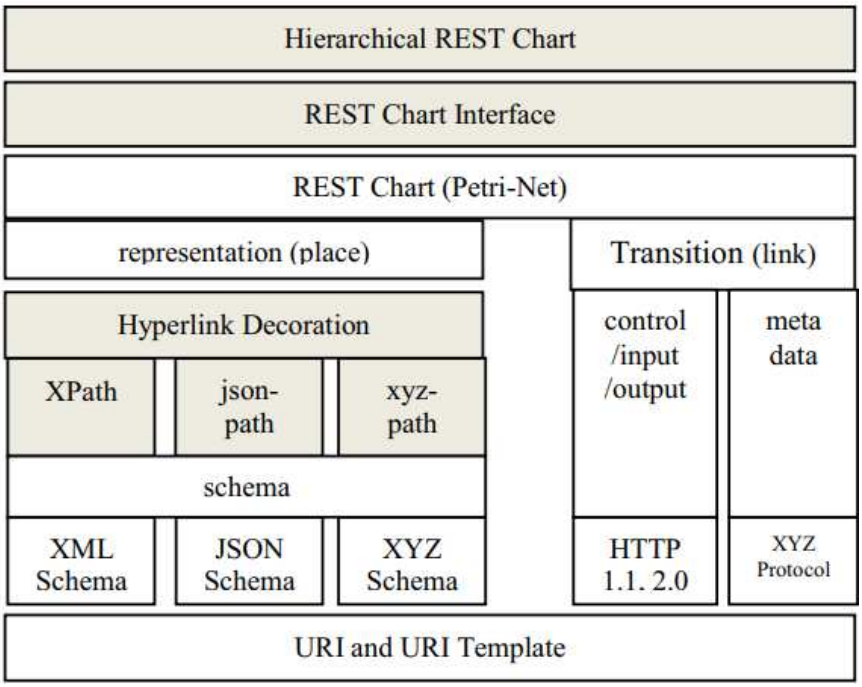
No todos los estados posibles se pueden alcanzar desde el estado inicial, y tales estados incluyen (*x1*, 0, *x3*) y (0, *x2*, *x3*). El estado (*x1*, 0, *x3*) significa que un cliente puede acceder a la cuenta sin una credencial válida, y el estado (0, *x2*, *x3*) significa que el cliente puede acceder a la cuenta sin seguir el hipervínculo. Según Li y colaboradores (2016), una característica importante del REST Chart es que el conjunto de estados alcanzables define las conexiones de recursos que los clientes pueden seguir mediante la navegación por hipertexto. Si un cliente puede alcanzar un estado inalcanzable, esto indica que la API REST viola el principio de navegación impulsada por hipertexto.

La ilustración de la arquitectura en capas del REST Chart se muestra en la figura 3. Este enfoque en capas incluye la parte principal del Petri-Net en la parte superior que conecta pilas de tecnologías de servicios REST en el

medio, como tipos de medios y protocolos de red, los cuales dependen de URI y URI *Template* en la parte inferior.

Figura 3.

Arquitectura de capas de REST Chart



Nota. Arquitectura de capas de REST Chart ilustrada. Obtenido de Li, L., & Chou, W. (2015). *Designing large scale REST APIs based on REST chart*. (https://ieeexplore.ieee.org/document/7195624), consultado 14 de septiembre de 2024. De dominio público.

El REST Chart Jerárquico proporciona un método para descomponer y extender una API REST en varias dimensiones, haciendo más manejable el desarrollo de APIs complejas y sujetas a cambios frecuentes. Estas mejoras se integran fácilmente en el XML del REST Chart con pequeñas modificaciones en la sintaxis, facilitando la creación de componentes

reutilizables tanto para el cliente como para el servidor. Estas innovaciones han demostrado ser efectivas en el diseño y verificación de APIs REST para entornos de computación en la nube y distribuidas (Li y Chou, 2015).

5.1.1. Protocolo OData en arquitecturas orientadas a servicios

OData, una extensión de los estándares Atom Publishing y Atom Syndication, fue desarrollada por Microsoft. Estos estándares están fundamentados en XML y HTTP(s), proporcionando una base sólida para la estructuración de datos. La capacidad de extensibilidad de OData permite la incorporación de información adicional específica del dominio a los tipos de datos. Atom, en sí mismo, comprende dos componentes clave: el Formato de Sindicación Atom, una estructura XML que describe los documentos ofrecidos por un sitio *web* (*Feed*), y el Formato de Publicación Atom, un protocolo HTTP que facilita la manipulación de documentos de Sindicación Atom (Cupek y Huczala, 2015).

Es relevante mencionar que el Formato de Publicación Atom no establece un método específico para codificar los datos dentro de un *feed*, lo que limita su conformidad con los principios RESTful, ya que sus mensajes carecen de la autodescripción necesaria. Según Cupek et al. (2015), OData amplía Atom al introducir una descripción de metadatos que incluye: tipos de datos simples y complejos, asociaciones y rutas de navegación entre entradas, así como comportamientos personalizados más allá de las operaciones CRUD estándar (del inglés *Create, Read, Update* y *Delete*). OData admite mensajes tanto en XML como en JSON (*JavaScript Object Notation*).

La tecnología OData se compone de cuatro elementos principales. El modelo de datos OData proporciona un método genérico para organizar y describir datos. El protocolo OData para facilitar a los clientes la realización de solicitudes y la obtención de respuestas de un servicio OData. Las bibliotecas de cliente OData ayudan en la creación de *software* que accede a datos a través del protocolo OData. Finalmente, el servicio OData expone un punto final que permite el acceso a los datos.

Los servicios OData se definen a través de un modelo de datos común, que es publicado en un formato legible, permitiendo una interacción clara y bien definida para los clientes. Un servicio OData expone dos recursos esenciales: un documento de servicio, que enumera los conjuntos de entidades y funciones disponibles, y un documento de metadatos, que detalla los tipos, conjuntos, funciones y acciones soportadas por el servicio (Cupek y Huczala, 2015). El documento de servicio permite a los clientes navegar por el modelo, mientras que el documento de metadatos ofrece una guía para la consulta e interacción con las entidades dentro del servicio.

5.2. Diseño y arquitectura de sistemas de autenticación y autorización

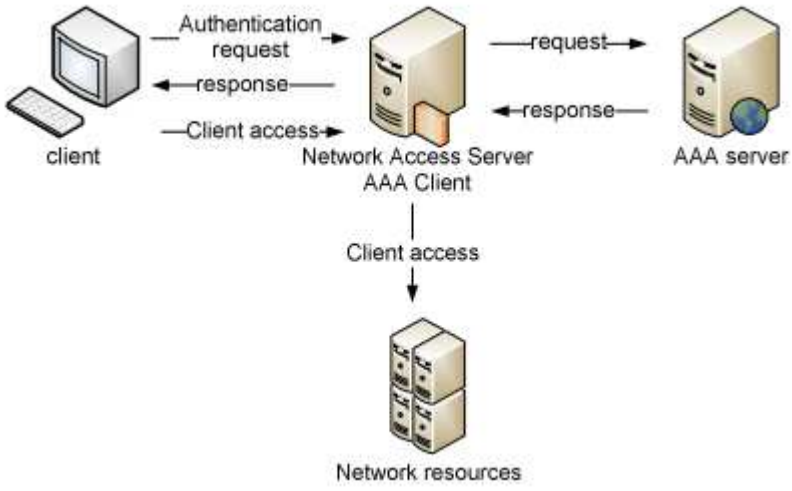
La seguridad en redes tradicionales y modernas es fundamental para asegurar la integridad y autenticidad de los datos. El modelo de Autenticación, Autorización y Contabilidad (AAA) es un enfoque adaptable y comúnmente utilizado para proteger redes, particularmente en entornos móviles (Papatheodoulou et al., 2009). La autenticación verifica la autenticidad de los datos almacenados y recibidos, mientras que la autorización otorga permisos específicos a clientes con credenciales verificadas. La contabilidad, dividida

en recopilación de métricas y análisis de tendencias, se encarga de registrar el uso de recursos por parte de los clientes.

En el diseño propuesto por Papatheodoulou et al. (2009), la arquitectura de autenticación utiliza el Protocolo de Autenticación Extensible (EAP) junto con la función *hash* MD5 y bases de datos SQL para almacenar las credenciales de autenticación. EAP, ampliamente reconocido y utilizado en redes móviles, no solo es relevante en AAA, sino también en diversos protocolos y aplicaciones de red actuales.

Figura 4.

Esquema de autenticación

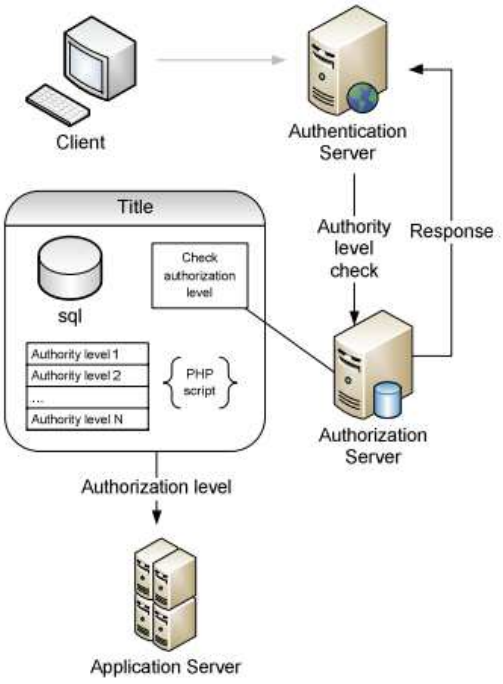


Nota. Autenticación en el Protocolo AAA. Obtenido de Papatheodoulou, N., & Sklavos, N. (2009). *Architecture & system design of Authentication, Authorization, & Accounting services*. (https://ieeexplore.ieee.org/document/5167894), consultado 14 de septiembre de 2024. De dominio público.

La Figura 4 muestra en el modelo de autenticación, como el cliente se comunica con el Servidor de Acceso a la Red (NAS), y luego reenvía las solicitudes al servidor de recursos. Cada conjunto de credenciales es único y está encriptado con MD5. El NAS envía la solicitud de autenticación al servidor AAA, que verifica la autenticidad de las credenciales mediante *scripts* conectados a la base de datos. Si la autenticación es exitosa, el servidor AAA permite el acceso del cliente a los recursos de la red por un tiempo limitado, determinado por las políticas de la sesión.

Figura 5.

Esquema de autorización



Nota. Autorización en el Protocolo AAA. Obtenido de Papatheodoulou, N., & Sklavos, N. (2009). *Architecture & system design of Authentication, Authorization, & Accounting services.* (https://ieeexplore.ieee.org/document/5167894), consultado 21 de septiembre de 2024. De dominio público.

El flujo de cómo el proceso de autorización se comunica con el proceso de autenticación se muestra en la figura 5. Después de finalizar correctamente el procedimiento de autenticación, el servidor envía una solicitud al servidor de autorización para verificar el nivel de autorización del cliente.

Para el proceso de contabilidad se utilizan algunos contadores, por ejemplo: registrar el dato de la fecha y hora del inicio y finalización de la sesión, la IP de cada sesión y la duración del uso del recurso o su tamaño. Luego, todos los datos se enviarían a otro servidor para iniciar el proceso de almacenamiento y análisis.

5.2.1. Gestión de identidad y acceso para servicios *web*

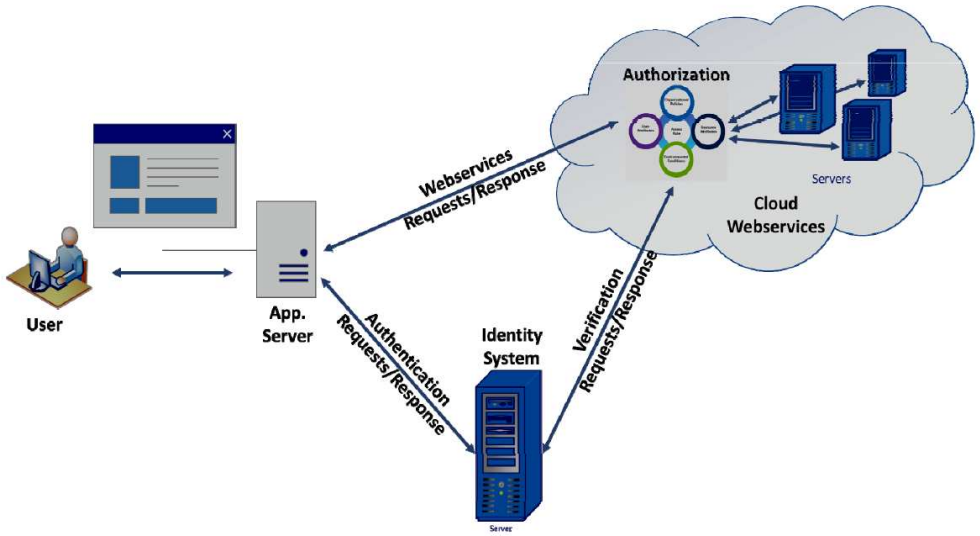
La computación Cloud o en la nube, se ha convertido en fundamental para las necesidades organizacionales modernas, destacándose por su flexibilidad y eficiencia de costos. Sin embargo, como advierten Indu et al. (2015), las opciones de servicios Cloud, como IaaS (*Infrastructure as a Service*), PaaS (*Platform as a Service*), SaaS (*Software as a Service*), y entornos *multi-tenant*, presentan riesgos significativos en términos de privacidad y seguridad. Para mitigar estos riesgos, las organizaciones necesitan un sistema de Gestión de Identidad y Acceso (IAM) robusto y escalable.

Indu y Anand (2015) proponen un sistema integrado de gestión para la identidad y control de acceso fundamentado en atributos para servicios *web* en la nube. Este enfoque combina la autenticación y el control de acceso que se basa en atributos, mejorando la seguridad al garantizar que solo usuarios autorizados accedan a los recursos bajo condiciones específicas. La Figura 6, representa en este modelo, como la autenticación exitosa genera un *token* de

acceso, que es validado y utilizado para la autorización basada en atributos en el servidor *web* de la nube.

Figura 6.

Modelo gestión de identidades y acceso



Nota. Modelo integrado de gestión de identidades y accesos para servicios web en la nube. Obtenido de Indu, I., & Anand, P. R. (2015). *Identity and access management for cloud web services*. (https://ieeexplore.ieee.org/document/7488450), consultado 21 de septiembre de 2024. De dominio público.

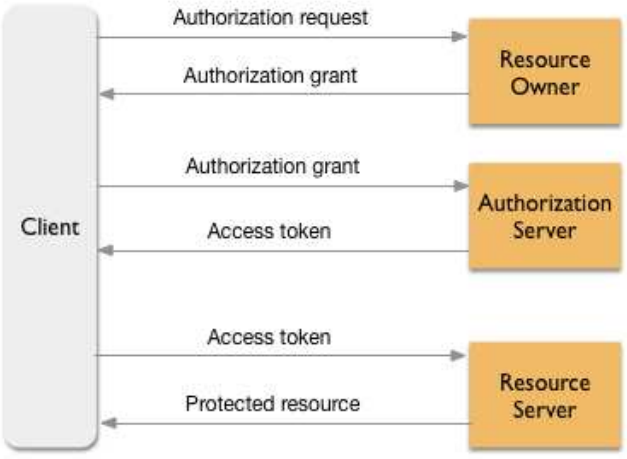
5.2.2. OAuth2 en la autorización de servicios *web* en la nube

OAuth 2, un marco de autorización *web*, permite que los servicios actúen en nombre de los usuarios al interactuar con otros servicios, evitando la necesidad de compartir credenciales sensibles entre ellos (Sendor et al., 2014). Aunque este protocolo es crucial para proteger a los usuarios, su implementación puede ser compleja y propensa a vulnerabilidades.

En un modelo de arquitectura cliente-servidor, el cliente utiliza sus credenciales para solicitar acceso a los recursos protegidos que se encuentran en el servidor. Si un tercero necesita acceder a esos recursos en representación del propietario, este último debe compartir sus credenciales con el tercero. Este enfoque puede generar varios problemas indeseables, como la duplicación de las credenciales del propietario en la ubicación del tercero, otorgar acceso total a los recursos sin restricciones, o la complejidad de revocar los derechos de acceso, lo cual requeriría un cambio de credenciales. OAuth se diseñó precisamente para resolver estos problemas, permitiendo el acceso sin necesidad de compartir las credenciales (Sendor et al., 2014).

Figura 7.

Flujo general de OAuth 2



Nota. Diagrama de flujo de OAuth2. Adaptado de Sendor, J., Lehmann, Y., Serme, G., & de Oliveira, A. S. (2014, March). *Platform-level support for authorization in cloud services with OAuth 2.* (https://ieeexplore.ieee.org/document/6903511), consultado 21 de septiembre de 2024. De dominio público.

OAuth define cuatro roles clave: el propietario del recurso, el servidor de recursos, cliente y servidor de autorización. El flujo general del protocolo, según la Figura 7, implica que el cliente solicite autorización del propietario del recurso para acceder a los activos protegidos. Si el propietario del recurso concede la autorización, el cliente obtiene un token de acceso del servidor de autorización, que luego utiliza para solicitar los recursos protegidos. Mientras el token de acceso no haya expirado y siga siendo válido, el cliente puede acceder a los recursos.

5.3. Comprobación y pruebas de las propiedades de los servicios REST

En el contexto de servicios en la nube accesibles a través de APIs REST, un cliente envía solicitudes a un servicio y recibe respuestas basadas en el protocolo HTTP/S, donde cada respuesta incluye un código de estado HTTP. Atlidakis et al. (2020) definen una API REST como un conjunto finito de solicitudes, donde cada solicitud es una combinación de un *token* de autenticación, tipo de solicitud, ruta de recurso y cuerpo de la solicitud. Cada solicitud *r* es una tupla de la forma *(a, t, p, b)* donde:

- a es un token de autenticación,
- t es el tipo de solicitud,
- p es una ruta de recurso, y
- b es el cuerpo de la solicitud.

Un tipo de solicitud *t* es cualquiera de los siguientes cinco valores permitidos por REST: PUT o POST (crear o actualizar), GET (leer o consultar), DELETE (eliminar), PATCH (actualizar). La ruta de recurso *p* es una cadena que identifica un recurso en la nube y su jerarquía principal. Comúnmente, *p*

es una cadena con la expresión: *(/resourceType/resourceName/).* Donde *resourceType* denota el tipo de un recurso en la nube y *resourceName* es el nombre específico del recurso de ese tipo. El cuerpo de la solicitud *b* puede incluir parámetros adicionales y sus valores que pueden ser requeridos u opcionales para que la solicitud se ejecute con éxito (Atlidakis et al., 2020).

Atlidakis et al. (2020) también presentan las siguientes cuatro reglas de seguridad para APIs REST y servicios:

- Acceso tras eliminación: Una vez que un recurso ha sido eliminado, no debe ser posible acceder a él. Cualquier intento de acceder, modificar o eliminar el recurso tras su eliminación debe resultar infructuoso.

- Pérdida de recursos: Si se produce un fallo durante la creación de un recurso, este no debe existir ni ser accesible, y su falla no debe impactar el estado del servicio ni consumir recursos, como la cuota.

- Estructura jerárquica de recursos: Un recurso hijo no debería ser accesible desde un recurso padre equivocado. Cada recurso hijo debe estar vinculado únicamente al recurso padre que lo originó.

- Pérdida de recursos anidados: Si se elimina un recurso padre, todos sus recursos hijos también deben dejar de estar disponibles o visibles.

5.3.1. Pruebas automatizadas de caja blanca a servicios RESTful

Arcuri (2017) señala que generar datos de prueba es una tarea compleja debido a la variabilidad del *software*, y que crear manualmente casos

de prueba puede resultar complicado para los desarrolladores. Aunque la generación aleatoria de casos de prueba es sencilla, no suele ser efectiva ya que puede cubrir solo pequeñas porciones del *software*. En su lugar, la ingeniería de *software* basada en búsqueda ha demostrado ser más eficiente para resolver problemas de pruebas de *software*, empleando algoritmos de búsqueda para optimizar tanto la cobertura del código como la detección de fallos.

Para las APIs RESTful, las pruebas consisten en solicitudes HTTP, que pueden ser complejas debido a la variedad de formatos de contenido, aunque JSON es el formato más común. Swagger es una herramienta popular para documentar APIs REST. Las pruebas generadas pueden ser útiles tanto para pruebas de regresión como de seguridad, detectando fallos en la verificación de autorizaciones y utilizando códigos de estado HTTP. El empleo de un Algoritmo Genético (GA) para generar casos de prueba permite crear conjuntos de pruebas que cubren tanto la cobertura de declaraciones del sistema como varios códigos de estado HTTP. El GA evoluciona conjuntos de pruebas al combinar y modificar casos existentes, optimizando así la calidad de las pruebas (Arcuri, 2017).

Arcuri (2017) propone un enfoque totalmente automatizado para las pruebas de caja blanca, empleando un algoritmo evolutivo para la generación de casos de prueba. Este enfoque está diseñado para maximizar la cobertura del código y detectar errores a través del análisis de los códigos de estado HTTP. Al tener acceso al código fuente, el método permite una evaluación detallada de la lógica interna y la estructura del código, identificando errores que podrían pasar inadvertidos con métodos de caja negra. Al diseñar pruebas que abarquen todas las rutas posibles, se incrementa considerablemente la efectividad de las pruebas.

5.3.2. Pruebas de caja negra a servicios RESTful

Viglianisi et al. (2020) proponen que cuando el código fuente no está disponible o es difícil de analizar, las pruebas de caja negra son una opción adecuada, ya que requieren solo el acceso al sistema a través de una interfaz específica. Partiendo de la definición de la interfaz (Swagger), se generan valores de entrada y solicitudes para cada operación de la API, con el fin de evaluar tanto escenarios nominales como situaciones de error. La investigación empírica realizada demuestra que esta herramienta es eficaz para detectar fallos en APIs REST reales.

Se utiliza la especificación Swagger para construir un grafo de dependencias de operaciones (ODG), que organiza las pruebas en función de las dependencias de datos entre las operaciones. El ODG es un grafo dirigido $G=(N,V)$, donde los nodos N representan las operaciones en la API REST. Existe una arista $v \in V$, con $v = n_2 \rightarrow n_1$, cuando hay una dependencia de datos entre n_2 y n_1. Esta dependencia se establece si un campo de la salida (respuesta) de n_1 coincide con un campo de la entrada (solicitud) de n_2. Intuitivamente, esto significa que n_1 debe ser probado antes que n_2, ya que la salida de n_1 podría ser utilizada para deducir los valores de entrada necesarios para probar n_2 (Viglianisi et al., 2020).

Laranjeiro et al. (2021) desarrollaron una herramienta para pruebas de caja negra. Esta herramienta genera solicitudes tanto válidas como inválidas para evaluar los servicios REST. La metodología se desarrolla en cuatro etapas: primero, se analiza la descripción de la interfaz para extraer información sobre las operaciones y los datos de entrada/salida a partir del documento de descripción de la API; luego, se generan y ejecutan cargas de trabajo válidas para comprender el comportamiento del servicio bajo

condiciones normales; a continuación, se crean y ejecutan cargas con fallos mediante la inyección de errores en los parámetros de las solicitudes válidas para provocar comportamientos incorrectos; finalmente, se almacenan y analizan los resultados, incluyendo las respuestas del servicio y la metadata de las pruebas.

6. PROPUESTA DE ÍNDICE DE CONTENIDOS

7. METODOLOGÍA

7.1. Características del estudio

La investigación tendrá un enfoque mixto. El tipo de estudio es cualitativo y cuantitativo.

Se considera cuantitativo, ya que se evaluará la efectividad de las mejoras en seguridad a través de métricas antes y después de la implementación. Se llevarán a cabo pruebas de rendimiento utilizando herramientas para medir la capacidad de la arquitectura propuesta de manejar cargas de trabajo y responder a incidentes de seguridad.

Desde una perspectiva cualitativa, se mostrará la calidad de la arquitectura de seguridad propuesta evaluando la adecuación de las medidas de autenticación, autorización y respuesta a incidentes desde el punto de vista de la organización.

El estudio será descriptivo y explicativo. Desde el enfoque descriptivo, permitirá una comprensión clara de las medidas y herramientas de seguridad implementadas para proteger la integridad, confidencialidad y disponibilidad de los datos en servicios basados en el protocolo OData. Esto incluye un análisis detallado de los componentes de la arquitectura propuesta y cómo estos elementos interactúan para garantizar la seguridad de la información. También se detallarán las herramientas utilizadas para las pruebas de calidad y su impacto en la identificación de vulnerabilidades.

Por otra parte, el enfoque explicativo proporcionará una visión de cómo y por qué estas medidas son efectivas para mejorar la seguridad de los servicios OData. Se buscará explicar la relación entre las diferentes medidas de seguridad implementadas y su efectividad en la mitigación de riesgos específicos. El estudio analizará cómo las pruebas automatizadas y las herramientas de evaluación de calidad contribuyen a mejorar la seguridad de la arquitectura propuesta, destacando las razones detrás del éxito o fracaso de ciertas medidas en escenarios específicos.

El diseño del estudio será experimental, centrado en la implementación y validación de un prototipo seguro basado en servicios OData. La arquitectura propuesta se evaluará a través de pruebas de calidad utilizando herramientas como para la automatización de pruebas funcionales y para pruebas de rendimiento, simulando múltiples escenarios de uso e intentos de autenticación y autorización. También se emplearán herramientas para la recolección de métricas de rendimiento y análisis de seguridad en tiempo real.

La validación de la arquitectura se realizará mediante la ejecución de pruebas de estrés y resistencia para evaluar la robustez del sistema ante picos de demanda y fallas del sistema. Estas pruebas permitirán identificar posibles puntos de fallo y áreas de mejora en la arquitectura de seguridad.

7.2. Variables

Las variables que se estarán abordando en el estudio se describen a continuación.

Tabla 1.

Variables en estudio

Variable	Definición teórica	Subvariables	Indicadores
Operatividad del sistema	Representa una medida del tiempo durante el cual los datos y los servicios se pueden acceder y se encuentran en estado disponible.	Disponibilidad	% de tiempo de disponibilidad del servicio
		Inactividad	Tiempo de inactividad del servicio
		Recuperación	Tiempo medio de recuperación ante desastres
Seguridad del acceso al sistema	Medidas de autenticación y autorización efectivas para proteger los servicios contra accesos no autorizados	Autenticación	Número de intentos de acceso no autorizado bloqueados
		Autorización	Número de usuarios con acceso al recurso solicitado
Calidad de los servicios implementados	Pruebas de seguridad para la detección de vulnerabilidades en los servicios OData.	Confiabilidad	Número de vulnerabilidades detectadas en las pruebas

Nota: Definición de variables, subvariables e indicadores. Elaboración propia, realizado con Word.

7.3. Fases de estudio

Fase de revisión documental y literatura

En esta fase se recopilará y revisará información relevante de fuentes académicas, publicaciones científicas, documentos técnicos, normativas de seguridad, y guías de implementación de protocolos de comunicación segura y fundamentos de los servicios OData, las mejores prácticas de implementación y los requisitos específicos de seguridad. Esta fase también incluirá una revisión de herramientas de evaluación de seguridad y pruebas de calidad.

Fase de análisis de seguridad

Se realizará un análisis de los riesgos y amenazas asociados con la implementación de servicios OData. Evaluar los posibles ataques, como inyecciones de datos, ataques de denegación de servicio (DoS) y accesos no autorizados. Utilizar estudios de caso y ejemplos de vulnerabilidades reportadas en implementaciones OData.

Fase de diseño de la arquitectura de seguridad y servicios

Diseñar una arquitectura de *software* que contemple las medidas de seguridad para la implementación de servicios OData. La arquitectura incluirá la selección de herramientas, la configuración de políticas de seguridad como autenticación multifactor (MFA), y control de acceso basado en roles. Este diseño servirá para demostrar mediante un prototipo como llevar a cabo una implementación segura. Se documentará un plan detallado para la infraestructura de *hardware* y *software* necesaria.

Fase de desarrollo del prototipo

Crear un prototipo que implemente los servicios OData en un entorno seguro. Desarrollar ejemplos de código y configuraciones para los servicios RESTful con OData, incluyendo los modelos para autenticación y autorización, sistemas de monitoreo y pruebas automatizadas. Esta fase también incluirá la instalación y configuración de todas las herramientas necesarias y estará orientada a demostrar cómo se puede implementar la arquitectura propuesta.

Fase de pruebas de seguridad

Se realizarán pruebas de seguridad para evaluar la robustez de la arquitectura de seguridad propuesta mediante simulaciones de amenazas y vulnerabilidades. Se utilizarán herramientas de pruebas automatizadas para simular ataques y verificar la resistencia del sistema ante accesos no autorizados. Incluyen la evaluación de los mecanismos de autenticación multifactor y el control de acceso basado en roles para demostrar la metodología de pruebas de seguridad aplicable a servicios OData.

Fase de pruebas de calidad y rendimiento

Ejecutar pruebas de calidad y rendimiento del prototipo en un entorno controlado. Utilizar herramientas o monitores para evaluar el comportamiento del sistema bajo condiciones de carga y estrés, verificando disponibilidad, el tiempo de respuesta, y el uso de recursos del sistema.

Fase de análisis de resultados y redacción de informe final

Analizar los resultados obtenidos de las pruebas de seguridad y rendimiento, evaluando si la arquitectura cumple con los objetivos de proteger la integridad, confidencialidad y disponibilidad de los datos. Documentar las observaciones, conclusiones y recomendaciones basadas en los resultados de las pruebas.

8. TÉCNICAS DE ANÁLISIS DE LA INFORMACIÓN

Como primer punto, se realizará un análisis de contenido para revisar la información recogida de fuentes documentales, como normativas de seguridad, *papers* o publicaciones científicas y guías de implementación. El análisis se centrará en identificar patrones, temas o conceptos clave relevantes para la seguridad de los servicios OData.

Este análisis permitirá, para los datos cualitativos, realizar una teoría fundamentada, examinando cómo la arquitectura de seguridad propuesta se adapta a las necesidades. Se estará analizando la implementación de las medidas de autenticación, autorización y respuesta a incidentes en un contexto controlado, evaluando su efectividad y adecuación desde el punto de vista de la organización. Se hará un análisis de caso del prototipo, incluyendo la revisión de cómo estas medidas responden a incidentes de seguridad y su impacto en las operaciones.

Para los datos cuantitativos, se utilizará estadística descriptiva para evaluar la eficacia de las medidas de seguridad implementadas. Se recopilarán métricas de los indicadores que corresponden a cada variable de operatividad, seguridad y calidad de la arquitectura implementada como el tiempo de disponibilidad, el número y la tasa de éxito de los intentos de autenticación y la frecuencia de detección de amenazas y vulnerabilidades encontradas. Los datos serán organizados en tablas y gráficos para facilitar su interpretación y se emplearán medidas de tendencia central (media, mediana, moda) y dispersión (desviación estándar) para analizar las variaciones y tendencias en los datos.

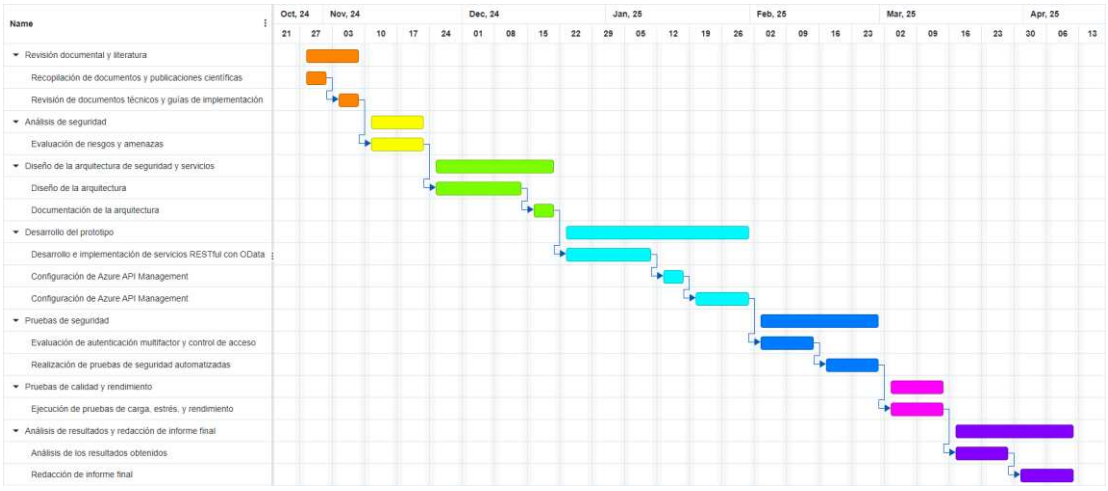
Finalmente, se elaborará un análisis comparativo para evaluar los resultados obtenidos durante las pruebas de seguridad con estándares y mejores prácticas documentadas. Este enfoque permitirá identificar áreas para mejoras y garantizar que la arquitectura cumpla con los requisitos de seguridad esperados.

9. CRONOGRAMA

A continuación, se detallan el cronograma de trabajo, las fases y la duración estimada para llevar a cabo el estudio.

Figura 8.

Diagrama de Gantt del estudio



Nota: Cronograma de las fases del estudio. Elaboración propia, realizado con OnlineGantt.

Tabla 2.

Fases del estudio

Fase	Descripción	Duración estimada
Fase de revisión documental y literatura	Recopilación de información, revisión de literatura, estudios técnicos y normativas de seguridad.	2 semanas
Fase de análisis de seguridad	Análisis de riesgos y amenazas, evaluación de posibles ataques y ejemplos de vulnerabilidades.	2 semanas
Fase de diseño de la arquitectura de seguridad y servicios	Diseño de la arquitectura, selección de herramientas y políticas de seguridad, documentación del plan.	4 semanas
Fase de desarrollo del prototipo	Desarrollo del prototipo de servicios OData, configuración de herramientas de seguridad.	6 semanas
Fase de pruebas de seguridad	Realización de pruebas automatizadas de seguridad, simulación de amenazas y evaluación de resultados.	4 semanas
Fase de pruebas de calidad y rendimiento	Pruebas de rendimiento, evaluación del comportamiento del sistema bajo carga y estrés.	2 semanas

Continuación de la tabla 2.

Fase	Descripción	Duración estimada
Fase de análisis de resultados y redacción de informe final	Análisis de los resultados de pruebas, redacción de conclusiones y recomendaciones.	4 semanas

Nota: Descripción y duración estimada de las fases del estudio. Elaboración propia, realizado con Word.

10. FACTIBILIDAD DEL ESTUDIO

10.1. Factibilidad temporal

Para evaluar la factibilidad del tiempo, se consideran las fases del estudio, las actividades específicas y su duración estimada. El objetivo es determinar que las tareas necesarias se pueden completar dentro del plazo de 6 meses (aproximadamente 26 semanas).

Fase de revisión documental y literatura (2 semanas): Recopilación de información de fuentes académicas, publicaciones científicas, documentos técnicos, normativas de seguridad, y guías de implementación de protocolos OData. También se realizará una revisión de herramientas de evaluación de seguridad y pruebas de calidad. Este tiempo es adecuado para investigar y reunir toda la información necesaria.

Fase de análisis de seguridad (2 semanas): Evaluación de riesgos y amenazas asociados con la implementación de servicios OData (inyecciones de datos, ataques DoS, accesos no autorizados). Análisis de casos de estudio y ejemplos de vulnerabilidades reportadas. Considerando el enfoque en ejemplos específicos de seguridad y riesgos de OData, este tiempo es conveniente para identificar vulnerabilidades críticas y definir los requisitos de seguridad.

Fase de diseño de la arquitectura de seguridad y servicios (4 semanas): Diseño de la arquitectura de *software*, definición de políticas de seguridad (autenticación multifactor, control de acceso basado en roles), documentación

del plan de infraestructura de *hardware* y *software*. Se requiere tiempo suficiente para diseñar una arquitectura robusta y documentarla correctamente.

Fase de desarrollo del prototipo (6 semanas): Implementación de servicios RESTful con OData en ASP.NET Core, configuración de Azure API Management, Azure Active Directory, bases de datos, y herramientas de monitoreo. Desarrollo de ejemplos de código y configuraciones para autenticación, autorización y monitoreo. Se estima suficiente tiempo para construir y configurar todos los componentes necesarios del prototipo, ya que esto implica el desarrollo de múltiples servicios y configuraciones en la nube de Azure.

Fase de pruebas de seguridad (4 semanas): Realización de pruebas de seguridad automatizadas (simulación de accesos y verificación de mecanismos de seguridad). Evaluación de autenticación multifactor y control de acceso basado en roles. Tiempo suficiente para realizar pruebas exhaustivas, simular ataques y evaluar la arquitectura de seguridad.

Fase de pruebas de calidad y rendimiento (2 semanas): Ejecución de pruebas de carga, estrés, y rendimiento. Evaluación del comportamiento del sistema (disponibilidad, tiempo de respuesta, uso de recursos). Esta fase es crítica para asegurar que el sistema funcione bajo diferentes condiciones, por lo que se necesita tiempo suficiente para llevar a cabo pruebas completas.

Fase de análisis de resultados y redacción de informe final (4 semanas): Análisis de los resultados obtenidos de las pruebas de seguridad y rendimiento. Documentación de observaciones, conclusiones y

recomendaciones, analizar los datos recopilados, redactar el informe final y revisar las conclusiones antes de la entrega final.

Duración Total Estimada: 24 semanas
Tiempo disponible: 26 semanas (6 meses).

Por la anterior se considera factible dentro del plazo. La duración total estimada de todas las fases del proyecto es de 23 semanas, lo que deja un margen de 2 semanas dentro del plazo de 6 meses. Las 2 semanas de margen pueden utilizarse para manejar cualquier retraso inesperado, ajustes de proyecto, o iteraciones adicionales en pruebas o desarrollo.

10.2. Factibilidad técnica

Se evalúan los recursos, el conocimiento y las herramientas para desarrollar y llevar a cabo el estudio. Este análisis permite identificar posibles limitaciones y necesidades técnicas para asegurar la viabilidad de la investigación.

10.2.1. Recursos Humanos

Ingeniero de *software* (1 recurso): El proyecto requiere un ingeniero de *software* con experiencia en desarrollo de APIs, seguridad de aplicaciones, manejo de Azure, OData, ASP.NET Core, y herramientas de monitoreo y pruebas. Deberá realizar todas las actividades técnicas, desde la revisión documental hasta la implementación del prototipo y pruebas de seguridad y rendimiento.

10.2.2. Recursos Tecnológicos

Equipo de cómputo: Computadora con capacidad adecuada para ejecutar herramientas de desarrollo como Visual Studio, herramientas de pruebas automatizadas (Selenium, JMeter), y herramientas de monitoreo. Se asume que el ingeniero de *software* ya cuenta con este equipo, por lo que no hay costos adicionales.

Infraestructura en la nube: El uso de servicios en la nube de Azure elimina la necesidad de infraestructura física adicional (servidores, redes, almacenamiento). Se utilizarán los siguientes servicios de Azure:

- Azure API Management (Developer Tier): Para la gestión de APIs y autenticación.
- Azure Active Directory (AAD): Para autenticación multifactor y control de acceso.
- Azure App Service (B1, Basic Tier): Para desplegar los servicios RESTful con OData.
- Azure SQL Database (Basic Tier): Base de datos para almacenar información.
- Azure Monitor y Application Insights: Para monitoreo y pruebas de rendimiento.
- Azure DevOps Services: Para integración continua (CI/CD) y pruebas automatizadas.

10.2.3. Acceso a información y permisos

Se necesita acceso a documentación técnica, publicaciones científicas, y normativas de seguridad relevantes, que se pueden obtener a través de

bibliotecas digitales, plataformas académicas y fuentes en línea. No se requieren permisos especiales para implementar el proyecto en un entorno de desarrollo.

El proyecto es técnicamente viable, ya que cuenta con un recurso humano calificado y con acceso a las tecnologías necesarias (Azure) y herramientas de desarrollo. No se requieren permisos especiales ni inversiones en infraestructura física adicional.

10.3. Factibilidad financiera

Para la factibilidad financiera de la investigación, se consideran los costos asociados al recurso humano, los servicios en la nube (Azure) y las herramientas necesarias para la implementación.

En este caso, se propone aprovechar los servicios gratuitos de Azure y el crédito inicial disponible para minimizar los costos durante la fase de desarrollo y pruebas. El proyecto puede ser autofinanciado con fondos propios, cubriendo tanto los costos del ingeniero de *software* (maestrando) como los servicios de Azure de ser necesario. Azure ofrece un plan gratuito que incluye varios servicios con límites de uso mensuales, además de un crédito inicial de USD 200 disponible durante los primeros 30 días. Servicios gratuitos de Azure que se utilizarán:

- Azure API Management (Nivel *Developer*): 1 millón de llamadas de API por mes.
- Azure Active Directory (Azure AD): Gestión de identidades y acceso con el plan gratuito.

- Azure SQL Database (Nivel gratuito): 250 GB de almacenamiento de base de datos SQL durante los primeros 12 meses.
- Azure Monitor (incluyendo Application Insights): 5 GB de datos de *logs* por mes durante 31 días y 1 millón de verificaciones de métricas por mes.
- Azure DevOps: Herramientas de CI/CD gratuitas para equipos de hasta 5 usuarios con 1,800 minutos de ejecución de pipelines por mes.

Durante los primeros 30 días (4 de las 6 semanas estimadas para la elaboración del prototipo), el proyecto hará uso del crédito de USD 200 proporcionado por Azure. Este crédito se podrá utilizar para cubrir cualquier exceso de uso de servicios gratuitos. Los Servicios con probabilidad de exceder el límite gratuito son:

- Azure API Management: Si el proyecto requiere más de 1 millón de llamadas de API por mes.
- Azure Monitor (incluyendo Application Insights): Si se exceden los 5 GB de datos de logs o las verificaciones de métricas.

Después del primer mes, se optimizará el uso de los servicios de Azure para permanecer dentro de los límites gratuitos tanto como sea posible. En caso de que sea necesario un uso adicional, se gestionarán los recursos para minimizar los costos. Se espera concluir el prototipo, las pruebas de seguridad, calidad y rendimiento en 8 semanas, después del mes gratuito. Los costos estimados para los siguientes 2 meses:

- Azure API Management (Nivel Developer): USD $0.60 por 10,000 llamadas adicionales de API.
- Azure Monitor: USD $2.76 por GB adicional de logs.

Dependiendo del uso y las necesidades, se estima un costo adicional de USD $20 por mes para servicios que puedan exceder los límites gratuitos después del primer mes. Utilizando el plan gratuito de Azure y el crédito de USD $200, los costos de servicios en la nube se mantienen al mínimo, lo que mejora la viabilidad financiera del proyecto.

Aunque el maestrando será quien desempeñe el rol de ingeniero de *software*, es importante considerar los costos asociados a este recurso humano para reflejar la inversión de tiempo y esfuerzo en el proyecto. Estos costos, aunque no sean pagados realmente, representan el valor del trabajo realizado.

El ingeniero de *software* dedicará 10 horas a la semana al proyecto y se calculan los costos tomando en cuenta un período de 6 meses (aproximadamente 26 semanas).

Total de horas por semana: 10 horas
Total de semanas: 26 semanas
Total de horas de trabajo: 10 horas/semana × 26 semanas = 260 horas

Tomando un costo por hora de USD $20 para un ingeniero de *software*, el costo total por el recurso humano es: 260 horas × USD $20/hora = USD $5,200.

Tabla 3.

Factibilidad financiera del estudio

Descripción	Cantidad	Costo Unitario	Costo Total
Recurso humano (Ingeniero de *software*)			
Horas semanales	10 horas	$20 / hora	$200 / semana
Semanas de trabajo (6 meses)	26 semanas		
Costo total por recurso humano			$5,200
Servicios de Azure			
Créditos gratuitos de Azure (primeros 30 días)	1 mes	Gratis	
Créditos adicionales para uso extendido (8 semanas)	2 meses	$20 / mes	$ 40
Costo total por servicios de Azure			$ 40
Total general			$5,240

Nota: Descripción y costos del estudio, expresados en dólares estadounidenses. Elaboración propia, realizado con Word.

En la Tabla 3, se presentan los costos estimados para la factibilidad financiera del proyecto, teniendo en cuenta el recurso humano (maestrando), el uso de servicios gratuitos de Azure y el crédito de $40 para cubrir los servicios que excedan el uso gratuito

En conclusión, la factibilidad financiera del proyecto es realizable ya que se maximiza el uso de los recursos gratuitos disponibles en Azure, se cuenta con la participación directa del maestrando como recurso humano, y no se anticipan gastos significativos fuera de los recursos ya disponibles o cubiertos por el crédito inicial de Azure.

REFERENCIAS

Agnelo, J. A. N. (2020). *A robustness testing approach for RESTful Web services* [Tesis de Maestría, Universidad de Coímbra]. Archivo digital. https://estudogeral.uc.pt/bitstream/10316/92292/1/MSc%20Thesis%20-%20Joa%cc%83o%20Agnelo.pdf

Arcuri, A. (2017). RESTful API automated test case generation. *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS),* 9-20. https://ieeexplore.ieee.org/document/8009904/

Atlidakis, V., Godefroid, P., & Polishchuk, M. (October). Checking security properties of cloud service REST APIs. *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST),* 387-397. https://ieeexplore.ieee.org/document/9159084

Cupek, R., & Huczala, L. (2015). OData for service-oriented business applications: Comparative analysis of communication technologies for flexible Service-Oriented IT architectures. *2015 IEEE International Conference on Industrial Technology (ICIT),* 1538-1543. https://ieeexplore.ieee.org/document/7125315

Ed-Douibi, H., Izquierdo, J. L. C., & Cabot, J. (2018). Model-driven development of OData services: An application to relational databases. *2018 12th International Conference on Research Challenges in Information Science (RCIS)*, 1-12. https://ieeexplore.ieee.org/document/8406667

Indu, I., & Anand, P. R. (2015). Identity and access management for cloud web services. *2015 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*, 406-410. https://ieeexplore.ieee.org/document/7488450

Laranjeiro, N., Agnelo, J., & Bernardino, J. (2021). A black box tool for robustness testing of REST services. *IEEE Access 9*, 24738-24754. https://ieeexplore.ieee.org/document/9344640

Li, L., Chou, W., Zhou, W., & Luo, M. (2016). Design patterns and extensibility of REST API for networking applications. *IEEE Transactions on Network and Service Management 13*(1), 154-167. https://ieeexplore.ieee.org/document/7378522

Li, L., & Chou, W. (2015). Designing large scale REST APIs based on REST chart. *2015 IEEE International Conference on Web Services*, 631-638. https://ieeexplore.ieee.org/document/7195624

Papatheodoulou, N., & Sklavos, N. (2009). Architecture & system design of Authentication, Authorization, & Accounting services. *IEEE EUROCON 2009*, 1831-1837. https://ieeexplore.ieee.org/document/5167894

Sendor, J., Lehmann, Y., Serme, G., & de Oliveira, A. S. (2014). Platform-level support for authorization in cloud services with OAuth 2. *2014 IEEE International Conference on Cloud Engineering*, 458-465. https://ieeexplore.ieee.org/document/6903511

Viglianisi, E., Dallago, M., & Ceccato, M. (2020). Restestgen: automated black-box testing of restful apis. In *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, 142-152. https://ieeexplore.ieee.org/document/9159077