UNIVERSIDAD DE COSTA RICA
SISTEMA DE ESTUDIOS DE POSGRADO

COMPARACIÓN DE CONTROLES DE SEGURIDAD EN PLATAFORMAS QUE OFRECEN FUNCIONES COMO SERVICIO

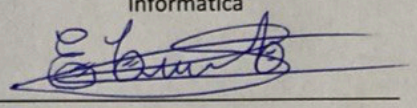
Trabajo final de investigación aplicada sometido a la consideración de la Comisión del Programa de Estudios de Posgrado en Computación e Informática para optar al grado y título de Maestría Profesional en Computación e Informática

DIANA GABRIELA ALVARADO SEQUEIRA

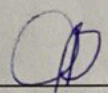
Ciudad Universitaria Rodrigo Facio, Costa Rica

2020

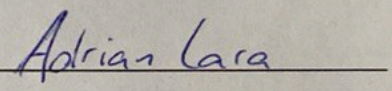
Este trabajo final de investigación aplicada fue aceptado por la Comisión del Programa de Estudios de Posgrado en Computación e Informática de la Universidad de Costa Rica, como requisito parcial para optar al grado y título de Maestría Profesional en Computación e Informática



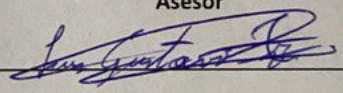
Dr. Edgar Casasola Murillo
Representante del Decano
Sistema Estudios de Posgrado



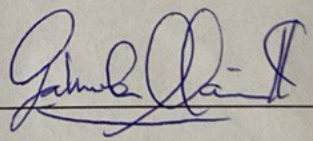
Dra. Elena Gabriela Barrantes Sliesarieva
Profesora Guía



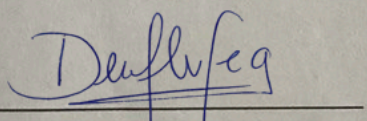
Dr. Adrián Lara Petitdemange
Asesor



Dr. Luis Gustavo Esquivel Quirós
Asesor



Dra. Gabriela Marín Raventós
Directora del Programa de
Posgrado en Computación e Informática



Diana Gabriela Alvarado Sequeira
Sustentante

ÍNDICE

Índice de Figuras

Índice de Tablas

RESUMEN

En este trabajo se comparan los controles de seguridad disponibles de dos de los proveedores de servicio en la nube para funciones como servicio. Con la utilización de guías de seguridad informática para funciones como servicio de organizaciones reconocidas, se analizaron y seleccionaron cinco de los riesgos de seguridad más importantes y las opciones de mitigación disponibles en dos de los proveedores existentes. Se recolectó información sobre la funcionalidad y disponibilidad de controles apropiados, con el fin de compararlos. La comparación se realizó de acuerdo a los lineamientos de seguridad publicados y lo que se considera como un control adecuado para cada riesgo en ellos.

Para llevar a cabo la comparación, se utilizaron las consolas, herramientas y documentación existentes de cada uno de los proveedores seleccionados. Para cada riesgo escogido se implementó un control de mitigación y se evaluó de acuerdo a los lineamientos proporcionados en los estándares de seguridad, comparando su efectividad antes y de después de la implementación del control, así como comparar diferencias en la implementación en cada proveedor que afecten el cumplimiento de las coberturas del riesgo.

PALABRAS CLAVE
Seguridad, Nube, Función como servicio

ABSTRACT

This project compares available security controls for function as a service from two different cloud services providers. Using existing guides and recommendations from recognized security organizations, it was possible to select five of the main security threats found for this type of service as well as its available mitigation options in the two providers selected. The information about the available controls and the services within the providers that offered them, was collected doing research on their online public documentation. The comparison between the controls as well as the providers was done considering appropriate security controls based on criteria from available security guidelines.

The tools used to compare the providers selected were their web consoles, programming tools, and public services documentation. A mitigation, in the form of a control, was implemented for each of the threats selected, its effect was then evaluated based on the criteria provided by the security guidelines selected, comparing its effectiveness in the application behavior before and after its implementation. The differences in the implementation for each of the providers selected was also covered in the analysis and results presented.

KEYWORDS
Security, Cloud, Function as a Service, serverless

UNIVERSIDAD DE COSTA RICA

SEP Sistema de Estudios de Posgrado

Autorización para digitalización y comunicación pública de Trabajos Finales de Graduación del Sistema de Estudios de Posgrado en el Repositorio Institucional de la Universidad de Costa Rica.

Yo, Diana Gabriela Alvarado Sequeira, con cédula de identidad 1-1151-0453, en mi condición de autor del TFG titulado COMPARACIÓN DE CONTROLES DE SEGURIDAD EN PLATAFORMAS QUE OFRECEN FUNCIONES COMO SERVICIO

Autorizo a la Universidad de Costa Rica para digitalizar y hacer divulgación pública de forma gratuita de dicho TFG a través del Repositorio Institucional u otro medio electrónico, para ser puesto a disposición del público según lo que establezca el Sistema de Estudios de Posgrado. **SI** [x] **NO** *[ ]

***En caso de la negativa favor indicar el tiempo de restricción: _____ año (s).**

Este Trabajo Final de Graduación será publicado en formato PDF, o en el formato que en el momento se establezca, de tal forma que el acceso al mismo sea libre, con el fin de permitir la consulta e impresión, pero no su modificación.

Manifiesto que mi Trabajo Final de Graduación fue debidamente subido al sistema digital Kerwá y su contenido corresponde al documento original que sirvió para la obtención de mi título, y que su información no infringe ni violenta ningún derecho a terceros. El TFG además cuenta con el visto bueno de mi Director (a) de Tesis o Tutor (a) y cumplió con lo establecido en la revisión del Formato por parte del Sistema de Estudios de Posgrado.

INFORMACIÓN DEL ESTUDIANTE:

Nombre Completo: Diana Gabriela Alvarado Sequeira.

Número de Carné: A00203 Número de cédula: 1-1151-0453.

Correo Electrónico: dianaalse@gmail.com.

Fecha: 16/04/2021. Número de teléfono: 2273-32-96.

Nombre del Director (a) de Tesis o Tutor (a): Dra. Elena Gabriela Barrantes Sliesarieva.

FIRMA ESTUDIANTE

Nota: El presente documento constituye una declaración jurada, cuyos alcances aseguran a la Universidad, que su contenido sea tomado como cierto. Su importancia radica en que permite abreviar procedimientos administrativos, y al mismo tiempo genera una responsabilidad legal para que quien declare contrario a la verdad de lo que manifiesta, puede como consecuencia, enfrentar un proceso penal por delito de perjurio, tipificado en el artículo 318 de nuestro Código Penal. Lo anterior implica que el estudiante se vea forzado a realizar su mayor esfuerzo para que no sólo incluya información veraz en la Licencia de Publicación, sino que también realice diligentemente la gestión de subir el documento correcto en la plataforma digital Kerwá.

CAPÍTULO I: INTRODUCCIÓN

El uso de Internet, que inició en la década de 1990, ha cambiado el mundo de la informática de una manera drástica. Se ha transformado el concepto de computación paralela, la computación distribuida, y, recientemente, la computación en la nube (Karabek et al, 2011). Aunque la idea de la computación en la nube existe desde hace bastante tiempo, es todavía un campo emergente de tecnología. La forma más básica de definir lo que es la 'Nube' es que es una computadora ubicada en otro lugar que se accede a través de Internet y se utiliza de alguna manera. La computación en la nube establece el consumo bajo demanda de poder de computación, almacenamiento, base de datos, aplicaciones y cualquier recurso de infraestructura tecnológica siguiendo el modelo de pago por uso. Su principal ventaja es que los clientes no tienen que pagar por la infraestructura, su instalación, la mano de obra requerida para manejar dicha infraestructura y mantenimiento (Dhar, 2012).

El modelo de consumo de recursos basado en servidores de la computación en la nube ha resultado ser ineficiente (Madni et al, 2016). La utilización de servidores completos individualmente no permite escalar el servicio para poder cubrir el crecimiento en la demanda, lo que ha generado una evolución constante en el modelo de aprovisionamiento de los recursos, lo que propició el surgimiento del paradigma de Función como Servicio. El concepto de Función como Servicio o FaaS, del inglés *Function as a Service* describe un modelo de programación y arquitectura en la nube donde una aplicación se descompone en "disparadores" (eventos) y "acciones" (funciones). Hay una plataforma que proporciona un alojamiento y entorno de ejecución, sin responsabilidades operativas, cobrando solo por el tiempo que tarda en completar la ejecución. Esto permite utilizar la capacidad de cada servidor de manera mucho más eficiente y constituye un ahorro para el usuario (Baldini et al, 2017).

La Función como Servicio tiene como una de sus características más atractivas delegar la mayor parte de las responsabilidades operativas al proveedor de servicios en la nube, esta

representa una oportunidad y un riesgo para los usuarios. Como oportunidad, permite reducir costos y simplificar el proceso de implementar pequeñas cantidades de código rápidamente. Como riesgo, dificulta comprender y controlar los accesos y controles necesarios para la ejecución segura del código (Eivy, 2017).

El modelo de precios y su facilidad de uso, han hecho de la Función como Servicio uno de los servicios más innovadores y populares en los proveedores de servicios en la nube (Eivy & Weinman, 2017).La evolución de los paradigmas de computación en la nube y en especial de la utilización de Funciones como Servicio de forma creciente hacen que sea clave la evaluación de las consideraciones de seguridad informática más importantes, y conocer los controles ofrecidos por los principales proveedores para contrarrestar los posibles riesgos que se presentan al utilizarlo, que será el foco de la presente investigación.

En el presente trabajo se busca evaluar controles de seguridad para funciones como servicio en plataformas de servicios en la nube para un grupo de amenazas conocidas, de esta manera se desea proporcionar una guía que sirva de ayuda a usuarios con conocimiento limitado del tema.

1.1 Justificación

La era de Internet y la demanda de recursos computacionales que esta ha ocasionado, ha creado una revolución tecnológica liderada por las empresas proveedoras de servicios en la nube. El crecimiento de la utilización de Funciones como Servicio y de código fácilmente ejecutable hace crítica la evaluación de la seguridad informática asociada con su uso (Alpernas et al, 2018).

El usuario promedio de servicios en la nube carece del conocimiento especializado de seguridad para considerar opciones de configuración requeridas para cubrir riesgos básicos que afectan a una gran parte de las aplicaciones que se encuentran en internet. Las aplicaciones sin controles de seguridad pueden ser aprovechadas por actores

maliciosos para el robo de información confidencial u obtener privilegios de acceso a infraestructura privada (Basu et al, 2018).

La importancia del desarrollo e implementación de este trabajo es orientar a los usuarios y empresas que utilizan o consideran utilizar Funciones como Servicio. Se abarcan las etapas que se deben considerar en estas implementaciones: desde la selección e implementación de una aplicación web común utilizando servicios de la nube, hasta los riesgos a considerar y los controles existentes en estas plataformas para su mitigación y cobertura, en al menos algunos de los proveedores más importantes.

1.2 Planteamiento del problema

La función como servicio es un paradigma de computación que se refiere a la ejecución de porciones de código en respuesta a eventos. Este código tiene la capacidad de actuar aisladamente y ser independiente del resto del código de una aplicación o sitio web, pero es vulnerables a riesgos de seguridad informática similares a las aplicaciones tradicionales, así como nuevos riesgos introducidos por la naturaleza de su funcionamiento (Jonas et al, 2019).

La función como servicio puede ejecutar múltiples acciones dentro de la infraestructura de una aplicación, puede acceder o modificar información crítica, generar alertas o cambios de configuración y mucho más, por lo que son blanco de ataques cibernéticos constantes. Los métodos utilizados por los usuarios son en su mayoría un reflejo de las practicas que se han desarrollado a través del tiempo para las aplicaciones tradicionales sin considerar la capacidad única que los proveedores de servicio pueden ofrecer para mitigar riesgos que afectan específicamente a este servicio (Pérez et al, 2018).

La falta de conocimiento y experiencia en la implementación de Funciones como Servicio juega un papel fundamental en que usuarios y empresas desconozcan la mejor manera

de evaluar y abordar los riesgos a los que esta expuesto el servicio e implementen los controles adecuados para ellos. Proporcionar de forma resumida y concisa los controles de la nube que se pueden utilizar para los riesgos de seguridad más importantes, de una manera práctica en una aplicación de referencia, son la principal meta de este trabajo. Los retos que conlleva implementar una aplicación de este tipo con múltiples componentes expuestos y la delegación de responsabilidades son bien conocidos (Hong et al, 2018). La implementación de la aplicación de referencia servirá para evaluar el efecto de los controles de seguridad sobre la funcionalidad de la misma además de la confirmación de la mitigación de la amenaza.

En la investigación *Serverless Computing Security: Protecting Application Logic* (O'Meara & Lennon, 2020) se refieren a la naturaleza impulsada por eventos y microservicios que caracterizan a la arquitectura de FaaS y como puede aumentar la superficie de ataque de una aplicación y hacerla vulnerable a ataques maliciosos. Parte de la problemática de la utilización del servicio es que no tiene supervisión o controles pre determinados que alerten sobre estos riesgos y es competencia de los usuarios comprenderlos y tomar las medidas adecuadas para asegurar sus aplicaciones. La educación es una de las armas más importantes para poder mejorar la capacidad de los usuarios de implementar mejores prácticas de seguridad informática. En resumen, el problema que motivó el desarrollo de esta investigación es la falta de una guía completa que caracterice las amenazas principales en las aplicaciones más comunes y al mismo tiempo dé una guía de como se pueden cubrir con controles o servicios ya disponibles en los proveedores.

1.3 Antecedentes

En la investigación "*Leveraging the serverless architecture for securing linux containers*" (Bila et al, 2017). se tratan dos temas muy específicos como son Linux y contenedores desde una perspectiva *FaaS*, se estudia como se puede utilizar la arquitectura *serverless* para implementar automatización en la mitigación de amenazas de los contenedores, utilizando tecnología como OpenWhisk y Kubernetes se establecen políticas que automáticamente mitigan los riesgos encontrados en los contenedores. Este trabajo ahonda en seguridad y *serverless*, pero no en los servicios específicos ofrecidos por las plataformas de servicio.

En el estudio "*Building a chatbot with serverless computing*" (Yan, Castro, Cheng & Ishakian, 2016) el enfoque es hacia la utilización de arquitectura *serverless* para la implementación de un chatbot en la nube de IBM, llamada Watson Developer Cloud. Con la implementación de esta arquitectura se logró hacer el modelo mas extensible, escalable y flexible. Se ahonda en las posibilidades de la utilización de este tipo de arquitectura para una aplicación especifica, pero no se menciona el tema de seguridad.

En la investigación "*Deviceless edge computing: extending serverless computing to the edge of the network*" (Glikson, Nastic & Dustdar, 2017) se examina el paradigma *serverless* para extenderlo al borde de internet e integrar aparatos de IoT y de borde a la ejecución de aplicaciones. El resultado es un prototipo al que llaman LEON, el que utiliza tecnologías como OpenWhisk, Docker y Node-RED que son software libre, en este modelo se ejemplifican las ventajas de *serverless* en implementaciones locales, no se exploran temas de implementación del prototipo en la nube o el tema de controles de seguridad.

En el trabajo "*A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-a-Service) Platforms*" (Lynn, Rosati, Lejeune & Emeakaroha, (2017) se examinan los principales proveedores de *serverless* en la computación de la nube. En este se compara cada uno de los proveedores a alto nivel y sus casos de uso, se citan siete proveedores específicos: AWS Lambda, Google Cloud Functions, Microsoft Azure Functions, IBM Bluemix OpenWhisk, Iron.io Ironworker, Auth0 Webtask, and Galactic Fog Gestal Laser.

Se propone una matriz de los servicios disponibles en cada proveedor y cómo se pueden utilizar, esta se refiere a las funcionalidades disponibles en el ano de publicación, lo cual lo hace obsoleto rápidamente ya que los proveedores publican nueva funcionalidad frecuentemente, tampoco se aborda el tema de seguridad y las opciones que cada plataforma ofrece.

1.4 Objetivos

Para el desarrollo de la presente investigación se quiso seleccionar al menos dos proveedores de servicios en la nube para analizar y comparar sus controles de seguridad para Funciones como Servicio. Como experimento, se implementó una aplicación en cada uno junto con los controles identificados con el objetivo de compararlos. Los objetivos propuestos se presentan en las siguientes dos secciones.

1.4.1 Objetivo General

Evaluar los controles de seguridad disponibles en plataformas que ofrecen funciones como servicio, para un grupo de amenazas bien conocidas.

1.4.2 Objetivos Específicos

1. Seleccionar al menos cinco amenazas y sus controles de seguridad más importantes, para funciones como servicio, con base en estándares de las organizaciones OWASP y CSA.
2. Implementar una aplicación que utilice funciones como servicio que permitan validar los controles de seguridad seleccionados, en al menos dos proveedores diferentes de servicios en la nube, que provean servicios equivalentes.
3. Comparar y evaluar los controles de seguridad seleccionados, en la aplicación implementada en cada proveedor de servicios en la nube.

1.5 Aporte

El presente trabajo aporta conocimiento sobre las opciones de seguridad disponibles para utilizar Funciones como Servicio en los proveedores de servicio en la nube. Primero, muestra los riesgos de seguridad más importantes a las que están expuestas y luego propone controles proporcionados por los proveedores de servicios en la nube para su prevención y mitigación.

El tema de la utilización de los servicios en la nube y FaaS es un tema actual, y los riesgos de seguridad son cada vez más importantes y visibles para desarrolladores y usuarios, es un aporte muy importante demostrar los riesgos que existen y como pueden ser prevenidos. Las recomendaciones para la implementación segura de aplicaciones que utilizan este servicio es uno los elementos de gran importancia que son poco considerados por usuarios.

De forma específica, se realizó una implementación en dos de los proveedores más importantes de servicios en la nube y se investigaron e implementaron estos mecanismos de prevención y mitigación. La investigación demuestra como las amenazas son inherentes a la utilización del servicio y depende del usuario comprender y utilizar los controles adecuados.

En este trabajo, el capítulo 2 enmarca y delimita la investigación, define conceptos básicos involucrados, así como los antecedentes de la investigación. El capítulo 3 presenta la metodología utilizada en la implementación y desarrollo de la investigación. El capítulo 4 muestra los resultados obtenidos y el análisis respectivo. El capítulo 5 presenta las conclusiones y propone un trabajo a futuro, como propuesta para continuar o ampliar el tema de esta investigación.

CAPÍTULO II: MARCO TEÓRICO

En este capítulo se describe la base teórica que fundamentó el desarrollo del presente trabajo. Se definen los principales conceptos que explican las funciones como servicio y su seguridad. En primer lugar, se presentan los principales proveedores de servicios en la nube y sus ofertas de servicios, estos proveen el fundamento de la función como servicio; asimismo, la definición de lo que ofrece cada uno como función como servicio. También se explican los estándares de seguridad para esta arquitectura, cuáles existen y cuál es su relevancia. Se detallan asimismo los principales controles a considerar en implementaciones de este tipo. Finalmente, se describe el fundamento teórico respecto a la arquitectura implementada para la valoración de los controles existentes.

2.1 La computación en la nube y sus modelos

La computación en la nube es un modelo que permite el acceso a internet de manera ubicua, conveniente y bajo demanda, contiene un conjunto de recursos informáticos configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que se pueden aprovisionar y liberar rápidamente con un mínimo esfuerzo o interacción con el proveedor de servicios. (Mell, P & Grance, 2011).

Este modelo de nube se compone de tres categorías principales: Software como servicio (SaaS), Plataforma como servicio (PaaS) e Infraestructura como servicio (IaaS), los cuales se detallan a continuación.

2.1.1 Software como servicio (SaaS)

La capacidad que se brinda al consumidor es utilizar las aplicaciones del proveedor, pero con infraestructura en la nube. Las aplicaciones son accesibles por medio de dispositivos cliente como un navegador web o un programa. El consumidor no gestiona ni controla la infraestructura subyacente que incluye red, servidores, sistemas operativos,

almacenamiento o incluso capacidades de aplicaciones individuales, con la posible excepción de ajustes de configuración de aplicación específicos de usuario limitados (Garg, Versteeg & Buyya, 2010).

2.1.2 Plataforma como servicio (PaaS)

La capacidad proporcionada al consumidor es implementar en la nube sus aplicaciones construidas con lenguajes de programación, bibliotecas, servicios y herramientas compatibles con el proveedor. El consumidor no puede administrar ni controlar la infraestructura de nube subyacente, incluida la red, los servidores, sistemas operativos o almacenamiento, pero tiene control sobre las aplicaciones implementadas y posiblemente ajustes de configuración para el entorno de alojamiento de las aplicaciones (Li, Yang, Kandula & Zhang, 2010).

2.1.3 Infraestructura como servicio (IaaS)

La capacidad proporcionada al consumidor es la proveer procesamiento, almacenamiento, redes y otros recursos informáticos fundamentales donde el consumidor puede implementar y ejecutar software arbitrario, que puede incluir operaciones sistemas y aplicaciones. El consumidor no gestiona ni controla la nube subyacente de infraestructura, pero tiene control sobre los sistemas operativos, el almacenamiento y las aplicaciones implementadas y posiblemente un control limitado de componentes de red seleccionados (por ejemplo, firewalls de host) (Prodan & Ostermann, 2009).

En la figura 1 se detalla de manera visual el enfoque incremental de las responsabilidades del usuario según el servicio que utilice en la nube.

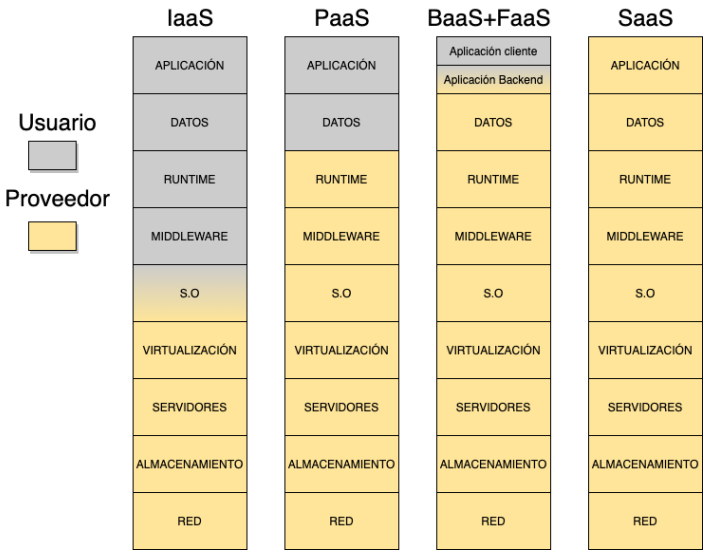


Figura 1: Representación gráfica de los servicios en la nube

En la figura 1, se nota como al tener menos control de los recursos, se tiene también menos responsabilidad de mantenimiento de la infraestructura y la tecnología.

2.2 Las opciones de función como servicio y sus principales proveedores

La función como servicio es un modelo de ejecución en el que el proveedor de la nube (AWS, Azure o Google Cloud) es responsable de ejecutar un fragmento de código asignando dinámicamente los recursos. Y solo cobra por la cantidad de recursos utilizados para ejecutar el código. El código generalmente se ejecuta dentro de contenedores sin estado que pueden ser activados por una variedad de eventos que incluyen solicitudes http, eventos de bases de datos, servicios de colas, alertas de monitoreo, carga de archivos, eventos programados y otros (Lynn, Rosati, Lejeune & Emeakaroha, 2017).

La función como servicio abstrae la infraestructura subyacente del desarrollador, pero los servidores aún están involucrados en la ejecución de las funciones. Algunos de los usos comunes de FaaS son (Lee, Satyam & Fox, 2018) :

- Procesamiento de multimedia: la implementación de funciones que ejecutan un proceso de transformación en respuesta a la carga de un archivo

- Cambios en la base de datos o captura de datos modificados (ETL): auditar o garantizar que los cambios cumplan con los estándares de calidad
- Mensajes de entrada en sensores IoT: la capacidad de responder a mensajes e incrementar el numero de respuestas requeridas
- Procesamiento de flujo a escala: procesamiento de datos dentro de un flujo de mensajes potencialmente infinito
- Chatbots: incremento automatizado para altas demandas
- Tareas programadas de trabajos por lotes: trabajos que requieren computación paralela intensa, IO o acceso a la red
- API REST HTTP y aplicaciones web: cargas de trabajo de solicitud y respuesta tradicionales

Los proveedores de nube ofrecen distintas funcionalidades en su producto, también cada uno se enfoca en algunos casos de uso específicos. En (Lynn, Rosati, Lejeune & Emeakaroha,2017) se provee un resumen de las principales funcionalidades ofrecidas en 7 categorías definidas por los autores, hay que tomar en cuenta que este estudio es del año 2016 y los proveedores mejoran y crean nuevas funcionalidades constantemente, esta referencia sirve como base para comprender las posibilidades existentes en cada plataforma.

Las ofertas de FaaS de los principales proveedores de la nube se describen en las secciones 2.2.1, 2.2.2 y 2.2.3.

2.2.1 Amazon Web Services (AWS): AWS Lambda

Presentado inicialmente en el 2014, Lambda es un servicio en la nube de AWS que puede integrarse con otras funcionalidades de su plataforma. Los eventos desencadenantes que se pueden utilizar incluyen cargas de imágenes, actividades in-app, clics en sitios web, información de dispositivos conectados u otras peticiones personalizadas. Desde su inicio,

Amazon Web Services destacó que AWS Lambda eliminaría la necesidad de aprovisionar o administrar servidores virtuales y auto escalado en sus múltiples zonas de disponibilidad. Entre los casos de usos mencionados se incluye el procesamiento de datos (Procesamiento de archivos en tiempo real, procesamiento de flujo en tiempo real y Extraer Transformar y Cargar - ETL) y back-ends sin servidor (IoT, móvil y web) (AWS Lambda, 2019).

Un ejemplo de un servicio con funciones como servicio se muestra en la figura 2. Este ejemplo es una representación de como uno de los usos comunes (API REST) se puede implementar de manera sencilla utilizando este paradigma, al interponer una función entre la solicitud y la respuesta. De esta manera, se logra abstraer la infraestructura y disminuir los costos de operación y mantenimiento.

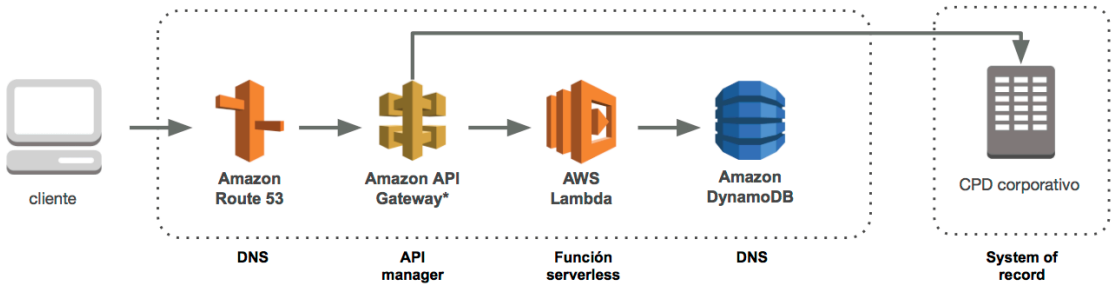


Figura 2: Representación de una función como servicio en Amazon

2.2.2 Google Cloud: Cloud Functions

Google introduce Cloud Functions en 2016. Diseñado principalmente para los servicios de Google Cloud, Google cita una serie de casos de uso específicos incluyendo aplicaciones móviles, APIs y desarrollo de micro servicios, procesamiento de datos / ETL, webhooks (para responder a terceros desencadenantes), IoT y aplicaciones de inteligencia artificial (Google Cloud, 2019)

2.2.3 Microsoft Azure: Azure Functions

Microsoft Azure Functions se introdujo al mercado en el 2016. Fue diseñado para extender la plataforma de Azure existente. La plataforma ofrece capacidades para implementar código disparado por eventos que ocurren en Azure o en servicios de terceros, así como en sistemas internos de las empresas. Microsoft cita los siguientes casos de uso: procesamiento basado en tiempo (cargas de trabajo Cron), activadores de Azure, evento de software como servicio (SaaS), Procesamiento, procesamiento de flujo en tiempo real (IoT), y mensajes de bot en tiempo real (Microsoft Azure, 2019).

2.3 Seguridad informática y la función como servicio

El uso de Internet y de aplicaciones web se ha convertido en parte fundamental de las actividades comerciales y, muchas de estas aplicaciones son implementadas con amenazas críticas que pueden ser explotadas. Es por esto que es importante recurrir a un punto de referencia reconocido y utilizarlo para evaluar la seguridad de las aplicaciones web, ya sea en la nube o localmente (Hendrickson et al,2016)

Al adoptar la tecnología de función como servicio, se elimina la necesidad de administrar un servidor para nuestra aplicación. Al hacerlo, también pasamos algunas de las amenazas de seguridad al proveedor de infraestructura como AWS, Azure o Google Cloud, algunos otros aspectos de seguridad también se pasan al proveedor y se debe confiar en su aseguramiento adecuado como parte del servicio (Hendrickson et al,2016).

Los proveedores de nube proveen la infraestructura de manera abstracta a los usuarios, pero siempre existe código que se esta ejecutando, si el código está escrito de manera insegura, la aplicación puede ser vulnerable al nivel de aplicación tradicional y a ataques, como Cross-Site Scripting (XSS), Inyección SQL, denegación de servicio (DoS), pérdida de autenticación y autorización y muchos más (OWASP, 2019).

Unos de los puntos de referencia más reconocido para evaluar la seguridad de las aplicaciones es el Open Web Application Security Project (OWASP), es un proyecto de código abierto dedicado a estudiar y combatir amenazas de seguridad en software web. OWASP gestiona varios proyectos de seguridad entre los más importantes OWASP Top 10; el mismo contiene los 10 tipos de ataque más influyentes hacia la seguridad de aplicaciones. El objetivo principal de este proyecto es proveer técnicas básicas de prevención y protección de ataques informáticos a nivel de desarrollo de software. Como parte del proyecto de top 10 recientemente se publicó una guía para función como servicio o *serverless,* la cual lista la relevancia y aplicación del top 10 general dentro de este paradigma, esta servirá de base para evaluar los controles de seguridad que se ofrecen en los proveedores de nube en la actualidad para su implementación (OWASP, 2019).

En la guía de interpretación del Top 10 para función como servicio, OWASP describe cada una de las amenazas listadas y provee ejemplos de utilidad para identificar los riesgos, así como un indicador de impacto específicamente para estas implementaciones.
Con esta guía como ayuda se puede proceder a evaluar aplicaciones que utilizan funciones como servicio en cualquiera de las opciones de implementación existentes.

Existen otras organizaciones que proveen estándares y mejores prácticas para la utilización de servicios en la nube, entre ellas está CSA (*Cloud Security Alliance*) que ofrece además certificaciones basadas en el sistema de certificación abierta (OCF) para calificar la madurez de los controles de seguridad de la empresa.

2.4 Amenazas identificadas por OWASP para funciones como servicio

El listado de las diez amenazas incluidas en el top 10 de OWASP se muestra en la tabla 1. En el apéndice A, se resume la información más relevante de la guía creada, esta se puede accesar de forma completa en el sitio web de OWASP (OWASP TOP 10, 2017).

Tabla 1: Resumen de las amenazas listadas por OWASP

Identificador	Nombre de la amenaza
A1:2017	Inyección
A2:2017	Pérdida de autenticación
A3:2017	Exposición de datos sensibles
A4:2017	Entidades Externas XML(XXE)
A5:2017	Pérdida de control de acceso
A6: 2017	Configuración de seguridad incorrecta
A7:2017	Secuencia de comandos en sitios cruzados (XSS)
A8:2017	Deserialización insegura
A9:2017	Componentes con amenazas conocidas
A10:2017	Registros y monitoreo insuficiente

El propósito de la guía de OWASP no es cubrir a cabalidad todas las áreas de seguridad informática que se deben considerar, pero sí las más importantes. Los proveedores de servicios en la nube pueden tener guías especificas de los aspectos a considerar cuando se utilizan servicios específicos que ofrecen.

CAPÍTULO III: METODOLOGÍA

La primera tarea que se realizó en la investigación fue seleccionar al menos cinco amenazas de seguridad y sus controles más importantes. Se estudió la información publicada por las entidades especializadas en seguridad de aplicaciones web y se familiarizó con los conceptos y controles mencionados y sugeridos.

La selección de las amenazas, que se describe en la sección 3.1, se hizo tomando en cuenta la frecuencia con la que se encuentran junto con los controles sugeridos y la opción de utilizar servicios de la nube. Una vez identificadas las amenazas se procedió a escoger dos de los proveedores de servicios en la nube para su implementación, como se describe en la sección 3.2, la selección se basó en evaluaciones de terceras partes de las capacidades de cada uno de los proveedores existentes en el área en cuestión.

Una vez seleccionados los proveedores se investigó y seleccionó el diseño de una aplicación web común, como se explica en la sección 3.3. Al implementar la misma aplicación en cada proveedor, como se describe en las secciones 3.4 y 3.5, se lograron comparar las opciones ofrecidas por cada uno para prevenir las amenazas estudiadas y si cumplen con lo establecido por las entidades especializadas.

3.1 Selección de las amenazas a considerar

Las amenazas se seleccionaron utilizando la guía publicada por OWASP sobre las amenazas más comunes en aplicaciones que utilizan funciones como servicio según se describió en la sección 2.4 del marco teórico de este documento.

Un resumen de la lista de amenazas se muestra en la tabla 2. Se incluye una evaluación sobre si las recomendaciones de mitigación de la guía incluyen controles nativos de la nube.

Tabla 2:Evaluación de las amenazas de OWASP de acuerdo al tipo de control sugerido

Identificador	Nombre de la amenaza	Incluye un control nativo de la nube
A1:2017	Inyección	No
A2:2017	Pérdida de autenticación	Sí
A3:2017	Exposición de datos sensibles	Sí
A4:2017	Entidades Externas XML(XXE)	No
A5:2017	Pérdida de control de acceso	Sí
A6: 2017	Configuración de seguridad incorrecta	No
A7:2017	Secuencia de comandos en sitios cruzados (XSS)	Sí
A8:2017	Deserialización insegura	No
A9:2017	Componentes con amenazas conocidas	No
A10:2017	Registros y monitoreo insuficiente	Sí

Las amenazas se seleccionaron considerando cuan factible era la opción de implementar el control de prevención con opciones específicas de un proveedor de servicios en la nube. Las cinco amenazas seleccionadas se muestran en la tabla 3. Como se puede notar, las recomendaciones específicas que son nativas de la nube se resaltan en negrita, los otros riesgos listados se podían mitigar utilizando código en la implementación de la aplicación sin necesariamente estar relacionado con controles ofrecidos en los proveedores de servicios de la nube. Las amenazas escogidas se muestran junto con la mención de la recomendación especifica de control ofrecido en la nube.

Tabla 3: Resumen de las amenazas seleccionadas para estudio y sus controles sugeridos

Nombre de la amenaza	Controles sugeridos por OWASP
Pérdida de autenticación	**• Utilice el tipo de identidad y control de acceso adecuado, si es posible, utilice las soluciones disponibles proporcionadas por el proveedor de servicios en la nube** **• Los recursos externos deben requerir autenticación y control de acceso según el servicio** **• Para la autenticación entre recursos internos, use métodos seguros conocidos, como federación de identidad**
Exposición de datos sensibles	• Identificar y clasificar datos confidenciales • Almacenar solo datos confidenciales necesarios • Proteja los datos en reposo y en tránsito • Utilizar solo HTTPS para los APIs **• Utilice los servicios del proveedor de servicios de nube para la gestión de claves y el cifrado de datos almacenados, secretos y variables de entorno**
Pérdida de control de acceso	**• Examine cada función cuidadosamente e intente seguir el principio de privilegio mínimo** • Se recomienda automatizar el proceso de configuración de permisos para funciones • Siga las mejores prácticas de implementación de controles según el proveedor de servicios de la nube
Secuencia de comandos en sitios cruzados (XSS)	• Codificar todos los datos no confiables antes enviarlo al cliente, así como usar marcos y encabezados conocidos **• Utiliza un cortafuegos para aplicaciones web**

Nombre de la amenaza	Controles sugeridos por OWASP
Registros y monitoreo insuficiente	**• Utilice las herramientas de monitoreo proporcionadas por el proveedor de servicios de nube para identificar e informar comportamientos no deseados** • Implemente un mecanismo de auditoría y monitoreo para los datos que el proveedor de servicios de infraestructura no provee completamente

3.2 Selección de los proveedores de servicios de la nube

Para poder escoger al menos dos proveedores de servicio en la nube se utilizó un criterio público y ampliamente reconocido que es publicado por la empresa Forrester (Forrester, 2020), Forrester Wave es una guía para compradores que consideran sus opciones en un mercado tecnológico. Para ofrecer un proceso equitativo para todos los participantes, Forrester sigue una metodología disponible públicamente, que aplica de manera consistente a todos los proveedores participantes. La figura 3 presenta un resumen de la publicación del primer trimestre del 2020, para la evaluación de los proveedores de funciones como servicio, como se puede ver los líderes según la estrategia y la capacidad de ejecutar se muestran hacia la derecha y arriba en el gráfico y en este caso son *Amazon Web Services* (AWS) y *Microsoft Azure Functions* en base a estos criterios e información, estos fueron los proveedores seleccionados para la presente implementación y trabajo.

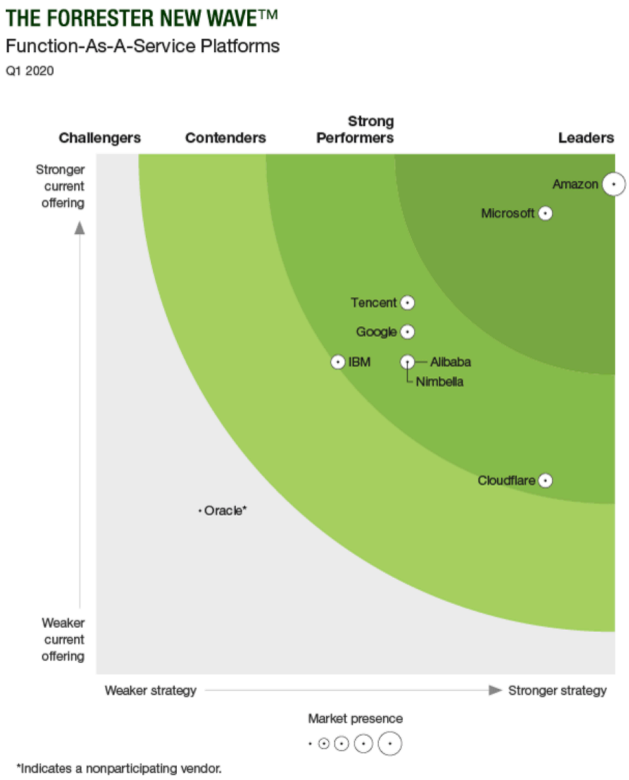


Figura 3: Cuadro de clasificación de proveedores de la nube para funciones como servicio. Tomado de (Forrester, 2020)

3.3 Diseño de la aplicación a implementar

La aplicación se diseñó basada en arquitecturas sugeridas en la literatura y se pudiera mejorar siguiendo las recomendaciones de OWASP. Se utilizó la información y guía recomendada en (Zambrano, 2018), en esta se analiza una aplicación de tres niveles y su descomposición en un patrón utilizando funciones como servicio. En la figura 4, se muestra una arquitectura básica sin componentes específicos de ningún proveedor, como se puede ver, se utiliza una capa de presentación web basada en los componentes comunes como HTML y se puede agregar css y javascript si se desea, una capa web en donde se ejecuta el código de la aplicación, esta se descompone en la API y una función como servicio que se ejecuta a raíz de la solicitud web, finalmente una capa de datos en donde se encuentra la base de datos.

.

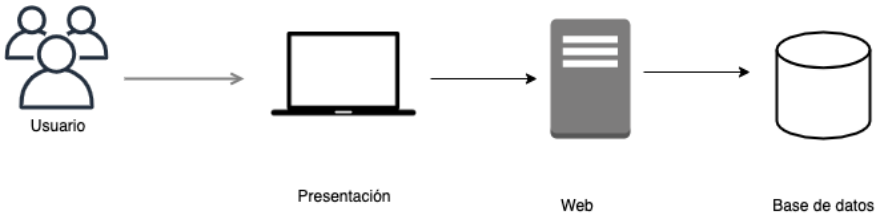


Figura 4: Esquema de componentes de una aplicación de tres niveles

En el caso específico de este trabajo, se implementó una aplicación simple basada en el lenguaje node.js(Node, 2020), esta ofrece la posibilidad de solicitar ítems de una base de datos por medio de solicitudes web creadas como APIs, la estructura de las solicitudes es la siguiente: el código consiste de una función principal, ItemsAPI, lista las opciones disponibles por medio de recursos APIs, en este caso el recurso es /ítems y el verbo HTTP utilizado para recuperar los ítems de la base de datos fue GET. El código utilizado está disponible en el enlace proporcionado en Github(TFIA, 2020)

En las sub secciones 3.3.1 y 3.3.2 a continuación se muestra como los componentes básicos del diseño sugerido se mapean a los servicios ofrecidos de cada proveedor de servicios de la nube.

3.3.1 Amazon Web Services (AWS)

Los componentes básicos de la aplicación se muestran en la figura 5, en ella se especifican los nombres de los servicios que realizan cada una de las funciones básicas requeridas.

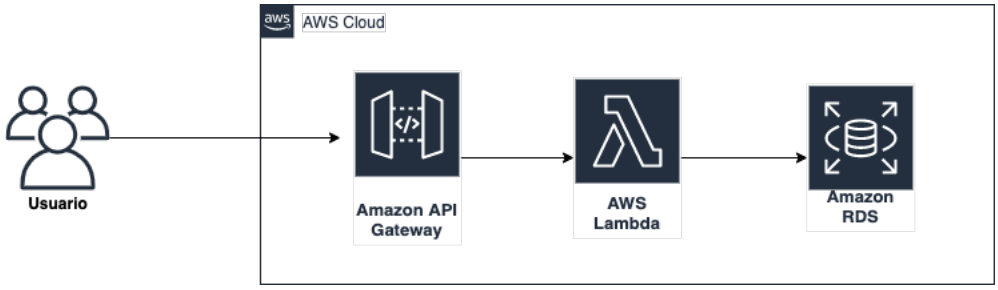


Figura 5: Descripción de la arquitectura básica utilizada en AWS para la implementación de la aplicación creada. Elaboración propia.

El flujo de la solicitud web a través de los componentes y la interacción entre estos se detalla a continuación:

1. Usuario: Inicia la ejecución de la solicitud.
2. Amazon API Gateway: Es el punto de entrada de la solicitud web, en él se registran los recursos de API que se listaron en la sección 3.3. Al ser contactado el recurso, se dispara la ejecución de la función.
3. Amazon Lambda (Función como servicio): Guarda y ejecuta el código principal de cada una de la función, la ejecución inicia por orden de API Gateway.
4. Amazon RDS Aurora (MySQL): Es la base de datos, guarda la información de las solicitudes que se recupera con el verbo HTTP GET.

3.3.2 Microsoft Azure

Los componentes básicos de la aplicación se muestran en la figura 6. En ella se especifican los nombres de los servicios que realizan cada una de las funciones básicas requeridas.

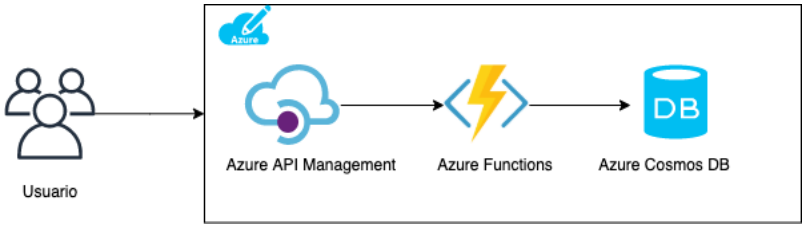


Figura 6: Descripción de la arquitectura básica utilizada en Azure para la implementación de la aplicación creada. Elaboración propia.

El flujo de la solicitud web a través de los componentes y la interacción entre estos se detalla a continuación:

1. Usuario: Inicia la ejecución de la solicitud.
2. Azure API Management: Es el punto de entrada de la solicitud web, en él se registran el recurso de API que se listó en la tabla 4, al ser contactado el recurso se dispara la ejecución de la función.
3. Azure Functions (Función como servicio): Guarda y ejecuta el código principal de cada función. La ejecución inicia por orden de API Management.
4. Azure Cosmos DB: Es la base de datos. Guarda la información de solicitudes que se solicita con el verbo HTTP GET.

Luego de establecer los componentes básicos de la aplicación y el flujo de la solicitud web, se puede proceder a identificar las opciones disponibles en cada proveedor para mitigar las amenazas seleccionadas en la sección 3.1, esto requiere implementar controles de seguridad que los proveedores ofrecen en forma de servicios. A continuación, se identificaron y seleccionaron estos servicios en cada proveedor.

3.4 Identificación de los controles a implementar en cada proveedor

Para poder identificar los controles de los proveedores que servirían para mitigar las amenazas seleccionadas se realizó una investigación en la plataforma de cada proveedor,

cada uno ofrece la opción de crear y utilizar una cuenta de forma gratuita por un período definido y con algunas limitaciones de acceso que no afectaron este trabajo.

La fuente de información principal para identificar y escoger los controles a implementar en AWS fue la documentación pública oficial. En ella se pueden encontrar casos de uso comunes, prácticas recomendadas, como trabajar con otros de los servicios que ofrecen (Documentación Lambda, 2020) e incluso un estudio de la seguridad para la función como servicio (Lambda estudio, 2019). Esto facilitó comprender los servicios que pueden ser utilizados con los componentes básicos ya identificados y tener referencias base de como implementar cada uno.

Los controles de Azure también se identificaron utilizando documentación pública oficial, una de las diferencias principales fue la cantidad de ejemplos y de información pública disponible, en el caso de Azure es menor que en AWS por lo que tomó más tiempo investigar y comprender cuales controles debían ser implementados y cómo.

El resumen de los controles identificados se muestra en la tabla 4. En ella se listan los controles identificados para la mitigación de cada una de las amenazas listadas en la tabla 3 anterior. En la próxima sección se detallan las implementaciones de cada control en cada proveedor.

Tabla 4: Controles identificados en cada proveedor para cubrir las amenazas seleccionadas

Nombre de la amenaza	Servicios AWS	Servicios Azure
Pérdida de autenticación	AWS Cognito	Azure AD
Exposición de datos sensibles	AWS Key Management Service(KMS)	Azure Key Vault
Pérdida de control de acceso	AWS Identity and Access Management(IAM)	Azure AD

Nombre de la amenaza	Servicios AWS	Servicios Azure
Secuencia de comandos en sitios cruzados (XSS)	AWS Web Application Firewall(WAF)	Azure Firewall Manager/ Azure Front Door
Registros y monitoreo insuficiente	AWS X-ray/ AWS CloudWatch	Azure Monitor

3.5 Implementación

La implementación se realizó utilizando el diseño básico de la aplicación como se detalló en la sección anterior para cada uno de los proveedores de servicios en la nube seleccionados. La implementación constó de 4 etapas principales: crear la base de datos, crear la función como servicio, crear la API y finalmente implementar los controles de seguridad identificados en la sección 3.4 anteriormente. En las subsecciones a continuación, se detalla cada una de estas etapas para cada proveedor seleccionado.

3.5.1 Implementación en AWS

La aplicación utilizó el diseño según se especificó en la metodología. El código implementado en cada proveedor es distinto dado que la forma en que cada plataforma implementa y ejecuta el lenguaje es diferente, el código especifico se encuentra en la referencia de GitHub que se compartió en la sección 3.4.

Los servicios en AWS se pueden utilizar y administrar por medio de una interfaz de usuario que se puede accesar utilizando un navegador web o por medio de herramientas programáticas. Para este trabajo se utilizó la interfaz de usuario para ilustrar los pasos con claridad.

3.5.1.1 Implementación de la aplicación

En la primera etapa de la implementación, la creación de la base de datos, los pasos detallados que se siguieron se encuentran en la documentación pública del proveedor (Creación de una instancia de base de datos de Amazon RDS, 2020), y los pasos generales que se tomaron se mencionan a continuación:

1. Se escoge crear una nueva base de datos y se siguen los pasos en la documentación del proveedor.
2. Luego de creada la base de datos se crea una tabla llamada ítems con el siguiente código SQL:

```
CREATE TABLE `items` (
`item_id` int(11) unsigned NOT NULL AUTO_INCREMENT,
`name` varchar(100) DEFAULT NULL,
PRIMARY KEY (`account_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

3. Se insertan dos registros en la base de datos para poder probar la implementación cuando se complete.
4. La imagen de la base de datos completa y creada se muestra en la figura 17 que se localiza en el apéndice B.

La segunda etapa fue la creación de la función como servicio en el servicio Lambda de AWS, este se describió en la sección 2.2.1, para esto se utiliza la consola y se siguen los pasos detallados en la documentación del proveedor (Creación de una función Lambda con la consola, s.f.) con el código y la estructura mencionada en la sección 3.3.

La implementación de la FaaS completada se muestra en la figura 8 y 9 del apéndice B. En la figura 8 se muestra la estructura de la función y en la figura 9 se muestra el detalle del código utilizado que se encuentra en el repositorio de GitHub mencionado anteriormente.

La tercera etapa fue la creación de la API, para esto se utiliza API Gateway (Introducción a Amazon API Gateway, 2020), se siguieron los pasos detallados en la documentación del proveedor (Uso de AWS Lambda con Amazon API Gateway, 2020) y el método HTTP implementado según la información de la sección 3.3. La API completada junto con sus componentes se muestra en la figura 10 del apéndice B.

Una vez completadas las primeras tres etapas se finaliza la implementación del diseño según se mostró en la figura 5 anteriormente.

3.5.1.2 Implementación de los controles de seguridad

En esta subsección se cubre la implementación de los controles para cada uno de los cinco riesgos de seguridad seleccionados en la sección 3.4. En cada caso se describe la funcionalidad requerida, el servicio utilizado, los pasos generales que se tomaron y la actualización del diseño de la aplicación una vez que se completa cada una de las integraciones.

3.5.1.2.1 Pérdida de autenticación

La aplicación implementada está abierta para que cualquiera en el mundo la pueda acceder. Un atacante podría abusar el uso de la API de muchas formas y no habría manera de saber quién realmente hizo la solicitud.

Para bloquear el acceso a la API solo a usuarios conocidos, se debe agregar autenticación y autorización a la API. Las opciones disponibles para autorización en el servicio API

Gateway de AWS se detallan en la documentación pública (Control del acceso a API Gateway, 2020)

En AWS se utilizó Amazon Cognito (Introducción a Amazon Cognito, 2020) para actuar como el servidor de autorización para generar token de acceso, e implementar la integración con API Gateway para poder hacer la validación correspondiente de lo que se envía en la solicitud API.

Las actividades que se realizaron como parte de la implementación se tomaron de la documentación pública (Llamar a una API de REST integrada con un grupo de usuarios de Amazon Cognito, 2020) y los pasos principales se mencionan a continuación:

a. Crear y configurar el grupo de usuarios de Cognito.
b. Agregar una aplicación como parte del grupo de usuarios.
c. Crear credenciales de autorización para la aplicación agregada.
d. Configurar la API y sus recursos en API Gateway validar las credenciales con Cognito.

En las figuras 11 y 12 del apéndice B se muestran los detalles de la implementación, en la figura 11 se muestran los detalles de la autorización y en la figura 12 se muestra la referencia al autorizador de Cognito creado en API Gateway.

Una vez implementado el flujo de la solicitud se actualiza como se muestra en la figura 7, la autorización sucede en el portal de API Gateway, antes de la ejecución del API, para poder corroborar los permisos antes de proseguir.

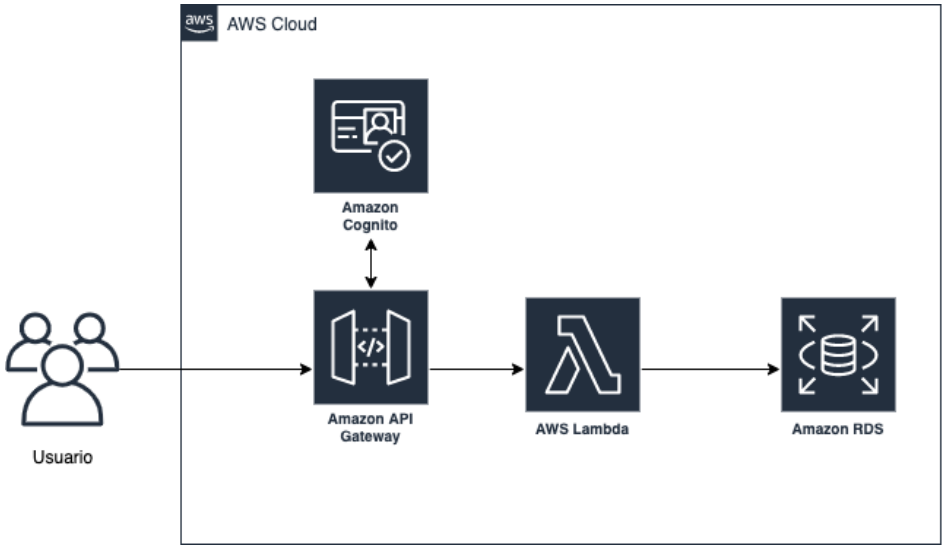


Figura 7: Actualización del diseño de la aplicación utilizada en AWS con el control para pérdida de autenticación. Elaboración propia.

3.5.1.2.2 Exposición de datos sensibles

Guardar la información de conexión y las credenciales de la base de datos de forma estática no es una práctica recomendada. Para mantener esta información resguardada y controlada de manera segura, se debe implementar acceso seguro y centralizado.

El control seleccionado para este riesgo fue AWS *Secrets Manager(AWS Secrets Manager, 2020)* y se utilizó para manejar las credenciales de base de datos. Además de mantener la base de datos segura, incluso si las personas tienen acceso para leer el código*, AWS Secrets Manager* también se integra con el servicio de base de datos RDS y puede gestionar automáticamente las rotaciones de contraseña.

Las actividades que se realizaron como parte de la implementación se tomaron de la documentación pública (Rotación de secretos para bases de datos de Amazon RDS, 2020) y los pasos principales se mencionan a continuación:

1. Crear un secreto en Secrets Manager para guardar la información de conexión de la base de datos que se creó inicialmente. Al crear el secreto se copia el numero de identificación que se genera.
2. Se vuelve a las funciones de Lambda anteriormente creadas y se actualizan para eliminar la referencia textual a las credenciales de la base de datos, esto se logra agregándolas como variables de ambiente.
3. Modificar los permisos de Lambda con permisos apropiados para leer las credenciales desde Secrets Manager.

En las figuras 14 y 15 del apéndice B se muestran los detalles de la implementación, en la figura 14 se muestra la creación del secreto y en la figura 15 se muestra la actualización requerida en el código de la FaaS para poder integrarse al servicio automáticamente y llamar las credenciales adecuadamente.

Al implementar este flujo de verificación de credenciales se modifica nuevamente la arquitectura de la aplicación como se muestra en la figura 8, se realiza la solicitud al servicio de credenciales durante la ejecución del código para poder acceder a la información en la base de datos.

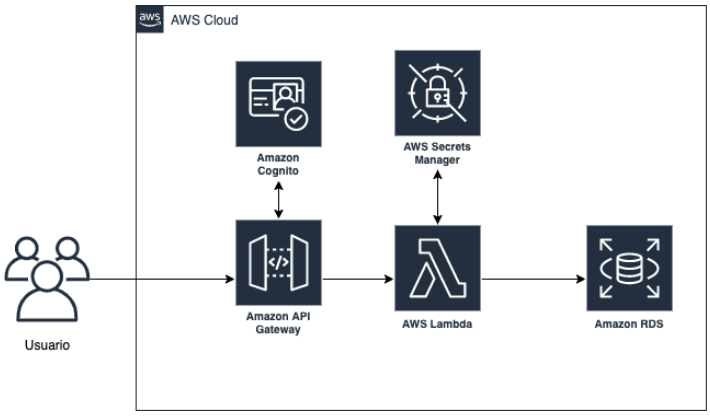


Figura 8: Diseño actualizado en AWS con el control para exposición de datos sensibles.

Elaboración propia.

3.5.1.2.3 Pérdida de control de acceso

La FaaS de la aplicación no poseen un usuario definido inicialmente, para poder limitar el acceso se debe utilizar una forma de limitar los permisos. Para controlar los permisos y accesos de la función se debe implementar control de acceso por medio de permisos granulares y específicos.

EL control seleccionado para este riesgo fue IAM (*Identity and Access Management*, Manejo de identidad y accesos) que permite definir acceso de usuarios, así como entre los servicios que tienen interacciones (¿Qué es IAM?, 2020).

Como parte de la implementación de los APIs se establecieron los siguientes roles y permisos para la ejecución y funcionamiento adecuado de las funciones, esto se muestran en las figuras 17 y 18 del apéndice B. La figura 17 muestra el detalle del rol utilizado y configurado en el servicio IAM. Los permisos listados en la figura 18 son los mínimos que se necesitan para poder ejecutar la funcionalidad requerida, si la funcionalidad o los componentes de la aplicación cambian estos podrían necesitar actualización. Cada permiso que forma parte del rol que se utiliza en la función permite acceder a recursos y ejecutar acciones dentro del sistema que se encuentran restringidas por defecto.

3.5.1.2.4 Secuencia de comandos en sitios cruzados (XSS)

La implementación inicial de la aplicación no tiene consideraciones de seguridad para inspeccionar y detectar ataques de aplicación web. Es necesario poder examinar las solicitudes que se reciben para determinar si pueden afectar la seguridad de la aplicación.

El control seleccionado para este riesgo fue WAF del inglés *Web Application Firewal l* (AWS WAF, 2020) que permite, por medio de reglas predefinidas y administradas por AWS, tener un control del contenido de las solicitudes que se reciben de las API, establecer alertas y acciones como bloquear solicitudes sospechosas.

Las actividades que se realizaron como parte de la implementación se tomaron de la documentación pública (Uso de AWS WAF para proteger sus API, 2020) y los pasos principales se mencionan a continuación:

a. Crear una política de WAF para proteger las APIs creadas en API Gateway.
b. Agregar condiciones especificas para detectar solicitudes web que contienen XSS.
c. Aplicar el cambio en modo bloqueo y detectar el tráfico.

En las figuras 19 y 20 del apéndice B se muestran los detalles de la implementación. En la figura 19 se muestran los detalles de la configuración de WAF para la función y en la figura 20 se muestran las reglas especificas utilizadas para el riesgo de XSS.

Al implementar este flujo de evaluación de solicitudes se modifica nuevamente el diseño de la aplicación como se muestra en la figura 9, se realiza la evaluación como un primer paso en el procesamiento de la solicitud, incluso antes de la autorización del usuario, y así prevenir el paso de solicitudes maliciosas.

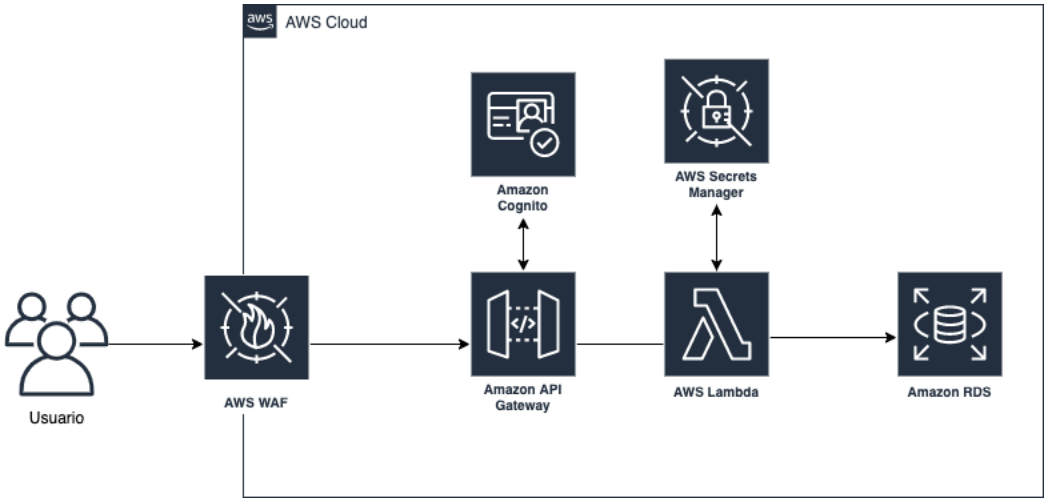


Figura 9: Actualización del diseño de la aplicación en AWS para el control de XSS. Elaboración propia.

3.5.1.2.5 Registros y monitoreo insuficiente

La implementación inicial de la aplicación no consideró el monitoreo de las transacciones, por lo que para poder mitigar este riesgo se utilizó un método que recolecta detalles de las solicitudes y ofrece información sobre lo que pasa en cada una y muestra información sobre las tendencias en general.

Para mitigar este riesgo se utilizó AWS X-Ray (AWS X-Ray, 2020), que brinda visibilidad del flujo de datos de la arquitectura de servicios web de la FaaS en AWS Lambda, y un mapa de cómo están conectados los componentes subyacentes de la aplicación. Existe otro servicio que se utiliza para monitoreo y visibilidad y complementa todos los servicios en la plataforma de AWS y ese es AWS CloudWatch (Amazon CloudWatch, 2020), este recolecta detalles de la operación de cada servicio y agrega la información de manera centraliza para visualizar datos operativos como duración, ejecuciones y errores.

Las actividades que se realizaron como parte de la implementación se tomaron de la documentación pública (Uso de AWS Lambda con AWS X-Ray, 2020) y los pasos principales se mencionan a continuación:

- Habilitar X-ray para Lambda al activar la opción en la consola de AWS, esto provee información sobre las invocaciones y las ejecuciones.
- Capturar las solicitudes del SDK con X-ray, esta opción da información más detallada sobre los llamados y permite anotaciones y metadata.
- Implementar los cambios en lambda y probar.
- Habilitar X-ray en API Gateway en la consola de AWS, desplegar las APIs nuevamente.

En la figura 22, que se encuentra en el apéndice B, se muestran los detalles de la implementación de este servicio, se activa de manera sencilla con un clic en la consola. También se muestra como la visibilidad de CloudWatch está activa de forma predefinida.

Al incluir y activar estos servicios de monitoreo se modifica nuevamente el diseño de la aplicación como se muestra en la figura 10. Los servicios de monitoreo recolectan la información para visualización de forma separada e independiente.

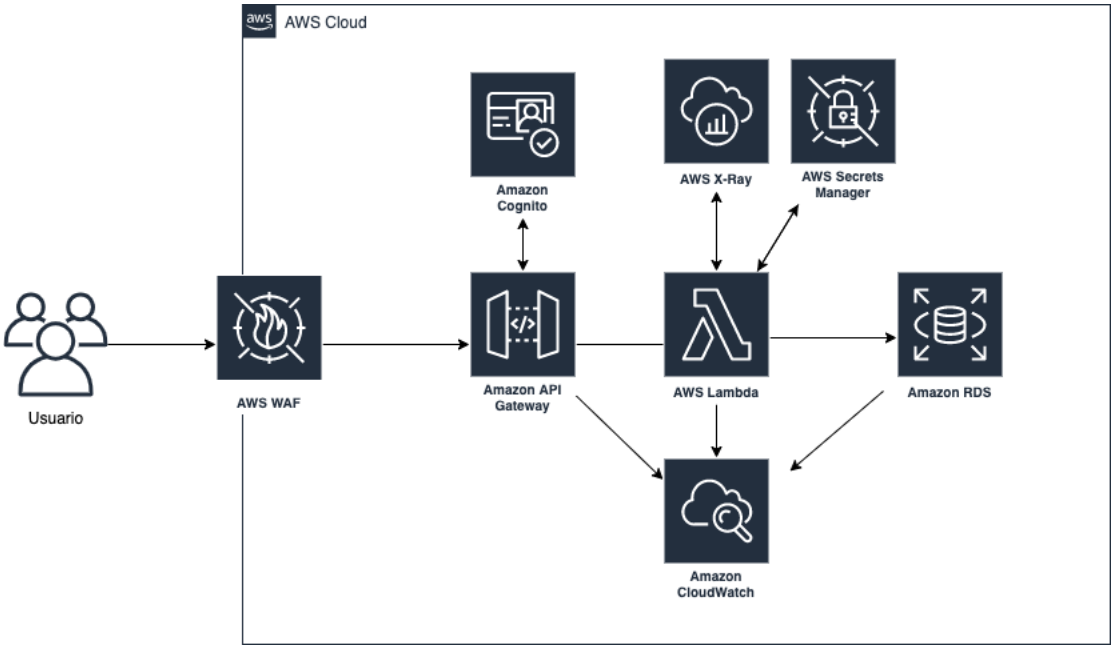


Figura 10: Actualización del diseño implementada en AWS con todos los controles identificados implementados. Elaboración propia.

3.5.2 Implementación en Azure

La aplicación utilizó el diseño según se especificó en la metodología. El código implementado en cada proveedor es distinto dado que la forma en que cada plataforma implementa y ejecuta el lenguaje es diferente, el código específico se encuentra en la referencia de GitHub que se compartió en la sección 3.4.

Los servicios en AWS se pueden utilizar y administrar por medio de una interfaz de usuario que se puede accesar utilizando un navegador web o por medio de herramientas programáticas. Para este trabajo se utilizó la interfaz de usuario para ilustrar los pasos con claridad.

3.5.2.2 Implementación de la aplicación

En la primera etapa de la implementación, la creación de la base de datos con el servicio Cosmos DB (Azure Cosmos DB, 2020), los pasos detallados que se siguieron se encuentran en la documentación pública del proveedor (Creación de una base de datos, un contenedor y elementos de Azure Cosmos desde Azure Portal, 2020), y los pasos generales que se tomaron se mencionan a continuación:

1. En la consola se busca el servicio Cosmos DB.
2. Se escoge crear una nueva base de datos.
3. Luego de creada la base de datos se crea una tabla llamada ítems.
4. Se insertaron datos para poder probar la implementación.

El detalle de la base de datos creada se muestra en la figura 24 del apéndice C, en ella se muestran el nombre y los datos de configuración del servicio utilizado.

La segunda etapa fue la creación de la función en Azure Functions, para esto se utilizó la consola y se siguen los pasos detallados en la documentación pública del proveedor (Creación de su primera función en Azure Portal, 2020) con el código que se localiza en GitHub (TFIA, 2020) y la estructura mencionada en la sección 3.4.

El detalle de la función como servicio creada se muestra en la figura 25 del apéndice C, en ella se muestran el nombre y los datos de configuración del servicio utilizado.

La tercera etapa fue la creación de la API, para esto se utilizó el servicio API Management Service (API management service, 2020), se siguieron los pasos detallados en la documentación del proveedor (Importación de una instancia de Azure Function App como API en Azure API Management, 2020) y la función como se definió en Azure Functions en el segundo paso. El resumen de los pasos generales se muestra a continuación:

a) Se crea una instancia de API Management.
b) Se definen los distintos tipos de métodos de la API a utilizar, según lo detallado en la sección 3.4.
c) Se conecta la definición de la función como servicio a la definición de la API.

La API completada junto con sus componentes se muestra en la figura 26 del apéndice C. Una vez completadas las primeras tres etapas se finaliza la implementación del diseño según se mostró en la figura 6 anteriormente.

3.5.2.3 Implementación de los controles de seguridad

En esta subsección se cubre la implementación de los controles para cada uno de los cinco riesgos de seguridad seleccionados en la sección 3.4. En cada caso se describe la funcionalidad requerida, el servicio utilizado, los pasos generales que se tomaron y la actualización del diseño de la aplicación una vez que se completa cada una de las integraciones.

La descripción de la funcionalidad general de cada control es equivalente a lo que se describió para los riesgos en el caso del proveedor de servicios AWS en la subsección 3.5.1.2, por lo que no se volverá a mencionar la misma información.

3.5.2.2.1 Pérdida de autenticación

Para la mitigación de este riesgo se utilizó Azure Active Directory (AD) (Azure Active Directory, 2020), siendo esta una implementación en la nube de la existente tecnología de AD de Microsoft, los pasos a seguir se detallan en la documentación del proveedor (Configuración de una aplicación de App Service o Azure Functions para usar el inicio de sesión de Azure AD, 2020), a continuación, se listan los pasos generales que se siguieron:

a. Crear un perfil para la aplicación de API Management que se creó anteriormente
b. Crear un perfil de autorización para una aplicación cliente. Esta se utiliza para probar el acceso
c. Se otorgan los permisos requeridos en AD
d. Se habilita la autorización OAuth para los usuarios de la aplicación
e. Se configura la validación del token por medio de una política para API Management

En las figuras 27 y 28 del apéndice C se muestran los detalles de la implementación. En la figura 27 se muestran los detalles de la configuración utilizada para la autorización de las solicitudes y en la figura 28 se muestran los permisos concedidos a la aplicación en la autorización creada.

Al implementar este servicio autenticación y autorización se modifica el diseño inicial de la aplicación como se muestra en la figura 11, la autenticación sucede antes de la ejecución de la función.

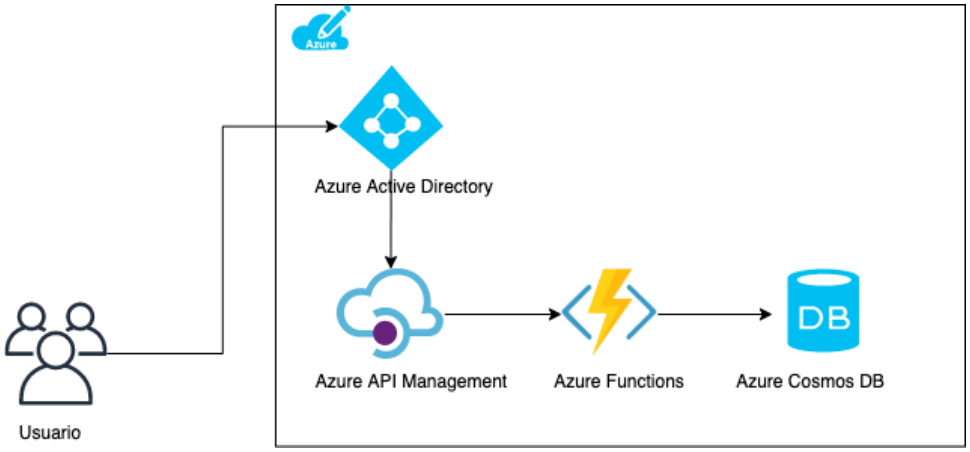


Figura 11: Actualización de la arquitectura utilizada en Azure para la implementación de la aplicación con el control de pérdida de autenticación. Elaboración propia.

3.5.2.4 Exposición de datos sensibles

El control implementado para este riesgo fue Azure es Key Vault (Azure Key Vault, 2020), que funciona de manera similar al que se implementó en AWS, Secrets Manager, los pasos de implementación se detallan en la documentación del proveedor (Seguridad simplificada para aplicaciones web y sin servidor con Azure Functions y App Service, 2018). Los pasos generales que se siguieron se listan a continuación:

a. Se crea el secreto de acceso a la base de datos en Key Vault.
b. Se copia la información del secreto creado a la función como servicio creada.
c. En Key Vault se autoriza a la función como servicio para el uso del secreto.
d. Se agrega el código del secreto como parte de la función.
e. Se modifica el código como se muestra debajo para cambiar el uso de las credenciales.

En las figuras 30 y 31 del apéndice C se muestran los detalles de la implementación. En la figura 30 se muestra el detalle del secreto creado y en la figura 31 se puede ver la manera

que se incluye la información dentro de la configuración de la FaaS para poder las credenciales durante su ejecución.

Al implementar este servicio de almacenamiento y administración de secretos se modifica el diseño inicial de la aplicación como se muestra en la figura 12, el llamado para crear la conexión a la base de datos sucede dentro de la ejecución de la función.

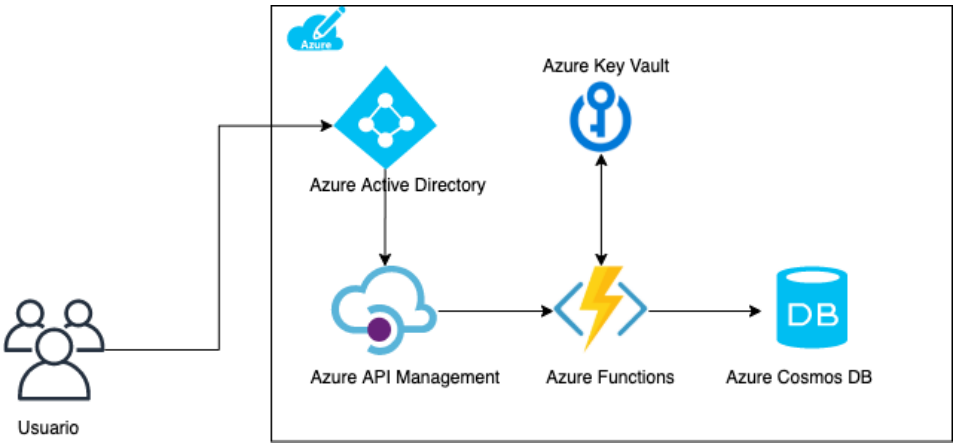


Figura 12: Actualización de la arquitectura utilizada en Azure para la implementación de la aplicación con el control de exposición de datos sensibles

3.5.2.2.3 Pérdida de control de acceso

Los controles de acceso en Azure se administran por medio de una integración con Azure Active Directory y establecen permisos basados en roles.

Para la implementación no se requirió establecer permisos ni roles dado que no existían otros usuarios dentro de la misma cuenta de Azure porque era de uso personal. Para una implementación a nivel empresarial sí se deben considerar estos privilegios y permisos, en la figura a continuación se muestran los detalles de configuración necesarios para

acceder y utilizar los servicios usando permisos basados en roles, como parte de una organización.

Como se nota en la figura 13, en la raíz del directorio se encuentra el administrador que tiene acceso a todos los recursos y servicios con acceso elevado, a partir de la raíz se pueden utilizar roles y grupos de administración, crear otros usuarios y definir permisos para uso de los recursos que se crean en Azure. Para esta implementación no hay grupos ni permisos específicos definidos para usuarios, tampoco fue necesario especificar ningún detalle de seguridad al intentar crear o ejecutar servicios durante toda la implementación.

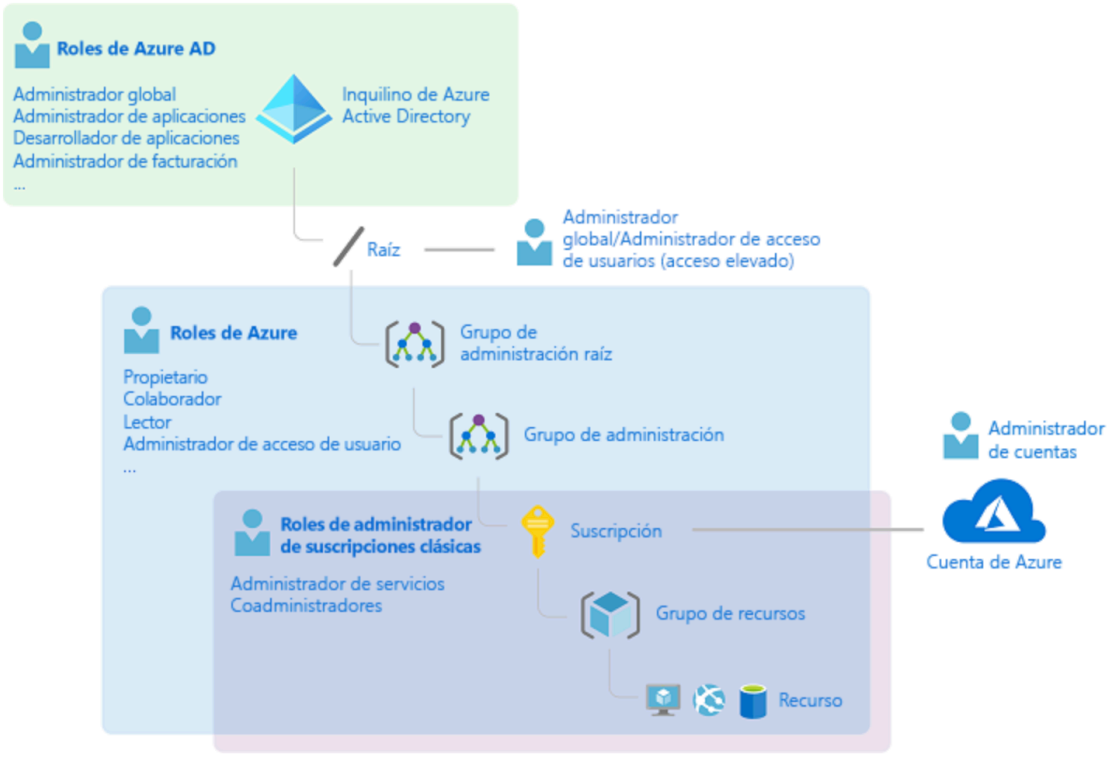


Figura 13: Detalles de las opciones de administración de permisos y usuarios en Azure. Tomado de (Azure, 2020)

3.5.2.2.4 Secuencia de comandos en sitios cruzados (XSS)

El control seleccionado para este riesgo fue Azure WAF (Azure Web Application Firewall, 2020), para poder utilizarlo se requirió agregar un servicio adicional a la solución, el servicio llamado Azure Front Door(Azure Front Door, 2020), que provee acceso global a la aplicación y ofrece la posibilidad de conectar Azure WAF con Azure API Management Service. Este servicio intermedio puede proveer servicios adicionales y avanzados a la aplicación que no son necesarios para cumplir con los objetivos establecidos en el alcance de este trabajo, por esta razón se utilizó exclusivamente para poder interconectar WAF.
Los pasos de implementación se detallan en la documentación del proveedor para Front Door (Cree una instancia de Front Door para una aplicación web global de alta disponibilidad, 2020) y WAF (Creación de una directiva de firewall de aplicaciones web en Azure Front Door mediante Azure Portal, 2020). Los pasos generales que se siguieron se listan a continuación:

1. Implementar Azure Front Door para poder agregar WAF.
2. Crear una nueva política de WAF.
3. Seleccionar el recurso como la implementación de API Management.
4. Seleccionar las reglas de XSS.
5. Seleccionar aplicar para activar la configuración seleccionada

En las figuras 34, 35 y 36 del apéndice c se muestran los detalles de la implementación. En la figura 34 se muestran los detalles de la configuración en Azure Front Door y en las figuras 35 y 36 se ilustran las reglas especificas que se utilizaron en la creación de los controles en Azure WAF.

Al implementar este servicio se modifica el diseño de la aplicación nuevamente y como se muestra en la figura 14, se nota que la solicitud de la API ahora inicia por Azure Front Door, en donde WAF examina el contenido de la información para identificar un posible ataque y poder bloquearlo al inicio del flujo de ejecución.

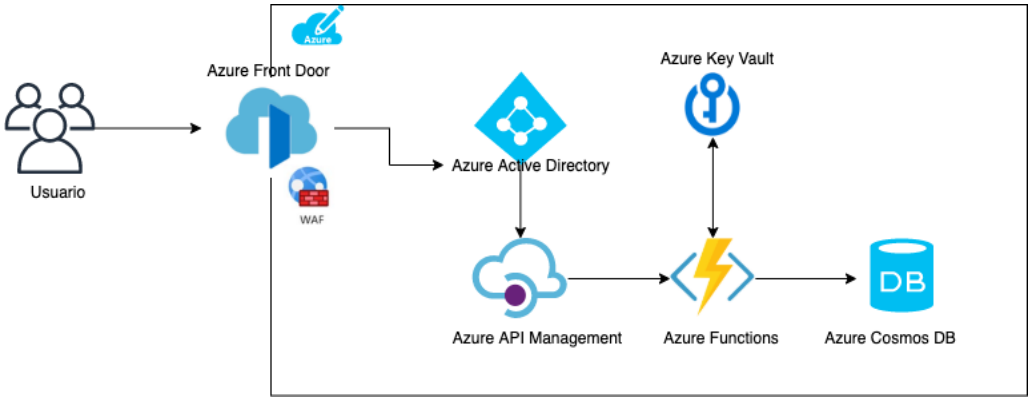


Figura 14: Actualización de la arquitectura utilizada en Azure para la implementación de la aplicación con el control de XSS

3.5.2.2.5 Registros y monitoreo insuficiente

El control de Azure seleccionado para este riesgo se llama Application Insights (¿Qué es Application Insights?, 2020), de la misma manera que AWS CloudWatch, recoge información sobre las solicitudes y puede dar detalles con el propósito de diagnosticar problemas y tener visibilidad de la actividad de la aplicación. En el caso de Azure, también es capaz de desplegar las dependencias de la función como servicio y detalles de su ejecución, de forma conjunta provee información sobre los otros servicios utilizados y sus métricas.

Las actividades que se realizaron como parte de la implementación se detallan en la documentación del proveedor (Supervisión de Azure Functions con Application Insights de Azure Monitor, 2020), el paso requerido fue habilitar Application Insights desde la consola de Azure Functions y dentro de la función que se creó.

En la figura 38 del apéndice c se muestran los detalles de la información requerida para poder activar el servicio, se requiere especificar un destino de la información para su análisis, así como el número de solicitudes que se quieren analizar en forma de porcentaje.

Al agregar este servicio se modifica el diseño de la aplicación nuevamente como se muestra en la figura 15, el servicio de recolección de información opera de forma independiente y es capaz de agregar información de los distintos servicios utilizados en Azure en un solo lugar.

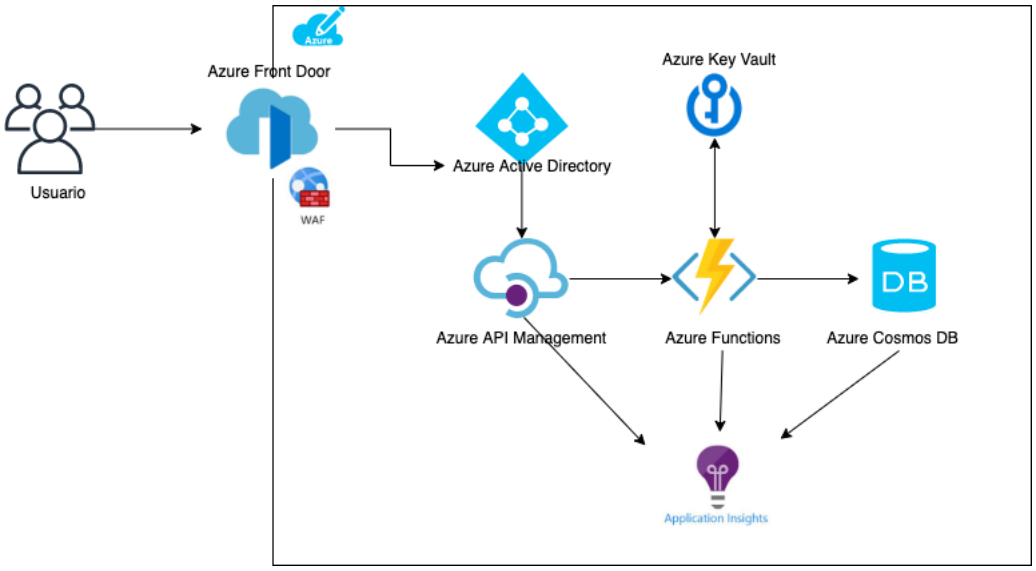


Figura 15: Actualización del diseño utilizado en Azure para la implementación de la aplicación con el control de monitoreo insuficiente. Elaboración propia.

En este capítulo se detallaron los pasos tomados para poder comparar los controles de seguridad para funciones como servicio en dos proveedores de servicios en la nube diferentes. Se detalló el proceso para escoger las amenazas, la aplicación a implementar y los proveedores de servicios en la nube. A continuación, se muestran los resultados obtenidos durante la implementación y comparación realizados.

CAPÍTULO IV: RESULTADOS

Los resultados de la implementación de la aplicación y cada uno de los controles seleccionados se cubre en las secciones que siguen. En la sección 4.1 se presentan los resultados de la creación de la aplicación base utilizada y la sección 4.2 se adentra en los detalles de los controles para cada uno de los riesgos seleccionados.

4.1 Resultados de la implementación general de la aplicación

Una de las diferencias más importantes que se encontró en la implementación de la aplicación básica con su función como servicio, fue que los servicios tienen diferencias fundamentales en la experiencia de usuario que hacen que la complejidad de la tarea sea distinta a pesar de que la función es fundamentalmente la misma.

En el caso de Azure la implementación se probó utilizando el navegador web y la consola de Azure. Las figuras 40 y 41 del apéndice c muestran el resultado al ejecutar la solicitud utilizando la FaaS implementada, esta despliega la información de la base de datos. En la figura 40 se ejecutó en el navegador web y en la figura 41 se probó la implementación desde la consola. Ambas fueron exitosas y se logró observar la información esperada.

De la misma forma se logró confirmar la implementación exitosa al realizar la ejecución de la solicitud en el caso de AWS. Las figuras 42 y 43 del apéndice c muestran el resultado desde el navegador web y desde la consola de forma correspondiente.

4.2 Resultados de los controles de seguridad

En esta sección se describen los resultados de la implementación de los controles de seguridad que se seleccionaron para cada uno de los proveedores de servicios en la nube.

Cada uno de los cinco controles se analizó en una subsección individual que incluye una definición del estado inicial de la aplicación, cómo el control cambió este estado y el resultado obtenido al ejecutar la solicitud web. Finalmente, se comparan los resultados de cada proveedor con respecto a la guía de OWASP.

4.2.1 Pérdida de autenticación

Al probar la implementación inicial la API se encuentra abierta a internet y cualquiera puede usarla sin tener que ofrecer credenciales o autorizarse de ninguna manera, esto no es un problema para un sistema de prueba como el se utilizó en este trabajo, pero para aplicaciones reales la seguridad y principalmente la autorización y autenticación son de suma importancia. Lo anterior define el estado inicial de la aplicación en donde se puede realizar una solicitud al API y recibir la respuesta sin proporcionar credenciales, luego de implementar el control de autorización se observa un error al intentar la solicitud sin credenciales.

Para probar el control implementado, en ambas implementaciones, se generó una solicitud al API luego de configurar la autorización, pero sin proveer las credenciales necesarias y se obtiene un error de acceso en la forma de error HTTP 401 de Acceso Denegado, esto se muestra en las figuras 44 y 45 del apéndice c.

Para obtener credenciales se debe registrar el usuario como autorizado y enviar el token en la solicitud, al reintentarlo con esta información, se nota como se recibe la respuesta esperada en la forma de un código HTTP 200 OK, esto se muestra en las figuras 46 y 47 del apéndice C para cada uno de los proveedores utilizados.

En la tabla 5 se comparan las implementaciones del control de autorización en cada proveedor y se observa que ambos cumplen con los lineamientos generales establecidos como aceptables por OWASP. En cada proveedor la implementación fue ligeramente

distinta y provee la cobertura de diferentes maneras, pero ambas fueron exitosas en mitigar el riesgo.

Tabla 5: Comparación de resultados para el control de pérdida de autenticación

Nombre de la amenaza	Controles de la nube sugeridos	Resultado AWS	Resultado Azure
Pérdida de autenticación	1. Utilice el tipo de identidad y control de acceso adecuado, si es posible, utilice las soluciones disponibles proporcionadas por el proveedor de servicios en la nube 2. Los recursos externos deben requerir autenticación y control de acceso según el servicio 3. Para la autenticación entre recursos internos, use métodos seguros conocidos, como federación de identidad	La solución es proporcionada por el proveedor, incluye todas las opciones en los controles sugeridos	La solución es proporcionada por el proveedor, incluye todas las opciones en los controles sugeridos

4.2.2 Exposición de datos sensibles

La implementación inicial de la función contiene las credenciales de conexión a la base de datos de forma legible o una referencia literal dentro del código. Lo anterior define el estado inicial como código con credenciales literales que pueden ser utilizadas por cualquier solicitud y leídas por cualquier persona con acceso al código. Al implementar el manejo de secretos se modificó el código para llamar las credenciales de manera segura, desde un repositorio controlado por el proveedor de servicios y configurado por el usuario.

Para probar el control implementado, se realizó la solicitud web luego de configurar las credenciales de forma encriptada utilizando los servicios que se describieron para el control en la metodología. El resultado obtenido para cada uno de los proveedores se muestra en las figuras 48 y 49 del apéndice c, en ellas se nota que la solicitud web es exitosa luego del cambio realizado y no hay cambios o errores visibles para el usuario.

En la tabla 6 se comparan las implementaciones en cada proveedor y se nota que el control implementado cumple con los lineamientos generales establecidos como aceptables por OWASP. En cada proveedor la implementación fue ligeramente distinta y provee la cobertura de diferentes maneras, pero ambas son exitosas en mitigar el riesgo.

Tabla 6: Comparación de resultados para el control de exposición de datos sensibles

Nombre de la amenaza	Controles de la nube sugeridos	Resultado AWS	Resultado Azure
Exposición de datos sensibles	1. Identificar y clasificar datos confidenciales 2. Almacenar solo datos confidenciales necesarios 3. Proteja los datos en reposo y en tránsito 4. Utilizar solo HTTPS para los APIs 5. Utilice los servicios del proveedor de servicios de nube para la gestión de claves y el cifrado de datos almacenados, secretos y variables de entorno	La solución es proporcionada por el proveedor, cubre el control sugerido y el control 4. Ofrece opciones para implementar la opción 3 a discreción y necesidad del usuario	La solución es proporcionada por el proveedor, cubre el control sugerido y el control 4. Ofrece opciones para implementar la opción 3 a discreción y necesidad del usuario

Ambos proveedores de servicios en la nube utilizan el mismo estándar de autorización en sus servicios por lo que las diferencias están en pequeños detalles de configuración durante la implementación del servicio utilizado.

Los servicios de los proveedores probaron ser muy similares en su implementación, en los cambios de código requerido y la arquitectura del servicio, no hay diferencias significativas que se puedan mencionar.

4.2.3 Pérdida de acceso de control

La implementación de la función en los dos proveedores probó ser imposible sin establecer controles de acceso.

En el caso de AWS la definición inicial de la función requiere la selección de un rol de ejecución lo que fuerza el establecimiento de permisos dentro del sistema, según los principios de seguridad mencionados anteriormente, se deben conceder los privilegios mínimos para el funcionamiento adecuado de la aplicación, en Amazon esto se logra por medio de roles de IAM.

En el caso de Microsoft los permisos no son requeridos en la configuración básica, al menos desde el punto de vista de una implementación de prueba como la presente, y a pesar de que los permisos son configurables, esto no impide la creación de la función o su funcionamiento. El único control que se requiere para la creación de la función es la autenticación como usuario de Azure para acceder a la consola, pero no requirió ninguna otra re-validación durante la implementación o entre los servicios mismos.

En las figuras 50 y 51 del apéndice c se muestra la configuración de acceso con la que se implementó la aplicación y la FaaS en Azure, en ellas se nota que no se configuraron usuarios ni roles de acceso específicos.

La configuración de acceso utilizada para AWS se muestra en la figura 52 del apéndice c, en este caso se tuvieron que incluir controles y permisos granulares para que la FaaS pudiera interactuar con los otros componentes de la aplicación como la base de datos.

En la tabla 7 se comparan las implementaciones para la amenaza de pérdida de control de acceso en cada proveedor y se nota que el control implementado cumple con los lineamientos generales establecidos como aceptables por OWASP. En cada proveedor la implementación fue ligeramente distinta y provee la cobertura de diferentes maneras, pero ambas son exitosas en mitigar el riesgo.

Tabla 7: Comparación de resultados para el control de pérdida de control de acceso

Nombre de la amenaza	Controles de la nube sugeridos	Resultado AWS	Resultado Azure
Pérdida de control de acceso	• Examine cada función cuidadosamente e intente seguir el principio de privilegio mínimo • Se recomienda automatizar el proceso de configuración de permisos para funciones • Siga las mejores prácticas de implementación de controles según el proveedor de servicios de la nube	Ofrece mecanismos claros para configurar privilegio mínimo	Ofrece control de acceso y roles con permisos pero no se configura como los servicios se acceden entre sí

Los servicios de los proveedores probaron ser muy diferentes en su implementación, en el caso de Azure la función no requiere de definición de permisos y roles explícitos. En el caso de Amazon, se deben especificar y considerar los permisos de manera granular para que la aplicación y cada componente que se le agrega funcionen de manera adecuada. La implementación de ambos provee seguridad, pero la de Amazon al ser explícita y granular requiere de un íntimo conocimiento de las definiciones de permisos en la plataforma y de su aplicación correcta para lograr que sea exitosa.

4.2.4 Secuencia de comandos en sitios cruzados (XSS)

La implementación inicial de la función no contiene controles de detección para ataques de aplicaciones web. Lo anterior define el estado inicial como un sistema sin capacidad de detección de ataques en las solicitudes recibidas, en las figuras 53 y 54 del apéndice c, se muestra como una solicitud con XSS es procesada de forma exitosa (respuesta HTTP 200 OK) en Azure y AWS respectivamente.

Al implementar el control en cada WAF, se logra examinar y evaluar cada solicitud recibida, así como detectar y bloquear las solicitudes que cumplen con las reglas de XSS incluidas, en las figuras 55 y 56 del apéndice c, se muestran las detecciones en cada proveedor y el error generado al usuario por la detección, en este caso es el código HTTP 403 de acceso prohibido.

En la tabla 8 se comparan las implementaciones en cada proveedor y se nota que el control implementado cumple con los lineamientos generales establecidos como aceptables por OWASP. Los servicios de los proveedores probaron ser muy similares en su implementación, en las opciones que ofrecen y la manera en que funciona el control. La diferencia puede encontrarse en la forma granular en que se detecta el intento de XSS, pero los detalles específicos no se pueden conocer dado que cada proveedor guarda la lógica de detección de manera confidencial.

Tabla 8: Comparación de resultados para el control de XSS

Nombre de la amenaza	Controles de la nube sugeridos	Resultado AWS	Resultado Azure
Secuencia de comandos en sitios cruzados (XSS)	• Codificar todos los datos no confiables antes enviarlo al cliente, así como usar marcos y encabezados conocidos • Utiliza un cortafuegos para aplicaciones web	El servicio proporcionado cubre el riesgo descrito con el control sugerido	El servicio proporcionado cubre el riesgo descrito con el control sugerido

4.2.5 Registros y monitoreo insuficiente

La implementación inicial de la función no incluye ninguna manera de visibilizar lo que pasa durante su ejecución. Lo anterior define el estado inicial como una aplicación sin monitoreo y sin capacidad de detección de errores ni dependencias.

Al implementar el control se logra visualizar cada solicitud en detalle, así como las dependencias y puntos de error en la ejecución dentro de los servicios utilizados en la plataforma. En las figuras 57-63 del apéndice c, se muestran las visualizaciones de cada proveedor, así como el detalle que cada uno provee que es diferente. En las figuras 57-59 se muestran los detalles de Azure Functions, se pueden visualizar datos sobre el desempeño y el mapeo de los servicios utilizados en conjunto con la FaaS, así como las tendencias de las solicitudes recibidas en el tiempo.

De la misma manera, en las figuras 60-63 del apéndice c, se muestra la información encontrada en AWS, se incluye el mismo tipo de información que se encuentra en Azure y además es posible examinar registros específicos de las solicitudes recibidas en el sistema.

En la tabla 9 se comparan las implementaciones en cada proveedor y se nota que el control implementado cumple con los lineamientos generales establecidos como aceptables por OWASP. Los servicios de los proveedores probaron ser muy similares en su implementación, ambos también ofrecen opciones adicionales de monitoreo y visibilidad que se pueden configurar para ahondar en la información y el detalle de la utilización de los servicios.

Tabla 9: Comparación de resultados para el control de registro y monitoreo insuficiente

Nombre de la amenaza	Controles de la nube sugeridos	Resultado AWS	Resultado Azure
Registros y monitoreo insuficiente	• Utilice las herramientas de monitoreo proporcionadas por el proveedor de servicios de nube para identificar e informar comportamientos no deseados • Implemente un mecanismo de auditoría y monitoreo para los datos que el proveedor de servicios de infraestructura no provee completamente	Se ofrecen múltiples servicios y opciones para monitoreo de la función como servicio y otros servicios utilizados en conjunto	Se ofrecen múltiples servicios y opciones para monitoreo de la función como servicio y otros servicios utilizados en conjunto

CAPÍTULO V: CONCLUSIONES Y TRABAJO A FUTURO

En el presente trabajo se planteó la evaluación de controles de seguridad para funciones como servicio en plataformas de servicios en la nube y para un grupo de amenazas conocidas.

La selección de las amenazas de seguridad se realizó por medio de una investigación bibliográfica, se debieron identificar entidades y organizaciones dedicadas a la seguridad informática y que publican información confiable sobre el tema. De la misma manera, los proveedores de servicios de la nube se escogieron utilizando criterios publicados por organizaciones con experiencia en la evaluación de los mismos según los servicios que ofrecen, para el caso de este trabajo el enfoque fue las funciones como servicio.

Durante el experimento se trabajó con una aplicación web con componentes de uso común. La aplicación se implementó en los dos proveedores de funciones como servicio seleccionados. El experimento consistió en implementar mitigaciones para los principales riesgos de seguridad de esta aplicación, en cada uno de los proveedores seleccionados, estas mitigaciones son configuradas como controles que los proveedores ofrecen en sus distintos servicios. Los controles se seleccionaron al investigar las opciones y recomendaciones ofrecidas por cada uno de los proveedores en su documentación oficial pública. Al implementar los controles se agregaron al diseño inicial de la función como servicio.

Se evaluó la aplicación y su comportamiento antes y después de agregar cada control para corroborar la cobertura adecuada de la amenaza. También se compararon los proveedores del servicio y sus opciones disponibles para cumplir este propósito.

En la sección 5.1 se presentarán las conclusiones y en la sección 5.2 el trabajo a futuro.

5.1 Conclusiones

Al inicio de la investigación se planteó identificar al menos cinco amenazas de seguridad para la función como servicio. Dado que el campo de seguridad es amplio, inicialmente se pensó que se encontraría una cantidad mayor. Sin embargo, las guías y estándares conocidos únicamente identificaron cinco amenazas que aplicaban de manera directa a proveedores de servicios en la nube: pérdida de autenticación, exposición de datos sensibles, Entidades Externas XML(XXE), pérdida de control de acceso y registros y monitoreo insuficiente. La elección de dichas amenazas fue exitosa en demostrar amenazas inherentes en la función como servicio, así como para encontrar opciones de mitigación en los proveedores escogidos.

La selección de la aplicación base se realizó tomando en cuenta componentes comunes en aplicaciones web, y de traducir estos componentes a los servicios que ofrece cada proveedor. La implementación se logró de manera exitosa dado que existe mucha documentación pública de parte de los proveedores para que los usuarios puedan lograr exactamente el objetivo que se buscaba al crear una aplicación web básica con tres componentes.

La gran variedad de proveedores de servicios en la nube hace de la selección de un par de ellos una tarea difícil que requiere de una evaluación confiable y objetiva de las capacidades de cada uno. Algunas organizaciones como Forrester (Forrester, 2020) proveen una herramienta no solo a las empresas sino también a usuarios individuales, estudiantes e investigadores para poder considerar objetivamente diferentes servicios de cada proveedor, aprender y conocer sobre las ofertas de cada uno, esto es importante ya que permite evaluar y comparar características de servicio que se ofrecen de forma imparcial. La selección de los proveedores de servicio fue exitosa ya que cada proveedor escogido ofrecía el servicio de manera completa y documentada, lo que hizo posible completar el experimento. De esta forma se cumplió el objetivo específico 2, al haber

implementado la aplicación que utiliza funciones como servicio, de manera exitosa en dos proveedores seleccionados.

La selección de los controles en cada proveedor fue un reto ya que cada uno ofrece muchas maneras de obtener el mismo objetivo y existen varias opciones de configuración en los servicios. La forma mejor documentada y exitosa fue utilizar la documentación pública de cada proveedor sobre mejores prácticas para función como servicio, así como ejemplos para poder identificar el mejor control y la configuración requerida para poder cubrir los riesgos seleccionados. Los controles encontrados luego de la revisión lograron ser los adecuados para las amenazas identificadas. De esta forma se cumplió el objetivo específico 1, al haber seleccionado cinco amenazas de seguridad y sus controles de seguridad más importantes.

La implementación de los controles de seguridad fue exitosa ya que la aplicación mantuvo su funcionalidad al incorporar los distintos servicios en cada proveedor. El proceso de implementación de los controles resultó ser extenso y enfocado en prueba y error, siendo el indicador principal la respuesta recibida por el servicio al realizar solicitudes de prueba, se utilizaron múltiples recursos en línea además de la documentación del proveedor para poder identificar el comportamiento esperado en cada control. La disponibilidad de información para cada proveedor resultó ser una diferencia importante ya que afectó el tiempo de implementación considerablemente. El proveedor con más tiempo de presencia en el mercado resultó ser del que se encontraba más información disponible, lo cual podría ser una variable importante a considerar en algún futuro proyecto.

En cada uno de los proveedores seleccionados y con los controles implementados fue posible cumplir con los criterios sugeridos de mitigación de OWASP. Los detalles de cada servicio y la configuración de cada uno resultó ser ligeramente diferente pero los lineamientos de seguridad se lograron cubrir de igual forma. La experiencia de usuario en cada proveedor es distinta e incluso el código utilizado en la función resulta ser diferente aun utilizando el mismo lenguaje. El diseño de la aplicación no es portable de un

proveedor a otro, lo cual es importante considerar para implementaciones a largo plazo. De esta forma se cumplió el objetivo específico 3, al comparar y evaluar los controles en los dos proveedores seleccionados de forma exitosa.

De acuerdo a lo descrito anteriormente se confirma que se cumplió el objetivo general del trabajo al lograr de forma satisfactoria evaluar los controles de seguridad disponibles en plataformas para funciones como servicio, para un grupo de amenazas bien conocidas.

5.2 Trabajo Futuro y Recomendaciones

La opción de automatizar la implementación de los controles de seguridad seleccionados por medio de la utilización de infraestructura como código (Artac et al, 2017), es definitivamente una posibilidad para explorar. En esta se utilizaría la estandarización y la simplificación en la implementación, para usuarios y empresas que desean crear sus funciones con la seguridad integrada desde su creación y no poseen el tiempo ni conocimiento de los servicios necesarios. Este es un tema que se puede analizar a fondo como trabajo a futuro.

La posibilidad de explorar otros controles de seguridad que puedan cumplir los objetivos de proteger de las amenazas seleccionadas utilizando una implementación diferente es otra de las opciones de trabajo a futuro. En los proveedores de servicios en la nube se notó la existencia de más de un camino para implementar funcionalidades o cumplir requerimientos por lo que otros servicios u opciones se pueden explorar también.

El tipo de aplicación que se exploró en este trabajo es tradicional y comúnmente utilizado, al usar otro tipo de aplicación, por ejemplo, una aplicación móvil, la manera de implementarla en los proveedores de la nube puede cambiar completamente y las

opciones de seguridad disponibles pueden variar, por lo que se lograría explorar otro grupo de servicios diferentes a fondo.

Otra opción a considerar a futuro es implementar este mismo experimento agregando o cambiando otro proveedor de servicios en la nube. De esta manera se puede analizar de mejor manera las mejores formas de implementar prácticas de seguridad utilizando la misma aplicación base con una función como servicio en distintos tipos de proveedores.

A modo de recomendación, la opción de utilizar servicios propietarios de los proveedores de la nube puede ofrecer una manera innovadora de explorar como la tecnología creada por los mismos puede compararse con la utilización de servicios tradicionales como servidores web y bases de datos.

BIBLIOGRAFIA

Almuhammadi & Alsaleh (2017). Information Security Maturity Model for Nist Cyber Security Framework. *Computer Science & Information Technology*, vol:51-62.

Alpernas, K., Flanagan, C., Fouladi, S., Ryzhyk, L., Sagiv, M., Schmitz, T., & Winstein, K. (2018). Secure serverless computing using dynamic information flow control. *arXiv preprint arXiv:1802.08984*.

Amazon. (11 de setiembre de 2020). Creación de una función Lambda con la consola. Obtenido de https://docs.aws.amazon.com/es_es/lambda/latest/dg/getting-started-create-function.html.

Amazon. (10 de setiembre de 2020). ¿Qué es IAM?. Obtenido de https://docs.aws.amazon.com/es_es/IAM/latest/UserGuide/introduction.html.

Amazon. (10 de setiembre de 2020). Uso de AWS Lambda con Amazon API Gateway. Obtenido de https://docs.aws.amazon.com/es_es/lambda/latest/dg/services-apigateway.html

Amazon. (10 de setiembre de 2020). Uso de AWS WAF para proteger sus API. Obtenido de https://docs.aws.amazon.com/es_es/apigateway/latest/developerguide/apigateway-control-access-aws-waf.html

Amazon. (11 de setiembre de 2020). Crear un servidor web y una instancia de base de datos de Amazon RDS. Obtenido de https://docs.aws.amazon.com/es_es/AmazonRDS/latest/UserGuide/TUT_WebAppWithRDS.html.

Amazon. (12 de setiembre de 2020). Uso de AWS Lambda con AWS X-Ray. Obtenido de https://docs.aws.amazon.com/es_es/lambda/latest/dg/services-xray.html.

Amazon. (15 de setiembre de 2020). Rotación de secretos para bases de datos de Amazon RDS admitidas. Obtenido de https://docs.aws.amazon.com/es_es/secretsmanager/latest/userguide/rotating-secrets-rds.html.

Amazon. (16 de setiembre de 2020). AWS Lambda. Obtenido de: https://docs.aws.amazon.com/es_es/lambda/latest/dg/welcome.html.

Amazon. (17 de septiembre de 2020). Llamar a una API de REST integrada con un grupo de usuarios de Amazon Cognito. Obtenido de: https://docs.aws.amazon.com/es_es/apigateway/latest/developerguide/apigateway-invoke-api-integrated-with-cognito-user-pool.html.

Amazon. (18 de setiembre de 2020). Amazon CloudWatch. Obtenido de https://aws.amazon.com/es/cloudwatch/.

Amazon. (18 de setiembre de 2020). AWS Secrets Manager. Obtenido de https://aws.amazon.com/es/secrets-manager/.

Amazon. (2 de setiembre de 2020*). Overview AWS Lambda Security*. Obtenido de https://d1.awsstatic.com/whitepapers/Overview-AWS-Lambda-Security.pdf.

Amazon. (20 de Agosto de 2020). *AWS Lambda – Getting Started*. Obtenido de https://aws.amazon.com/lambda/getting-started/.

Amazon. (20 de setiembre de 2020). AWS X-ray. Obtenido de https://aws.amazon.com/es/xray/.

Amazon. (20 de setiembre). Introducción a Amazon Cognito. Obtenido de https://aws.amazon.com/es/cognito/getting-started/?nc1=h_ls.

Amazon. (21 de setiembre). AWS WAF- Web Application Firewall. Obtenido de https://aws.amazon.com/es/waf/.

Amazon. (3 de setiembre de 2020). Introducción a Amazon API Gateway. Obtenido de https://docs.aws.amazon.com/es_es/apigateway/latest/developerguide/getting-started.html.

Amazon. (4 de setiembre de 2020). Control del acceso a API Gateway. Obtenido de https://docs.aws.amazon.com/es_es/apigateway/latest/developerguide/getting-started.html.

Artac, M., Borovssak, T., Di Nitto, E., Guerriero, M. y Tamburri, D. A. (2017, May). DevOps: introducing infrastructure-as-code. *2017 IEEE/ACM 39th International Conference on Software Engineering Companion* (ICSE-C) (pp. 497-498).

Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V. y Suter, P. (2017). Serverless computing: Current trends and open problems. *In Research Advances in Cloud Computing (pp. 1-20)*.

Basu, S., Bardhan, A., Gupta, K., Saha, P., Pal, M., Bose, M. y Sarkar, P. (2018, January). Cloud computing security challenges and solutions-A survey. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 347-356).

Bila, N., Dettori, P., Kanso, A., Watanabe, Y., & Youssef, A. (2017, June). Leveraging the serverless architecture for securing linux containers. *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)* (pp. 401-404).

Chaudhary, S., Somani, G., & Buyya, R. (Eds.). (2017). Research advances in cloud computing. Springer Singapore.

Cloud Security Alliance. (10 de agosto de 2020). *About.* Obtenido de http://cloudsecurityalliance.org/.

Dhar, S., (2012). From outsourcing to Cloud computing: evolution of IT services. *Management Research Review*.

Eivy, A., & Weinman, J. (2017). Be wary of the economics of "serverless" cloud computing. *IEEE Cloud Computing*, vol: 6-12.

Forrester. (5 de agosto de 2020). *New Wave Function-as-a-Service Platforms, Q1 2020*. Obtenido de https://reprints.forrester.com/#/assets/2/108/RES155938/reports.

Garg, S. K., Versteeg, S., & Buyya, R. (2011). Smicloud: A framework for comparing and ranking cloud services. *2011 Fourth IEEE International Conference on Utility and Cloud Computing* (pp. 210-218).

GitHub. (Agosto de 2020). TFIA. Obtenido de https://github.com/dianaalse/TFIA

Glikson, A., Nastic, S., & Dustdar, S. (2017). Deviceless edge computing: extending serverless computing to the edge of the network. *Proceedings of the 10th ACM International Systems and Storage Conference* (pp. 1-1)

Google Cloud. (20 de Agosto de 2020*). Cloud Functions - Event-driven Serverless Computing*. Obtenido de https://cloud.google.com/functions/

Hendrickson, S., Sturdevant, S., Harter, T., Venkataramani, V., Arpaci-Dusseau, A. C., & Arpaci-Dusseau, R. H. (2016). Serverless computation with openlambda. 8th {*USENIX} Workshop on Hot Topics in Cloud Computing*.

Höfer, C. N., & Karagiannis, G. (2011). Cloud computing services: taxonomy and comparison. *Journal of Internet Services and Applications*, vol: 81-94.

Hong, S., Srivastava, A., Shambrook, W., & Dumitraș, T. (2018). Go serverless: securing cloud via serverless design patterns. *10th {USENIX} Workshop on Hot Topics in Cloud Computing* (HotCloud 18)

Jin, H., Ibrahim, S., Bell, T., Gao, W., Huang, D., & Wu, S. (2010). Cloud types and services. *Handbook of Cloud Computing* (pp. 335-355).

Jonas, E., Schleier-Smith, J., Sreekanti, V., Tsai, C. C., Khandelwal, A., Pu, Q., ... & Gonzalez, J. E. (2019). Cloud programming simplified: A Berkeley view on serverless computing. arXiv preprint arXiv:1902.03383.

Karabek, M.R., Kleinert, J. and Pohl, A., (2011). Cloud Services for SMEs–Evolution or Revolution?'. *Business+ Innovation*, pp.26-33.

Lee, H., Satyam, K., & Fox, G. (2018, July). Evaluation of production serverless computing environments. *2018 IEEE 11th International Conference on Cloud Computing* (CLOUD) (pp. 442-450).

Li, A., Yang, X., Kandula, S., & Zhang, M. (2010, November). CloudCmp: comparing public cloud providers. *Proceedings of the 10th ACM SIGCOMM conference on Internet measuremen*t (pp. 1-14).

Lynn, T., Rosati, P., Lejeune, A., & Emeakaroha, V. (2017). A preliminary review of enterprise serverless cloud computing (function-as-a-service) platforms. *2017 IEEE International Conference on Cloud Computing Technology and Science* (pp. 162-169).

Malawski, M., Gajek, A., Zima, A., Balis, B., & Figiela, K. (2017). Serverless execution of scientific workflows: Experiments with hyperflow, aws lambda and google cloud functions. *Future Generation Computer Systems*.

Madni, S. H. H., Abd Latiff, M. S., & Coulibaly, Y. (2016). Resource scheduling for infrastructure as a service (IaaS) in cloud computing: Challenges and opportunities. *Journal of Network and Computer Applications*, *vol:* 173-200.

McGrath, G., & Brenner, P. R. (2017, June). Serverless computing: Design, implementation, and performance. *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops* (ICDCSW) (pp. 405-410).

Microsoft Azure, (12 de agosto de 2020). Supervisión de Azure Functions con Application Insights de Azure Monitor. Obtenido de https://docs.microsoft.com/es-es/azure/azure-monitor/app/monitor-functions.

Microsoft Azure. (15 de agosto de 2020). Azure Key Vault. Obtenido de https://azure.microsoft.com/es-es/services/key-vault/.

Microsoft Azure. (15 de agosto de 2020). Configuración de una aplicación de App Service o Azure Functions para usar el inicio de sesión de Azure AD. Obtenido de: https://docs.microsoft.com/es-es/azure/app-service/configure-authentication-provider-aad?toc=%2fazure%2fazure-functions%2ftoc.json

Microsoft Azure. (15 de agosto de 2020). Creación de su primera función en Azure Portal. Obtenido de: https://docs.microsoft.com/es-es/azure/azure-functions/functions-create-first-azure-function

Microsoft Azure. (15 de agosto de 2020). Creación de una directiva de firewall de aplicaciones web en Azure Front Door mediante Azure Portal. Obtenido de https://docs.microsoft.com/es-es/azure/web-application-firewall/afds/waf-front-door-create-portal

Microsoft Azure. (15 de agosto de 2020). ¿Qué es Azure AD?. Obtenido de https://docs.microsoft.com/es-es/azure/active-directory/fundamentals/active-directory-whatis

Microsoft Azure. (16 de agosto de 2020). Azure Web Application Firewall Obtenido de https://azure.microsoft.com/en-us/services/web-application-firewall/

Microsoft Azure. (16 de agosto de 2020). Creación de una base de datos, un contenedor y elementos de Azure Cosmos desde Azure Portal. Obtenido de https://docs.microsoft.com/es-es/azure/cosmos-db/create-cosmosdb-resources-portal

Microsoft Azure. (17 de Agosto de 2020). Azure Application Insights. Obtenido de: https://docs.microsoft.com/es-es/azure/azure-monitor/app/app-insights-overview

Microsoft Azure. (17 de Agosto de 2020). Azure Front Door. Obtenido de https://azure.microsoft.com/es-es/services/frontdoor/

Microsoft Azure. (17 de agosto de 2020). Cree una instancia de Front Door para una aplicación web global de alta disponibilidad. Obtenido de https://docs.microsoft.com/es-es/azure/frontdoor/quickstart-create-front-door

Microsoft Azure. (17 de agosto de 2020). Importación de una instancia de Azure Function App como API en Azure API Management. Obtenido de https://docs.microsoft.com/es-es/azure/api-management/import-function-app-as-api.

Microsoft Azure. (18 de agosto de 2020). Seguridad simplificada para aplicaciones web y sin servidor con Azure Functions y App Service. Obtenido de

https://azure.microsoft.com/es-es/blog/simplifying-security-for-serverless-and-web-apps-with-azure-functions-and-app-service/.

Microsoft Azure. (18 de agosto de 2020). Supervisión de Azure Functions con Application Insights de Azure Monitor. Obtenido de: https://docs.microsoft.com/es-es/azure/azure-monitor/app/monitor-functions

Microsoft Azure. (18 de agosto, 2020). Azure Cosmos DB. Obtenido de: https://docs.microsoft.com/es-es/azure/cosmos-db/introduction

Microsoft Azure. (19 de Agosto de 2020.)Azure API management. Obtenido de https://azure.microsoft.com/es-es/services/api-management/

Microsoft Azure. (20 de agosto, 2020). Introducing Azure Functions. Obtenido de https://azure.microsoft.com/en-gb/blog/introducing-azure-functions/

Node. (5 de setiembre de 2020). Descripción. Obtenido de https://nodejs.org/es/

O'Meara, W., & Lennon, R. G. (2020, June). Serverless Computing Security: Protecting Application Logic*. 2020 31st Irish Signals and Systems Conference* (ISSC) (pp. 1-5).

OWASP. (12 de Agosto, 2020). *Main Page*. Obtenido de: www.owasp.org/index.php/Main_Page

OWASP. (20 de agosto de 2020). *Top Ten Project*. Obtenido de: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Pérez, A., Moltó, G., Caballer, M. y Calatrava, A. (2018). Serverless computing for container-based architectures. *Future Generation Computer Systems*, 83, 50-59.

Prodan, R., y Ostermann, S. (2009). A survey and taxonomy of infrastructure as a service and web hosting cloud providers. *2009 10th IEEE/ACM International Conference on Grid Computing* (pp. 17-25).

Shen, L., 2014. The NIST cybersecurity framework: Overview and potential impacts. Scitech Lawyer, p.16.

Villamizar, M., Garces, O., Ochoa, L., Castro, H., Salamanca, L., Verano, M. y Lang, M. (2016). Infrastructure cost comparison of running web applications in the cloud using AWS lambda and monolithic and microservice architectures. 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid) (pp. 179-182).

Wang, L., Li, M., Zhang, Y., Ristenpart, T., & Swift, M. (2018). Peeking behind the curtains of serverless platforms. *2018 Annual Technical Conference (USENIX ATC 18*) (pp. 133-146).

Zambrano, B. (2018). Serverless Design Patterns and Best Practices: Build, secure, and deploy enterprise ready serverless applications with AWS to improve developer productivity. Packt Publishing Ltd.

APENDICES

APENDICE A. INFORMACIÓN DETALLADA SOBRE LAS AMENAZAS DE SEGURIDAD SEGÚN OWASP

Tabla 10. Resumen de las amenazas para funciones como servicio según OWASP

Nombre de la amenaza	Resumen	Potencial Impacto
A1:2017 Inyección	Las fallas de inyección, ocurren cuando se envían datos que no son de confianza a un intérprete como parte de un comando o consulta	Los datos hostiles del atacante pueden engañar al intérprete para que ejecute comandos no deseados o acceda a los datos sin la debida autorización
A2:2017 Pérdida de autenticación	Las funciones de la aplicación relacionadas a autenticación y gestión de sesiones son implementadas incorrectamente, permitiendo a los atacantes comprometer usuarios y contraseñas, token de sesiones, o explotar otras fallas de implementación para asumir la identidad de otros usuarios (temporal o permanentemente)	Las funciones de la aplicación relacionadas con la autenticación y la administración de sesiones a menudo se implementan de manera incorrecta, lo que permite a los atacantes comprometer contraseñas, claves o tokens de sesión, o explotar otras fallas de implementación para asumir las identidades de otros usuarios de forma temporal o permanente.
A3:2017 Exposición de datos sensibles	Muchas aplicaciones web y APIs no protegen adecuadamente datos sensibles, tales como información financiera, de salud o Información Personalmente Identificable (PII). Los atacantes pueden robar o modificar estos datos protegidos inadecuadamente para llevar a cabo fraudes con tarjetas de crédito, robos de identidad u otros delitos. Los datos sensibles requieren métodos de protección adicionales, como el cifrado en almacenamiento y tránsito	Los atacantes pueden robar o modificar esos datos débilmente protegidos para cometer fraude con tarjetas de crédito, robo de identidad u otros delitos. Los datos confidenciales pueden verse comprometidos sin protección adicional
A4:2017 Entidades Externas XML(XXE)	Muchos procesadores XML antiguos o mal configurados evalúan referencias a entidades externas en documentos XML. Las entidades externas pueden utilizarse para revelar archivos internos mediante la URI o archivos internos en servidores no actualizados, escanear puertos de la LAN, ejecutar código de forma remota y realizar ataques de denegación de servicio (DoS).	Se pueden usar entidades externas para revelar archivos internos usando el URI del archivo

A5:2017 Pérdida de control de acceso	Las restricciones sobre lo que los usuarios autenticados pueden hacer no se aplican correctamente. Los atacantes pueden explotar estos defectos para acceder, de forma no autorizada, a funcionalidades y/o datos, cuentas de otros usuarios, ver archivos sensibles, modificar datos, cambiar derechos de acceso y permisos, etc	Los atacantes pueden aprovechar estas fallas para acceder a funciones y / o datos no autorizados, como acceder a las cuentas de otros usuarios, ver archivos confidenciales, modificar los datos de otros usuarios, cambiar los derechos de acceso, etc.
A6: 2017 Configuración de seguridad incorrecta	La configuración incorrecta de seguridad es el problema más común. Esto suele ser el resultado de configuraciones predeterminadas inseguras, configuraciones incompletas o ad hoc, almacenamiento en la nube abierta, encabezados HTTP mal configurados y mensajes de error detallados que contienen información confidencial	No solo todos los sistemas operativos, marcos, bibliotecas y aplicaciones deben estar configurados de forma segura, sino que también deben ser parcheados / actualizados de manera oportuna
A7:2017 Secuencia de comandos en sitios cruzados (XSS)	Los defectos de XSS ocurren cuando una aplicación incluye datos que no son de confianza en una nueva página web sin la validación o el escape adecuados, o actualiza una página web existente con datos proporcionados por el usuario mediante una API de navegador que puede crear HTML o JavaScript	XSS permite a los atacantes ejecutar scripts en el navegador de la víctima que pueden secuestrar sesiones de usuario, desfigurar sitios web o redirigir al usuario a sitios maliciosos
A8:2017 Deserialización insegura	Es una amenaza que se produce cuando se usan datos no confiables para abusar de la lógica de una aplicación, infligir un ataque o incluso ejecutar código arbitrario. La deserialización insegura a menudo conduce a la ejecución remota de código.	Incluso si las fallas de deserialización no dan como resultado la ejecución remota de código, se pueden usar para realizar ataques, incluidos ataques de reproducción, ataques de inyección y ataques de escalada de privilegios.
A9:2017 Componentes con amenazas conocidas	Los componentes, como bibliotecas, marcos y otros módulos de software, se ejecutan con los mismos privilegios que la aplicación. Si se explota un componente vulnerable, dicho ataque puede facilitar la pérdida de datos o la toma de control del servidor	Las aplicaciones y API que utilizan componentes con amenazas conocidas pueden socavar las defensas de las aplicaciones y permitir varios ataques e impactos

A10:2017 Registros y monitoreo insuficiente	El registro y la supervisión insuficientes, junto con una integración faltante o ineficaz con la respuesta a incidentes, permiten a los atacantes atacar aún más los sistemas, mantener la persistencia, cambiar a más sistemas y manipular, extraer o destruir datos	La mayoría de los estudios de infracciones muestran que el tiempo para detectar una infracción es de más de 200 días, generalmente detectados por partes externas en lugar de procesos internos o monitoreo

Tabla 11: Evaluación de las amenazas de OWASP de acuerdo al tipo de control sugerido

Nombre de la amenaza	Controles sugeridos	Incluye un control nativo de la nube
A1:2017 Inyección	• Utilización de validez de entrada • Ejecute funciones con los mínimos privilegios necesarios para realizar la tarea • Use una solución comercial de defensa en tiempo de ejecución para proteger las funciones	No
A2:2017 Pérdida de autenticación	• Utilice el tipo de identidad y control de acceso adecuado, si es posible, utilice las soluciones disponibles proporcionadas por el proveedor de servicios en la nube • Los recursos externos deben requerir autenticación y control de acceso según el servicio • Para la autenticación entre recursos internos, use métodos seguros conocidos, como federación de identidad	Sí
A3:2017 Exposición de datos sensibles	• Identificar y clasificar datos confidenciales • Almacenar solo datos confidenciales necesarios • Proteja los datos en reposo y en tránsito • Utilizar solo HTTPS para los APIs • Utilice los servicios del proveedor de servicios de nube para la gestión de claves y el cifrado de datos almacenados, secretos y variables de entorno	Sí
A4:2017 Entidades Externas XML(XXE)	• Use el SDK del proveedor de servicios de la nube siempre que sea posible • Analice la cadena de suministro en busca de amenazas conocidas de bibliotecas relevantes • Identifique y pruebe los ataques XXE a través de llamadas API • Deshabilitar la resolución de entidad	No
A5:2017 Pérdida de control de acceso	• Examine cada función cuidadosamente e intente seguir el principio de privilegio mínimo • Se recomienda automatizar el proceso de configuración de permisos para funciones • Siga las mejores prácticas de implementación de controles según el proveedor de servicios de la nube	Sí

A6: 2017 Configuración de seguridad incorrecta	Utilice herramientas automáticas que detecten configuraciones incorrectas de seguridad en aplicaciones sin servidor Establezca los tiempos de espera al mínimo requerido por la función Siga las sugerencias de configuración del proveedor de servicios de la nube Verifique las funciones con disparadores no vinculados	No
A7:2017 Secuencia de comandos en sitios cruzados (XSS)	• Codificar todos los datos no confiables antes enviarlo al cliente, así como usar marcos y encabezados conocidos • Utiliza un cortafuegos para aplicaciones web	Sí
A8:2017 Deserialización insegura	• Validar objetos serializados, que se originan a partir de datos no confiables • Revise las bibliotecas de terceros para detectar amenazas de deserialización conocidas • Monitorear el uso de deserialización y las excepciones para identificar posibles ataques	No
A9:2017 Componentes con amenazas conocidas	• Monitoree continuamente las dependencias y sus versiones en todo el sistema • Solo obtenga componentes de fuentes oficiales a través de enlaces seguros • Monitoree continuamente fuentes para detectar amenazas • Se recomienda escanear dependencias para detectar amenazas conocidas	No
A10:2017 Registros y monitoreo insuficiente	• Utilice las herramientas de monioreo proporcionadas por el proveedor de servicios de nube para identificar e informar comportamientos no deseados • Implemente un mecanismo de auditoría y monitoreo para los datos que el proveedor de servicios de infraestructura no provee completamente	Sí

APENDICE B. DETALLE DE LA IMPLEMENTACIÓN DE LA APLICACIÓN Y LOS CONTROLES DE SEGURIDAD

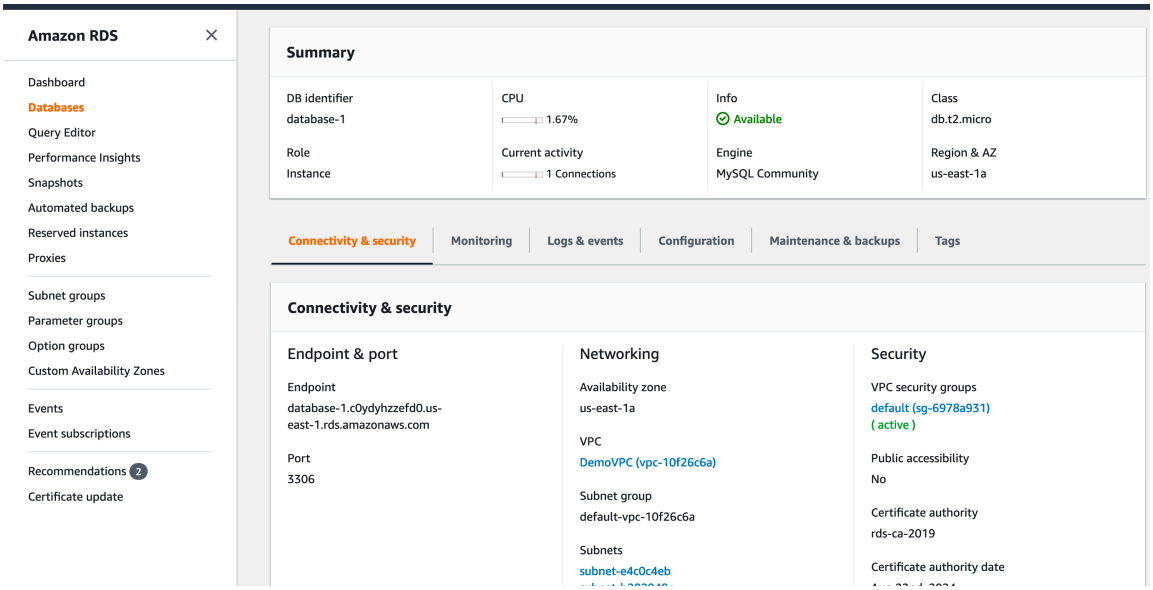


Figura 16: Imagen de la base de datos creada y listada en la consola AWS

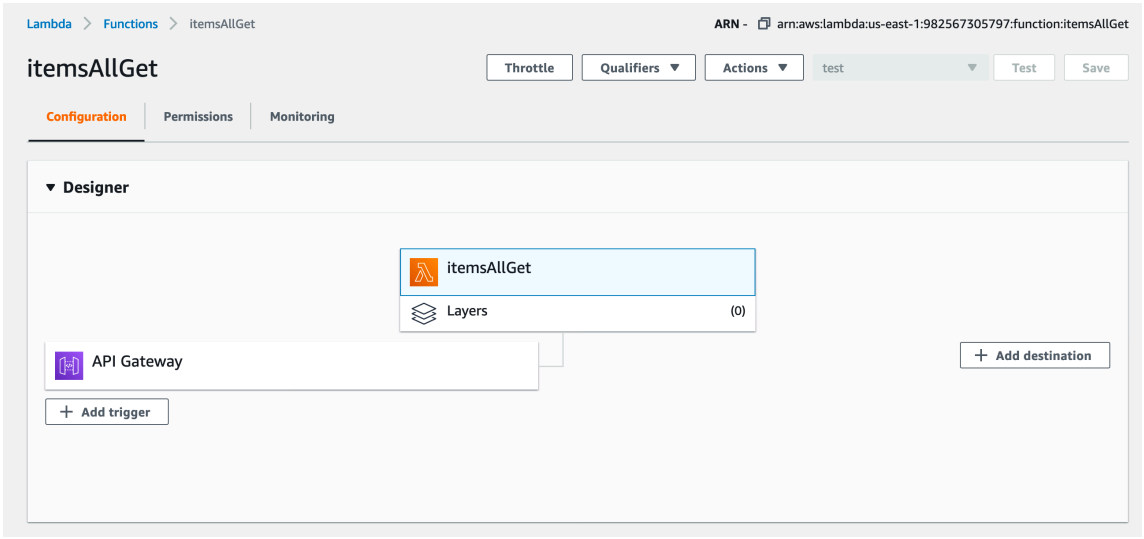


Figura 17: Imagen de la función como servicio creada en la consola de AWS

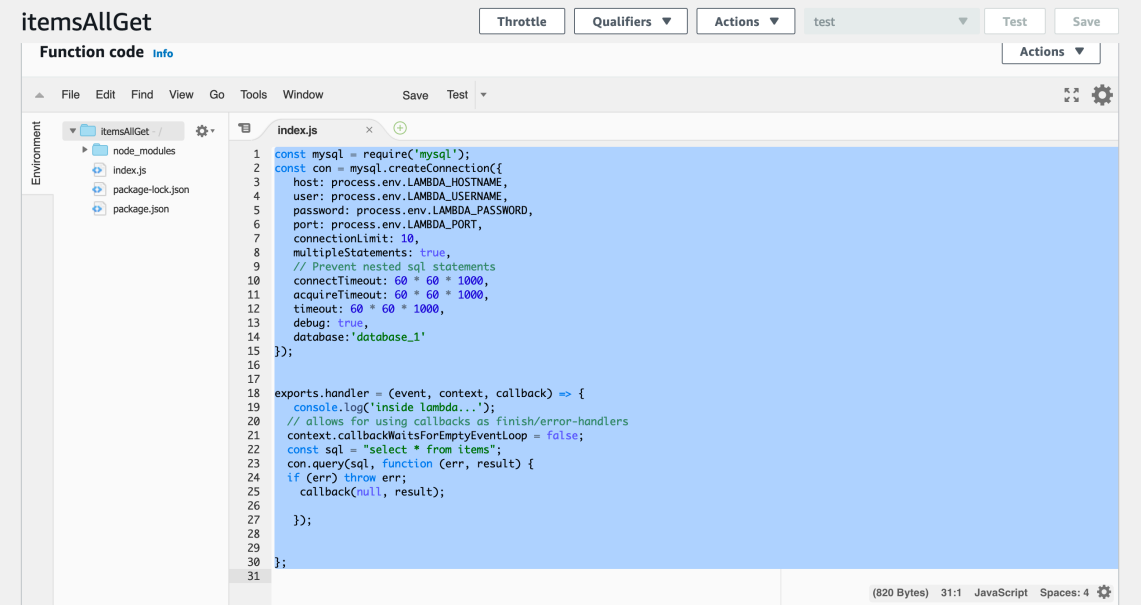


Figura 18: Muestra de los detalles de la función como servicio implementada en AWS

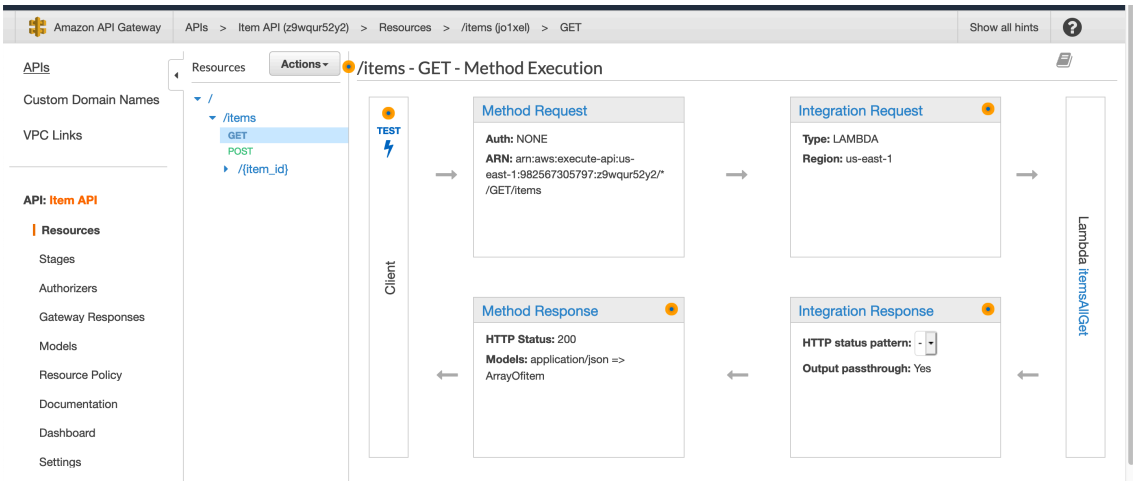


Figura 19: Detalles de la implementación de la API en AWS

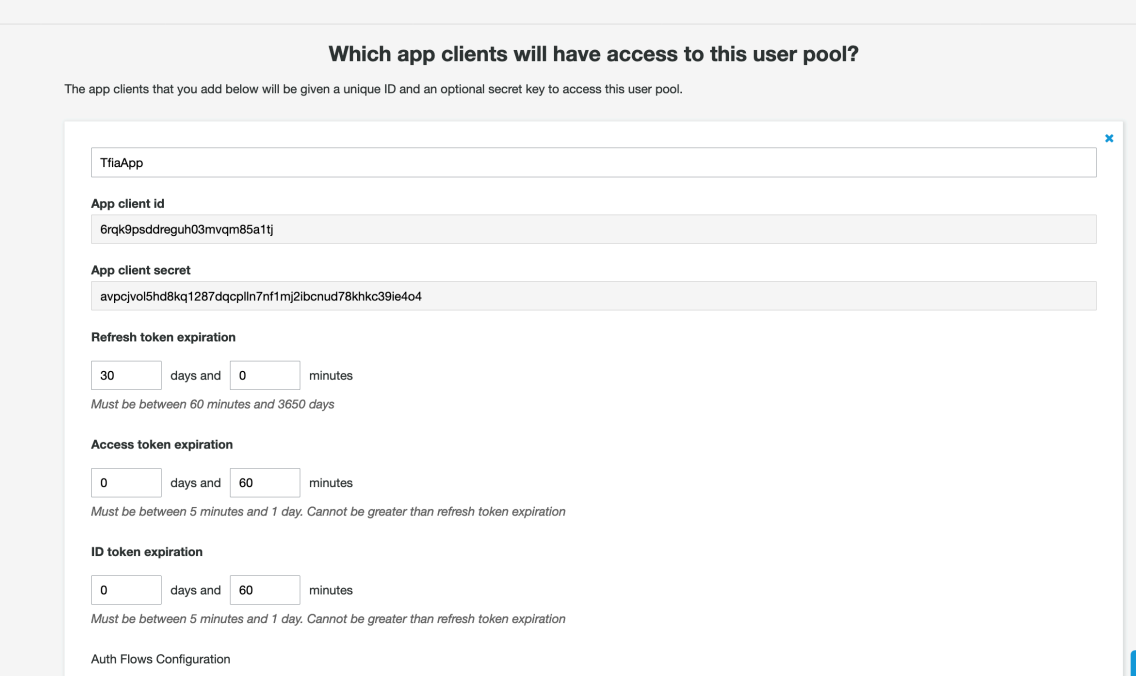


Figura 20: Implementación de la autenticación y autorización usando Cognito en AWS

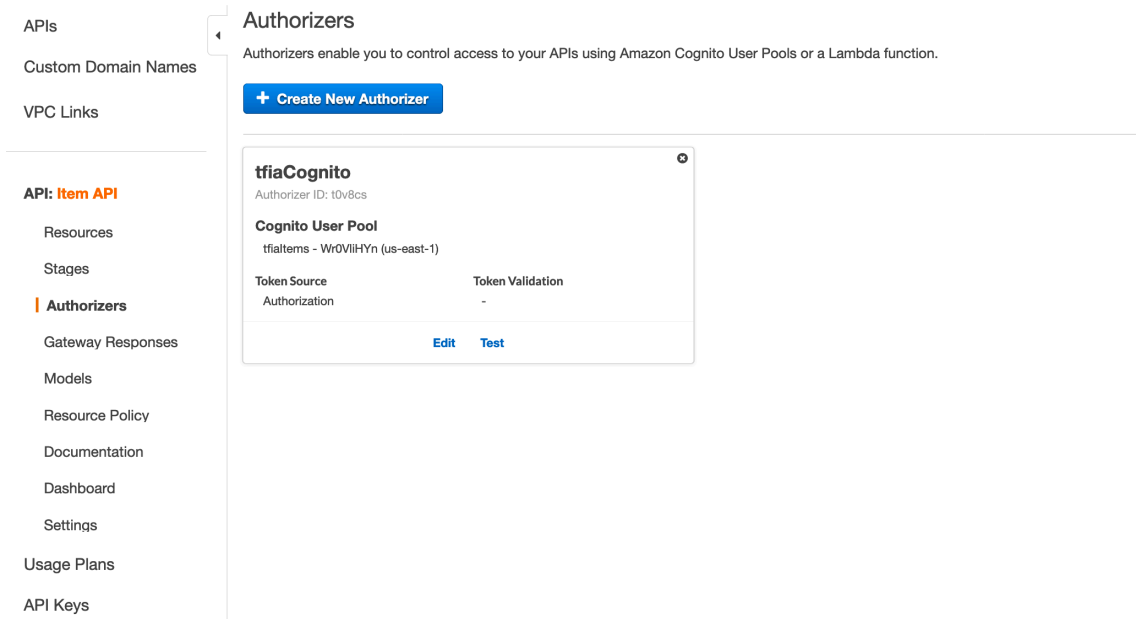


Figura 21: Detalles de la configuración de la validación del control en API Gateway en AWS

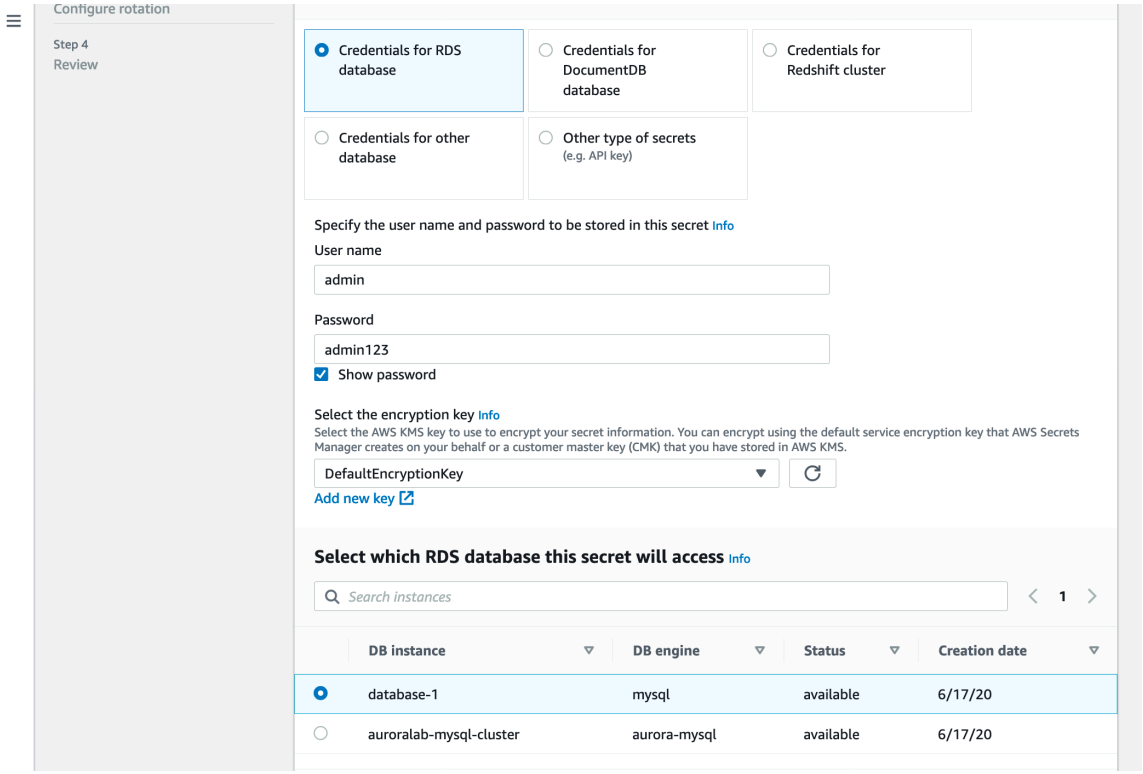


Figura 22: Creación del secreto para accesar la base de datos RDS en AWS

```
// Load the AWS SDK
var AWS = require('aws-sdk'),
    region = "us-east-1",
    secretName = "tfiaDB",
    secret,
    decodedBinarySecret;

// Create a Secrets Manager client
var client = new AWS.SecretsManager({
    region: region
});

// In this sample we only handle the specific exceptions for the 'GetSecretValue' API.
// See https://docs.aws.amazon.com/secretsmanager/latest/apireference/API_GetSecretValue.html
// We rethrow the exception by default.

client.getSecretValue({SecretId: secretName}, function(err, data) {
    if (err) {
        if (err.code === 'DecryptionFailureException')
            // Secrets Manager can't decrypt the protected secret text using the provided KMS key.
            // Deal with the exception here, and/or rethrow at your discretion.
            throw err;
        else if (err.code === 'InternalServiceErrorException')
            // An error occurred on the server side.
            // Deal with the exception here, and/or rethrow at your discretion.
            throw err;
        else if (err.code === 'InvalidParameterException')
            // You provided an invalid value for a parameter.
            // Deal with the exception here, and/or rethrow at your discretion.
            throw err;
        else if (err.code === 'InvalidRequestException')
            // You provided a parameter value that is not valid for the current state of the resource.
            // Deal with the exception here, and/or rethrow at your discretion.
            throw err;
        else if (err.code === 'ResourceNotFoundException')
            // We can't find the resource that you asked for.
            // Deal with the exception here, and/or rethrow at your discretion.
            throw err;
    }
    else {
        // Decrypts secret using the associated KMS CMK.
        // Depending on whether the secret is a string or binary, one of these fields will be populated.
        if ('SecretString' in data) {
            secret = data.SecretString;
        } else {
            let buff = new Buffer(data.SecretBinary, 'base64');
            decodedBinarySecret = buff.toString('ascii');
        }
    }
```

Figura 23: Detalles del cambio de código requerido en la función como servicio para utilizar el secreto creado

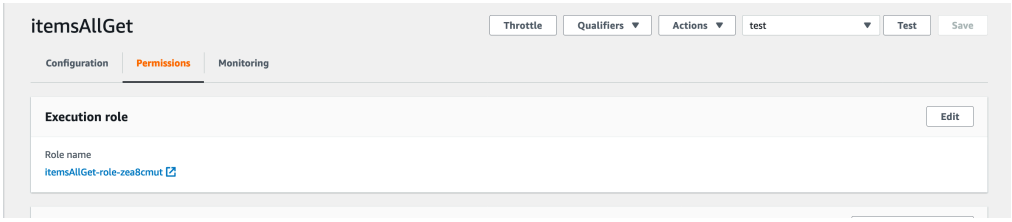


Figura 24: Detalle del role creado y utilizado para el funcionamiento de la función como servicio

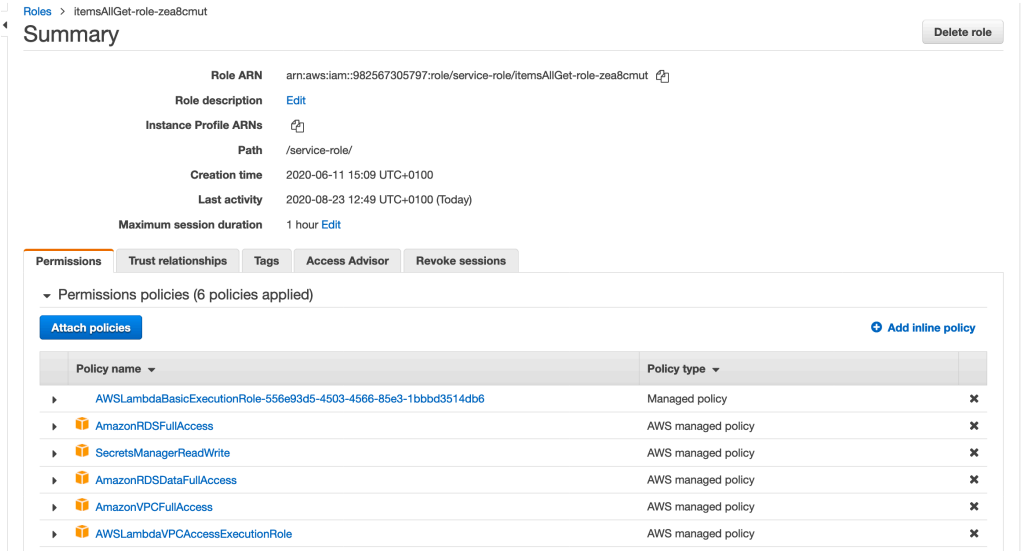


Figura 25: Detalles de los permisos específicos agregados al rol utilizado para la ejecución de la función como servicio

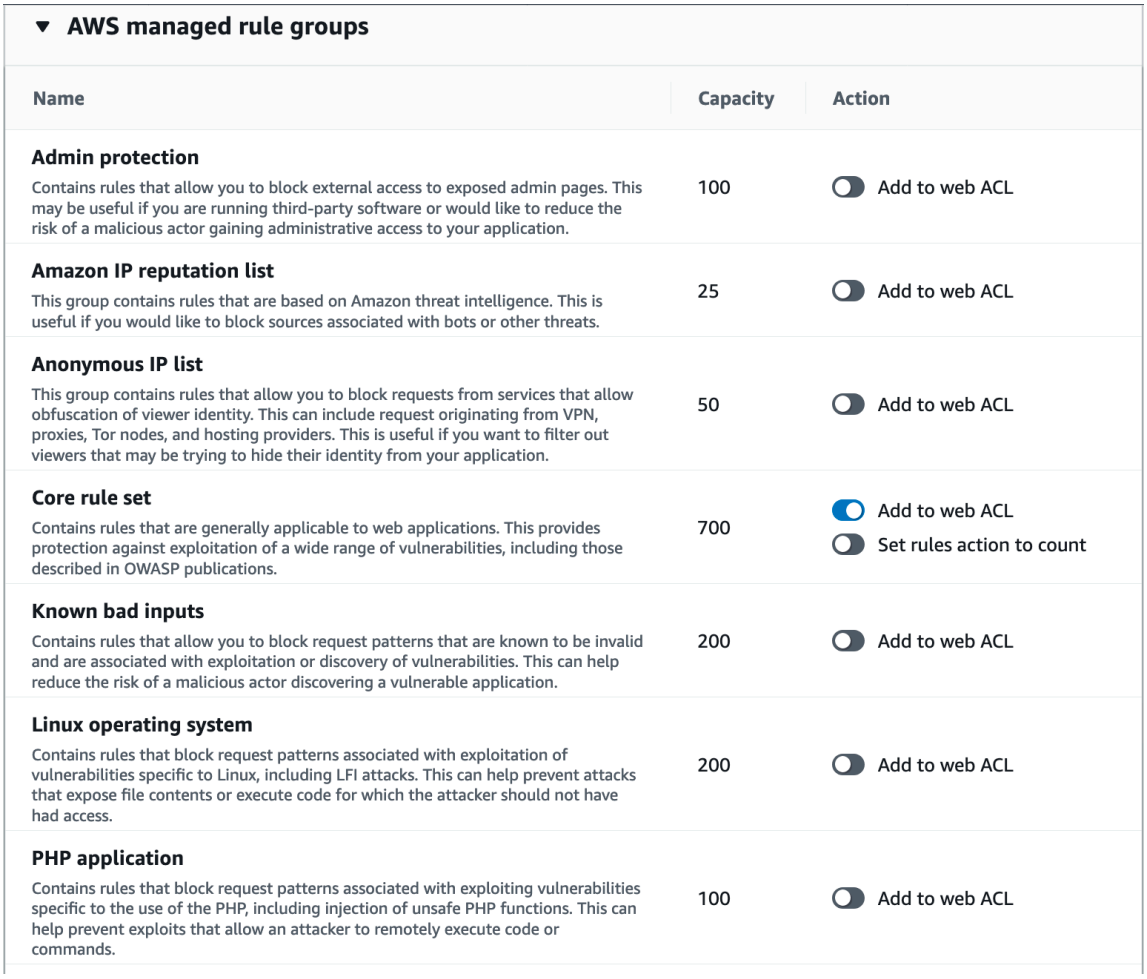


Figura 26: Detalle de los controles específicos implementados en la configuración de WAF

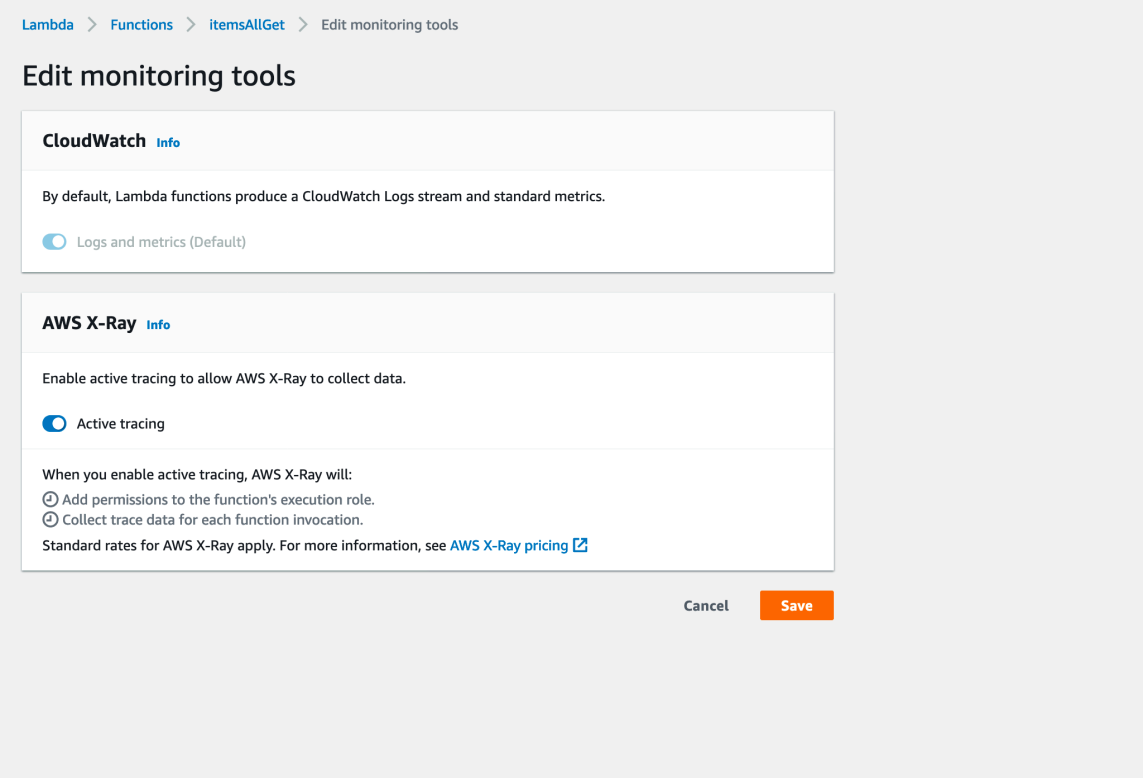


Figura 27: Detalle de la activación del servicio X-ray en la función como servicio

.

APENDICE C. DETALLES DE LOS RESULTADOS DE LA IMPLEMENTACION DE LOS CONTROLES

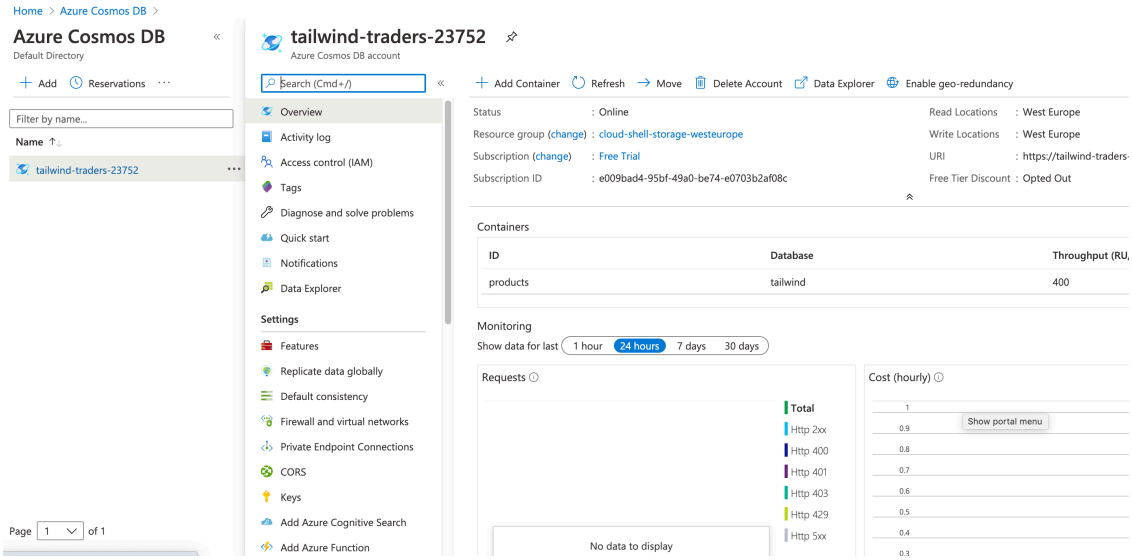


Figura 28: Detalles de las base de datos creada en Azure para la implementación de la aplicación

Create Function App

Subscription	b1dc360d-6b24-48dd-b36c-cc80fabf120a
Resource Group	Items
Name	Items
Runtime stack	Node.js - 12

Hosting

Storage (New)

Storage account	storageaccountitems88f2

Plan (New)

Plan type	Consumption (Serverless)
Name	ASP-Items-b652
Operating System	Linux
Region	East US
SKU	Dynamic

Monitoring (New)

Application Insights	Enabled
Name	Items
Region	East US

Figura 29: Detalles de la función como servicio creada en Azure Functions para la implementación de la aplicación

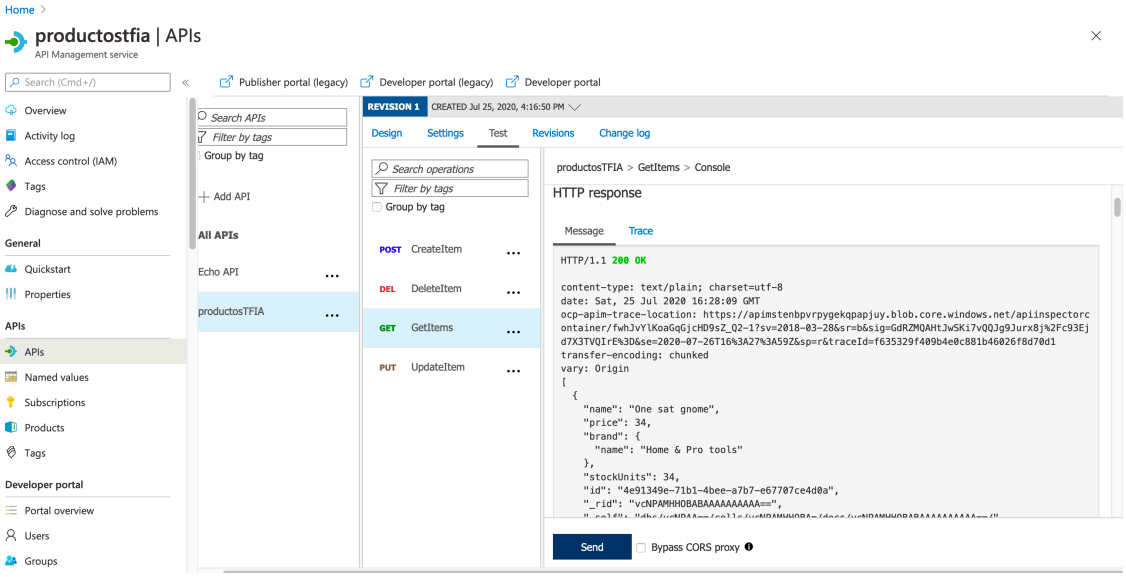


Figura 30: Detalles de la implementación de la aplicación en Azure utilizando API Management

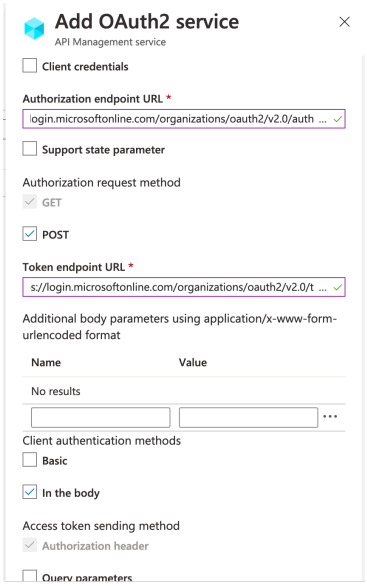


Figura 31: Detalles de la implementación del control de pérdida de autenticación en Azure

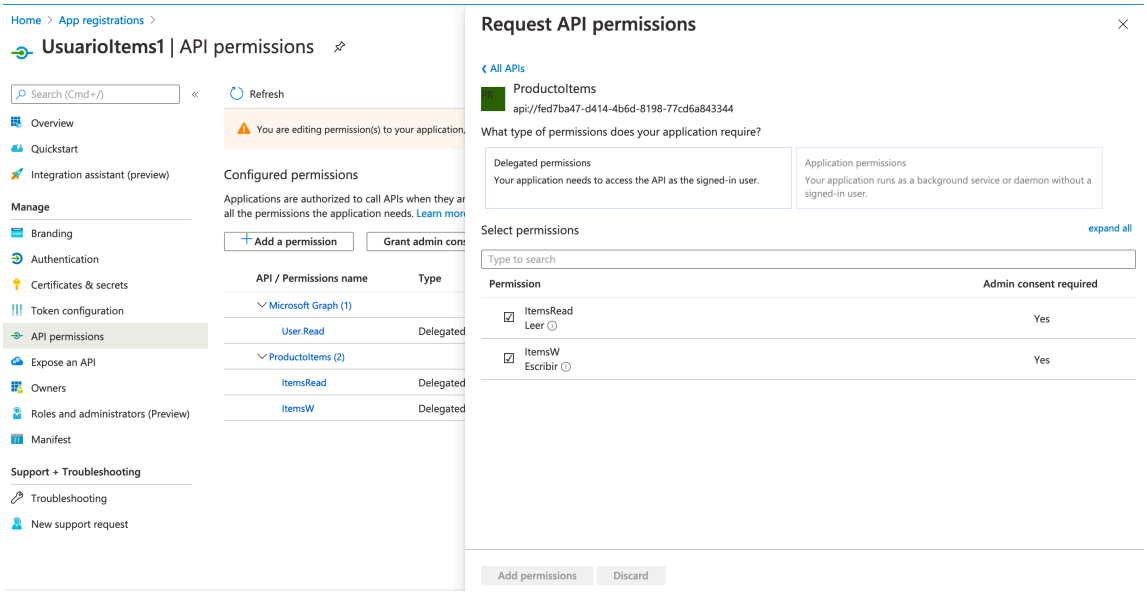


Figura 32: Detalles de la configuración de los detalles de autenticación y autorización en Azure

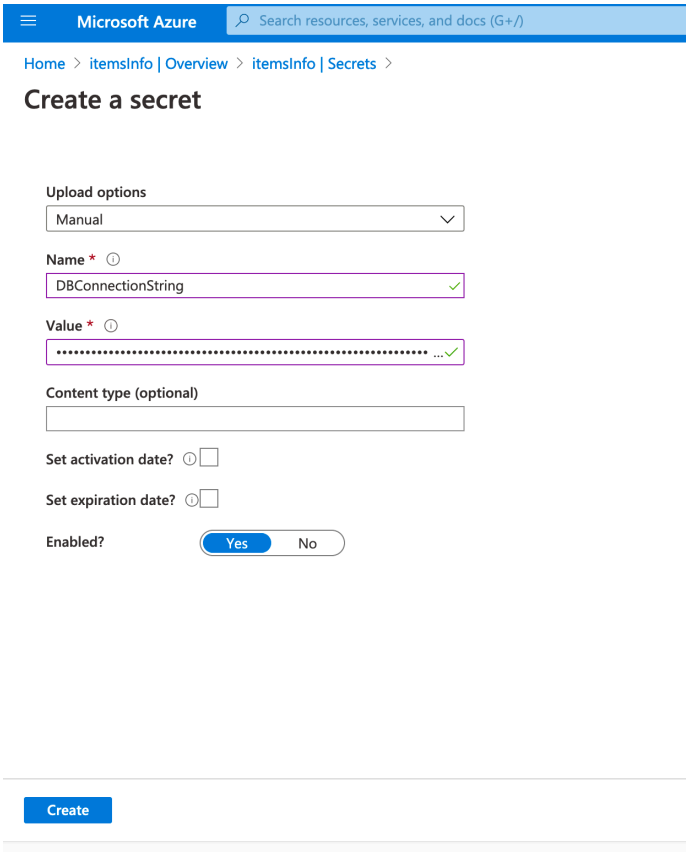


Figura 33: Detalles de la creación del secreto para la base de datos en Azure

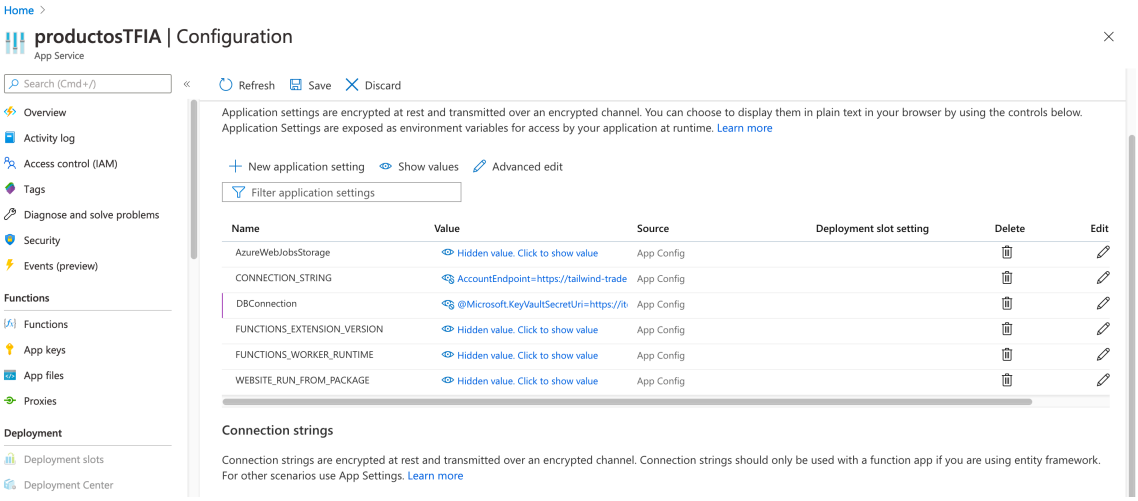


Figura 34: Detalles de la configuración utilizada en Azure Functions para la implementación del secreto de base de datos

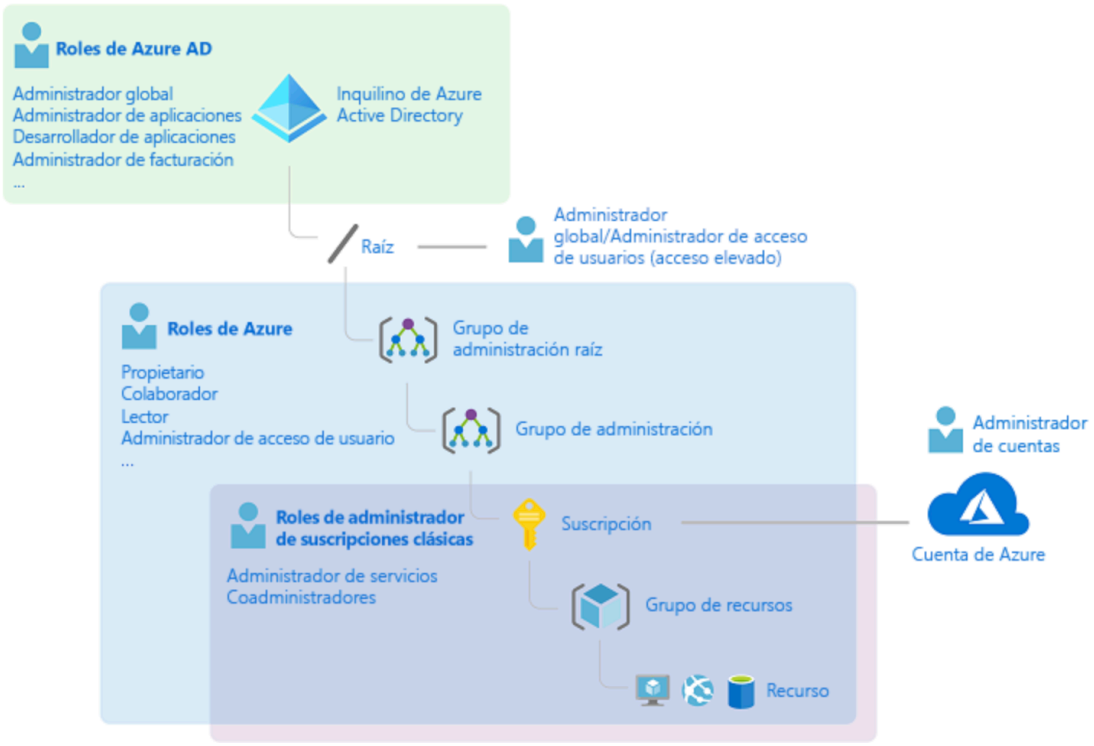


Figura 35: Detalles de las opciones de administración de permisos y usuarios en Azure

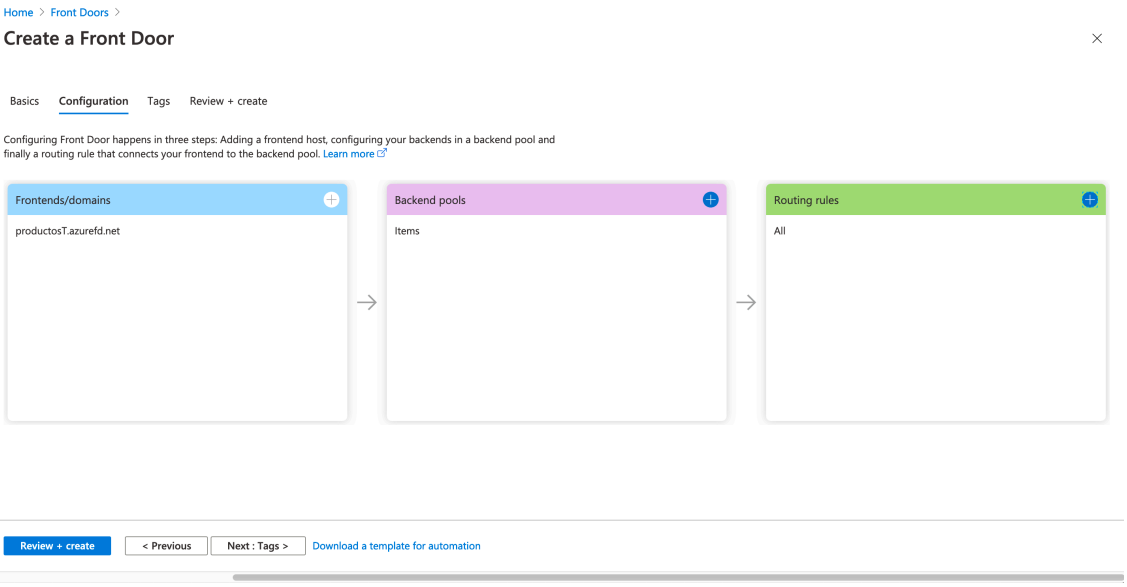


Figura 36: Detalles de la creación de Front Door para el control de XSS

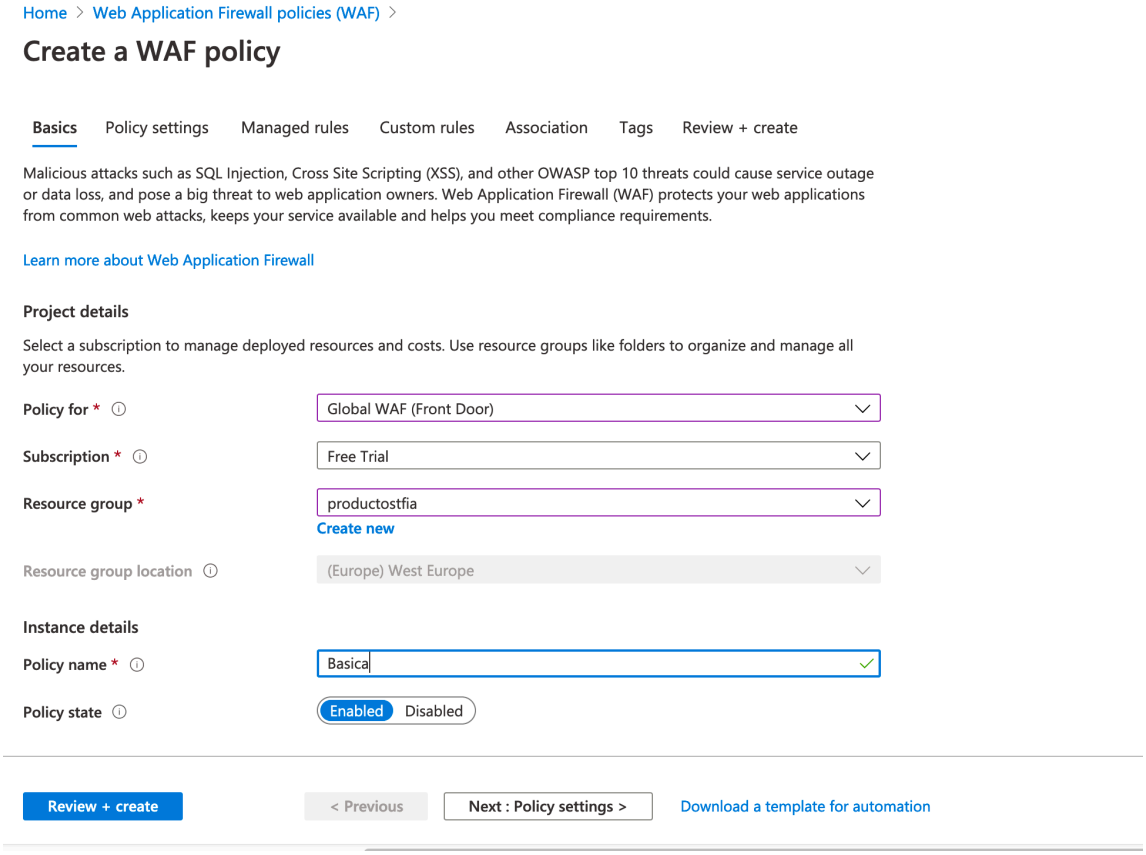


Figura 37: Detalles de la configuración de WAF para Front Door

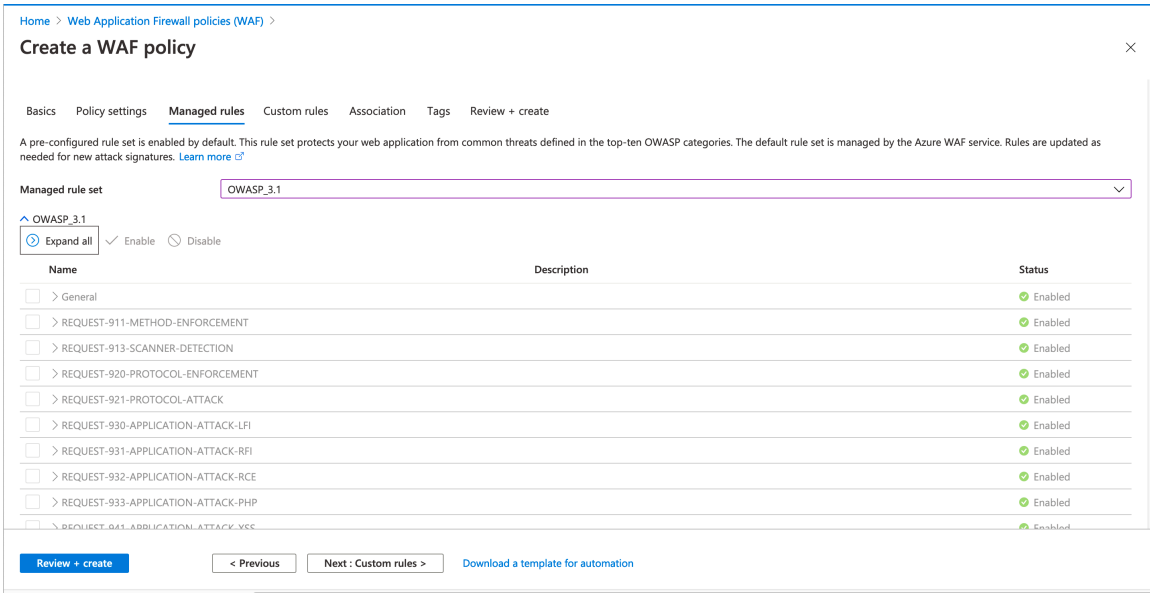


Figura 38: Detalles de las reglas implementadas en WAF para el control de XSS

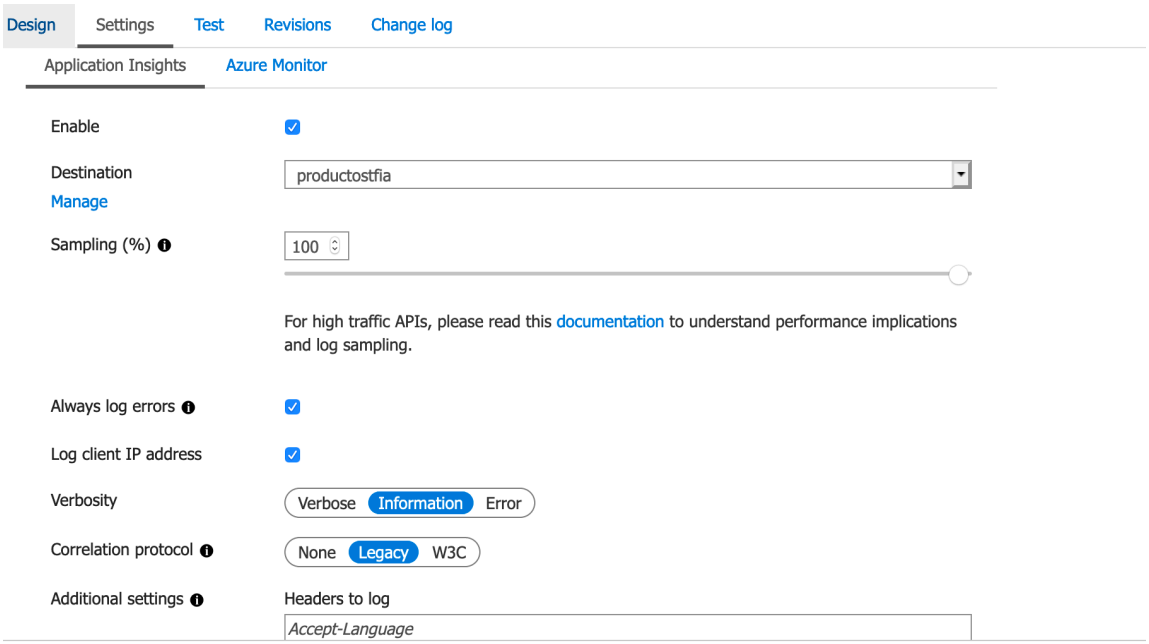


Figura 39: Detalles de la activación del servicio de monitoreo para Azure Functions

```
[
  {
    "name": "One sat gnome",
    "price": 34,
    "brand": {
      "name": "Home & Pro tools"
    },
    "stockUnits": 34,
    "id": "4e91349e-71b1-4bee-a7b7-e67707ce4d0a",
    "_rid": "vcNPAMHHOBABAAAAAAAAAA==",
    "_self": "dbs/vcNPAA==/colls/vcNPAMHHOBA=/docs/vcNPAMHHOBABAAAAAAAAAA==/",
    "_etag": "\"39018235-0000-0d00-0000-5f1c21680000\"",
    "_attachments": "attachments/",
    "_ts": 1595679080
  },
  {
    "name": "Oven 900 W",
    "price": 300,
    "brand": {
      "name": "Drills Co"
    },
    "stockUnits": 389,
    "id": "bac3a540-1a69-43f0-a53e-ebe378a09885",
    "_rid": "vcNPAMHHOBACAAAAAAAAAA==",
    "_self": "dbs/vcNPAA==/colls/vcNPAMHHOBA=/docs/vcNPAMHHOBACAAAAAAAAAA==/",
    "_etag": "\"39018335-0000-0d00-0000-5f1c21680000\"",
    "_attachments": "attachments/",
    "_ts": 1595679080
  },
  {
    "name": "Bathroom Sink Faucet Waterfall",
    "price": 175,
    "brand": {
      "name": "Home & Pro tools"
    },
    "stockUnits": 32,
    "id": "42803540-398a-4680-a149-05b7ebbf8988",
    "_rid": "vcNPAMHHOBADAAAAAAAAAA==",
    "_self": "dbs/vcNPAA==/colls/vcNPAMHHOBA=/docs/vcNPAMHHOBADAAAAAAAAAA==/",
    "_etag": "\"39018435-0000-0d00-0000-5f1c21680000\"",
    "_attachments": "attachments/",
    "_ts": 1595679080
```


Figura 40: Resultado de la implementación de la aplicación en Azure en el navagedor web

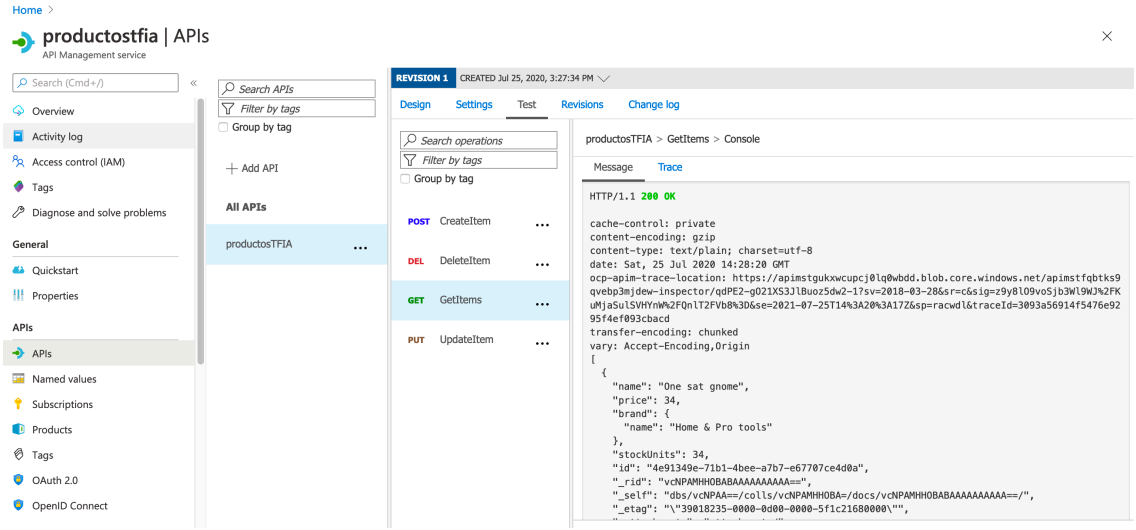


Figura 41: Resultado de la implementación de la aplicación en Azure en la consola

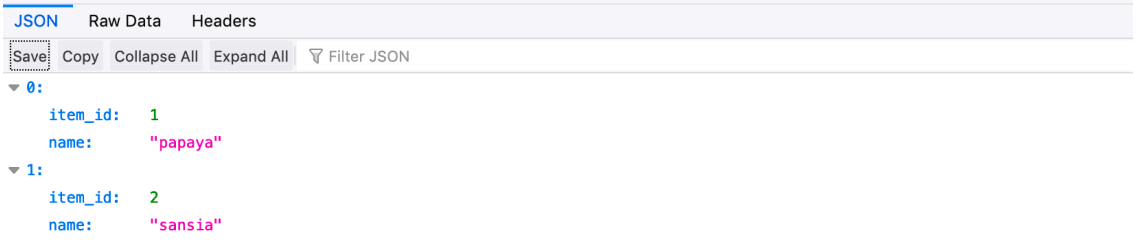


Figura 42: Resultado de la implementación de la aplicación en AWS en el navagedor web

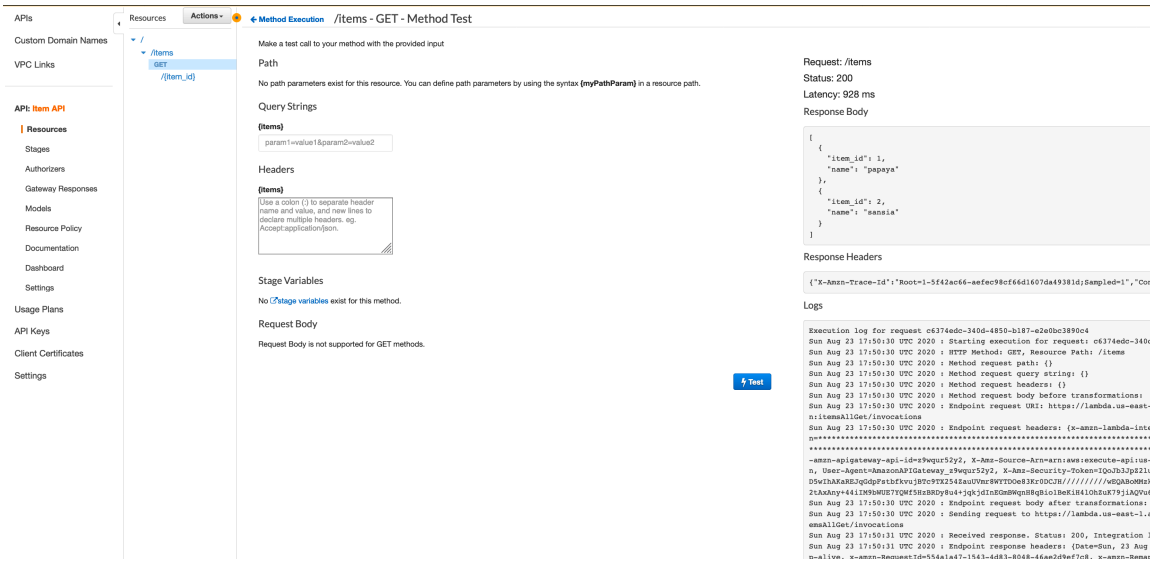


Figura 43: Resultado de la implementación de la aplicación en AWS en la consola

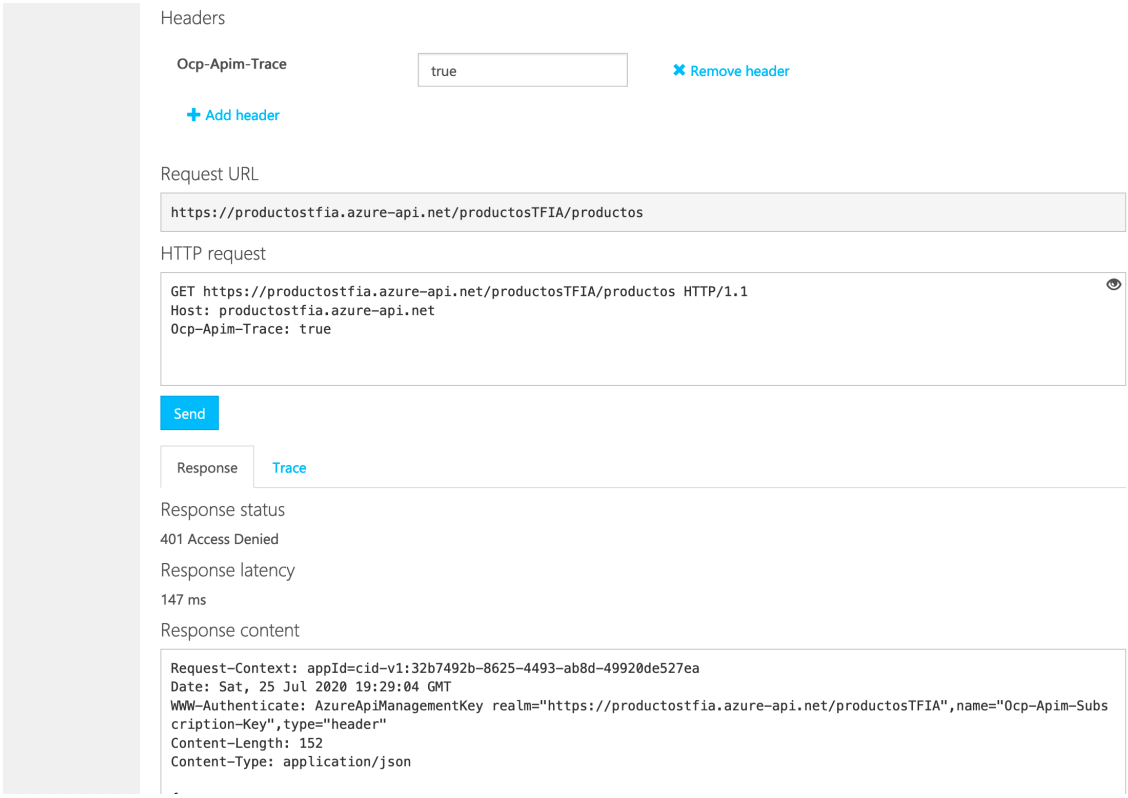


Figura 44: Resultado de tratar de accesar la aplicación sin proporcionar la información de autenticación/autorización en Azure

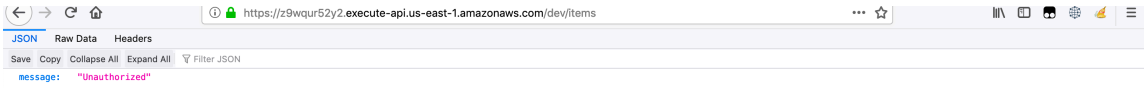


Figura 45: Resultado de tratar de accesar la aplicación sin proporcionar la información de autenticación/autorización en AWS

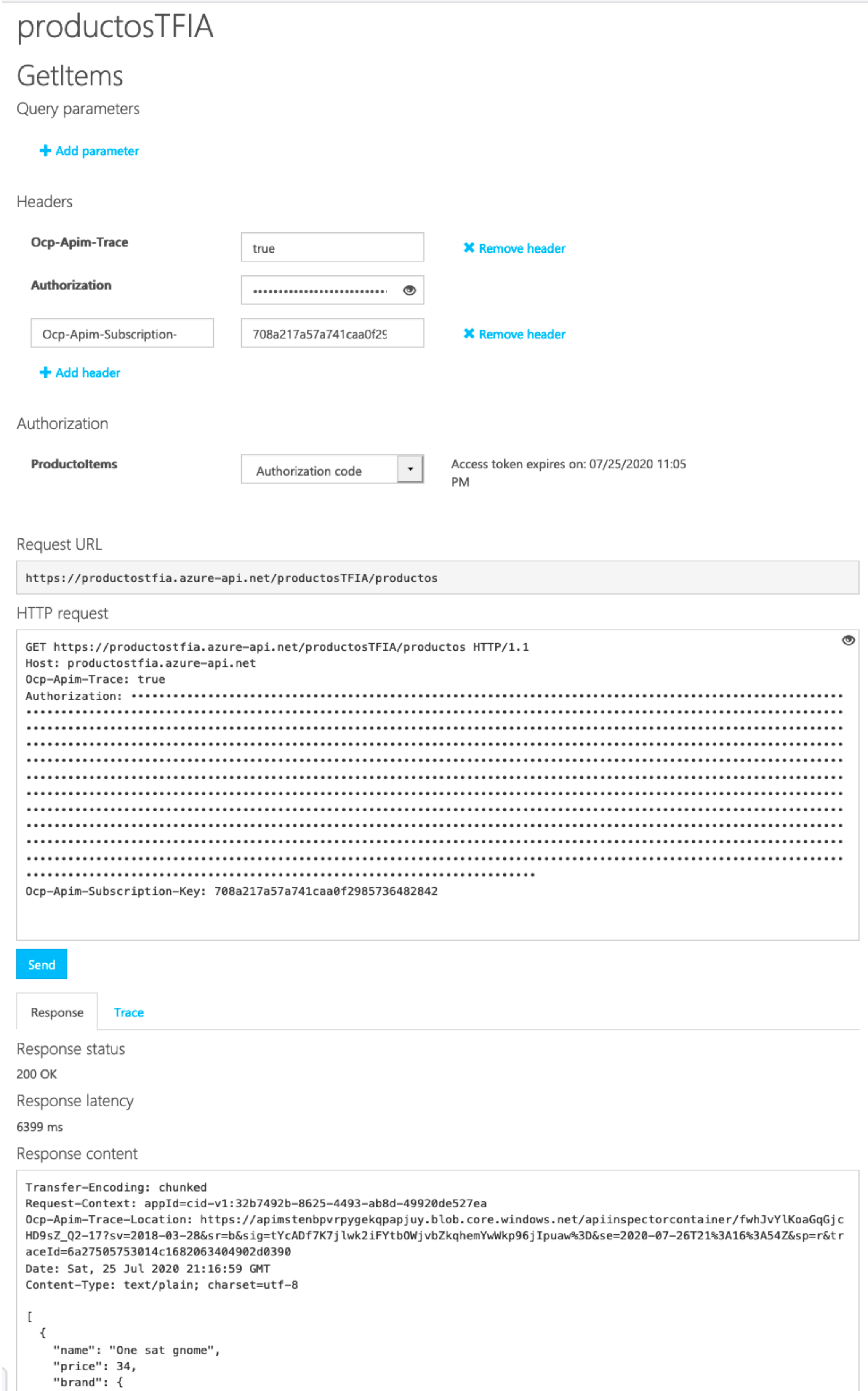

productosTFIA
GetItems
Query parameters
Add parameter
Headers
Ocp-Apim-Trace
true
Remove header
Authorization
Ocp-Apim-Subscription-
708a217a57a741caa0f29
Remove header
Add header
Authorization
ProductoItems
Authorization code
Access token expires on: 07/25/2020 11:05 PM
Request URL
https://productostfia.azure-api.net/productosTFIA/productos
HTTP request
GET https://productostfia.azure-api.net/productosTFIA/productos HTTP/1.1
Host: productostfia.azure-api.net
Ocp-Apim-Trace: true
Authorization:
Ocp-Apim-Subscription-Key: 708a217a57a741caa0f2985736482842
Send
Response
Trace
Response status
200 OK
Response latency
6399 ms
Response content
Transfer-Encoding: chunked
Request-Context: appId=cid-v1:32b7492b-8625-4493-ab8d-49920de527ea
Ocp-Apim-Trace-Location: https://apimstenbpvrpygekqpapjuy.blob.core.windows.net/apiinspectorcontainer/fwhJvYlKoaGqGjc
HD9sZ_Q2-17?sv=2018-03-28&sr=b&sig=tYcADf7K7jlwk2iFYtbOWjvbZkqhemYwWkp96jIpuaw%3D&se=2020-07-26T21%3A16%3A54Z&sp=r&tr
aceId=6a27505753014c1682063404902d0390
Date: Sat, 25 Jul 2020 21:16:59 GMT
Content-Type: text/plain; charset=utf-8
[
{
"name": "One sat gnome",
"price": 34,
"brand": {

Figura 46: Resultado de la solicitud a la aplicación al proporcionar la información requerida de acceso en Azure

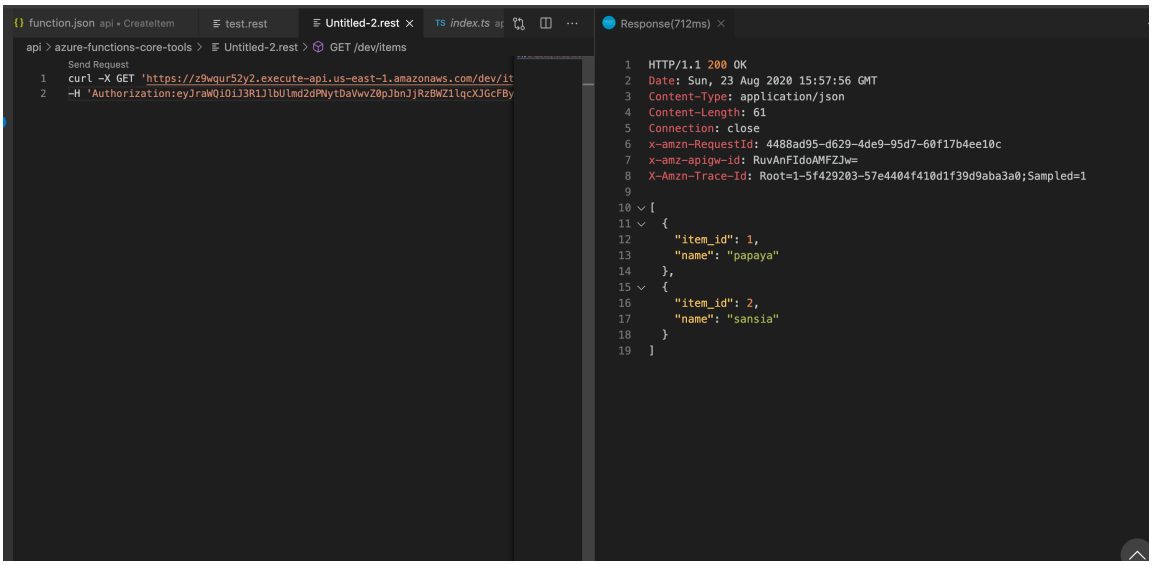


Figura 47: Resultado de la solicitud a la aplicación al proporcionar la información requerida de acceso en AWS

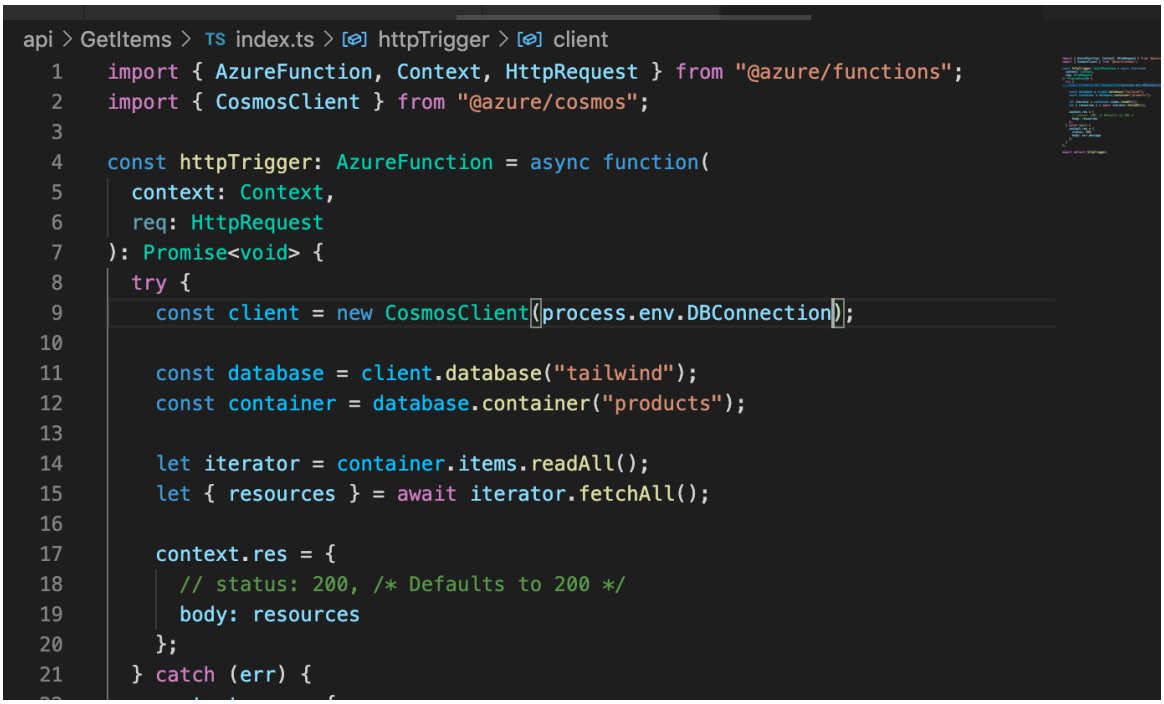


Figura 48: Detalle del cambio de código y resultado para el control de exposición de datos sensibles en Azure

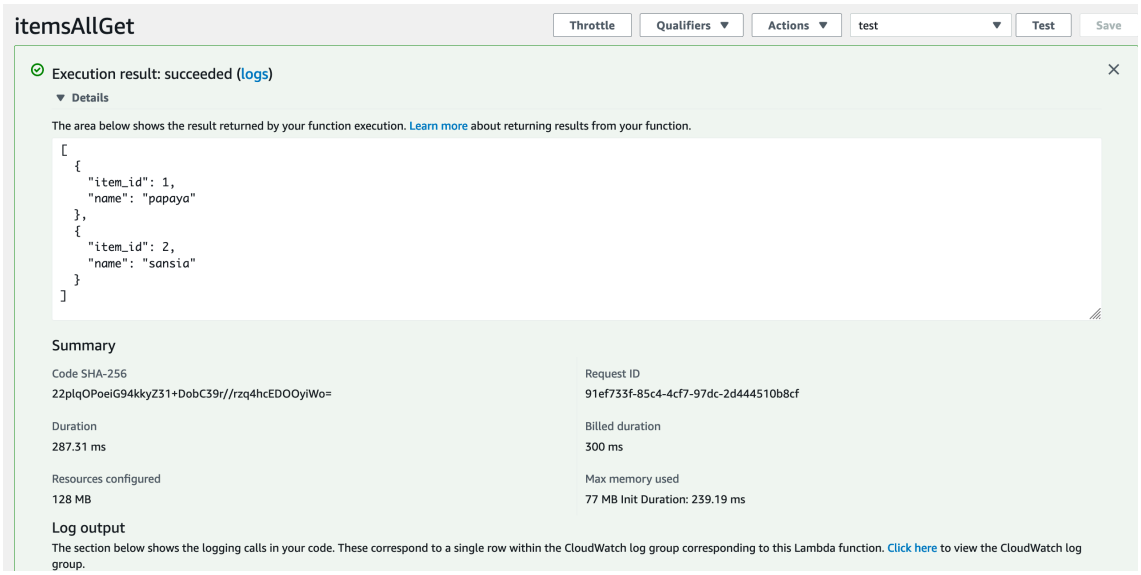


Figura 49: Detalle del cambio de código y resultado para el control de exposición de datos sensibles en AWS

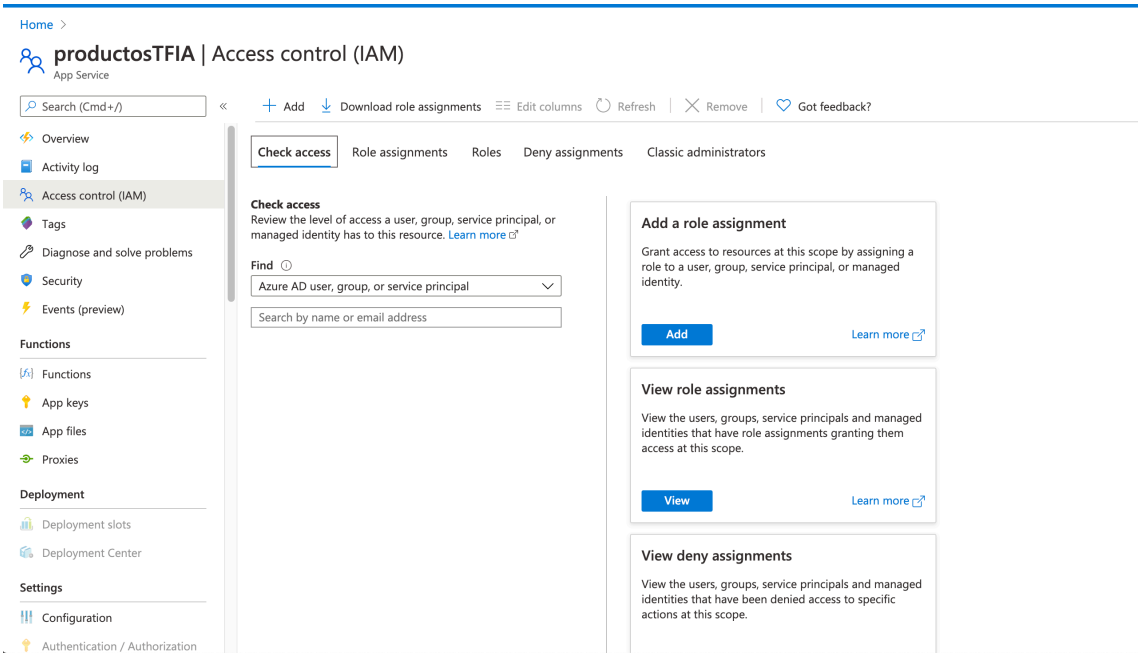


Figura 50: Detalles de los controles de acceso utilizados por Azure en la implementación de la aplicación

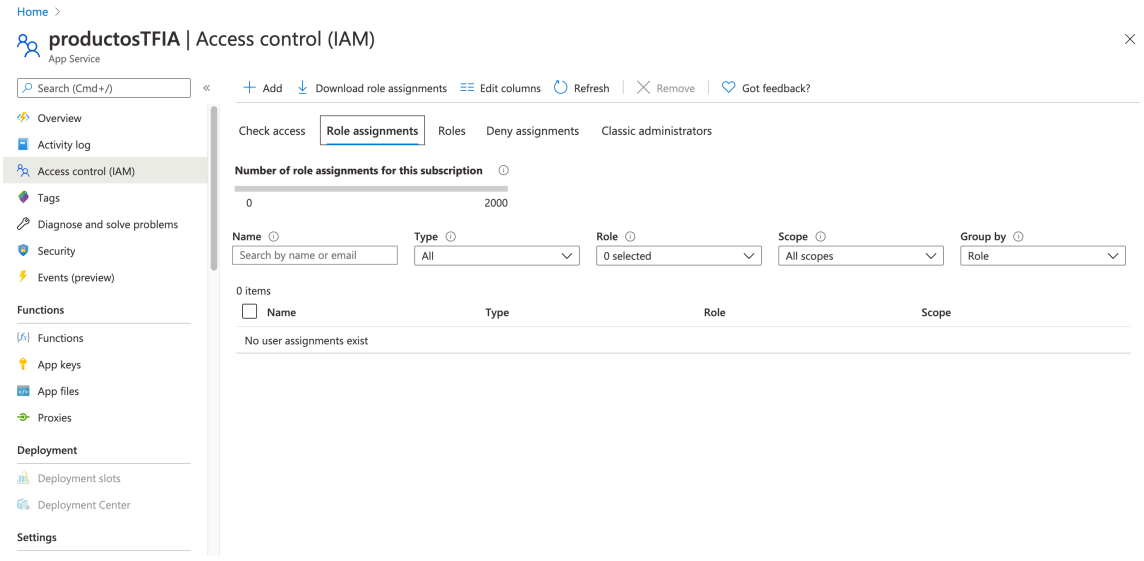


Figura 51: Detalles de los roles creados y utilizados en la implementación de Azure

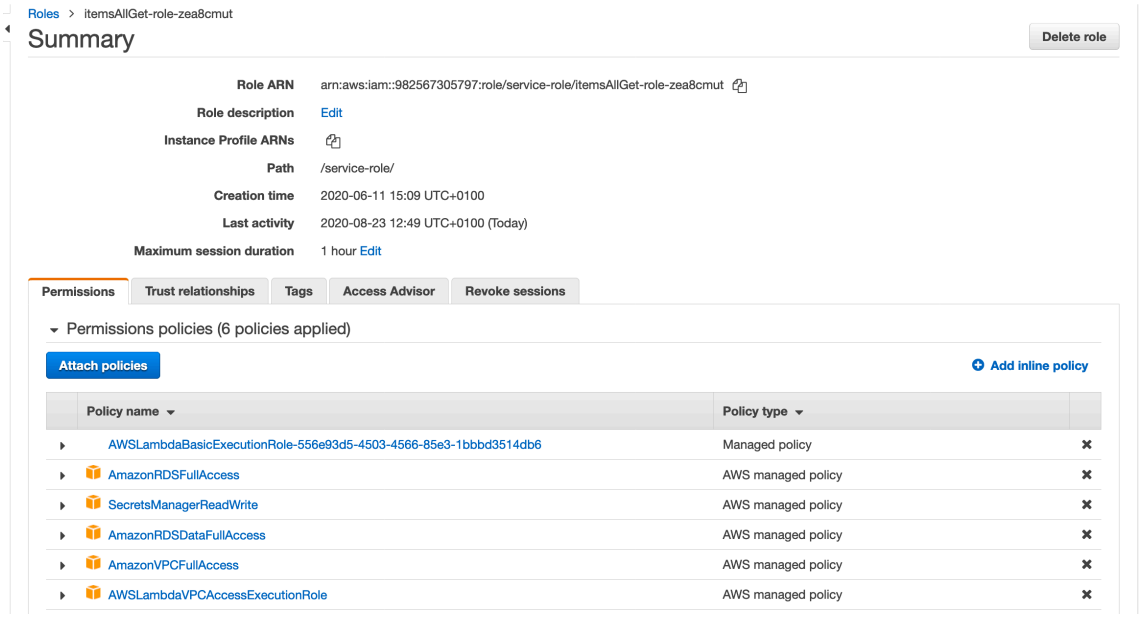


Figura 52: Permisos específicos requeridos para el funcionamiento de la aplicación en AWS

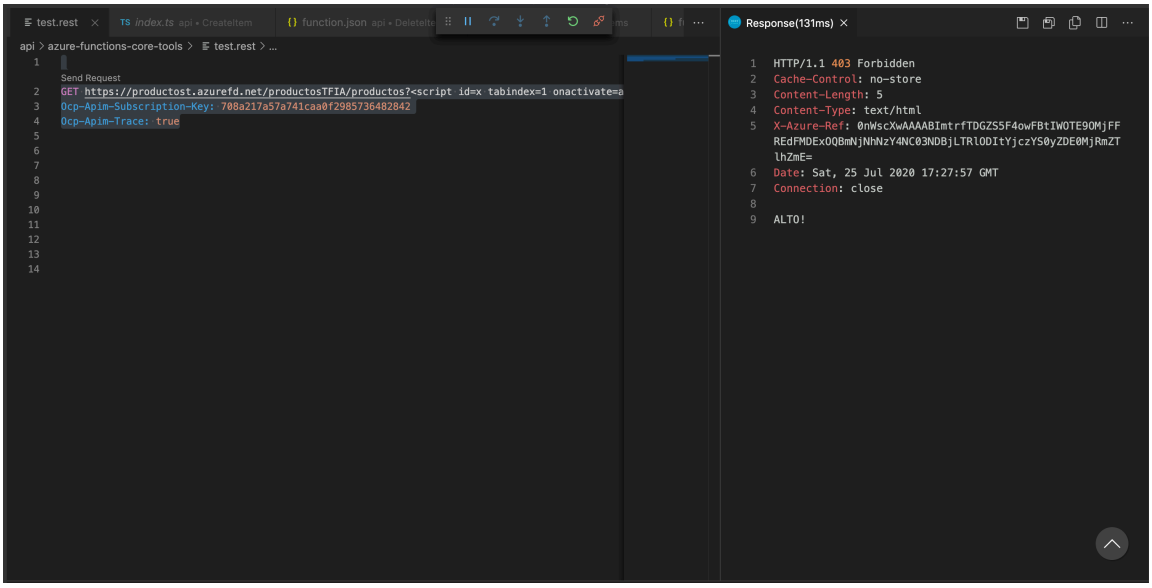


Figura 53: Resultado de una solicitud con XSS y sin control implementado en Azure

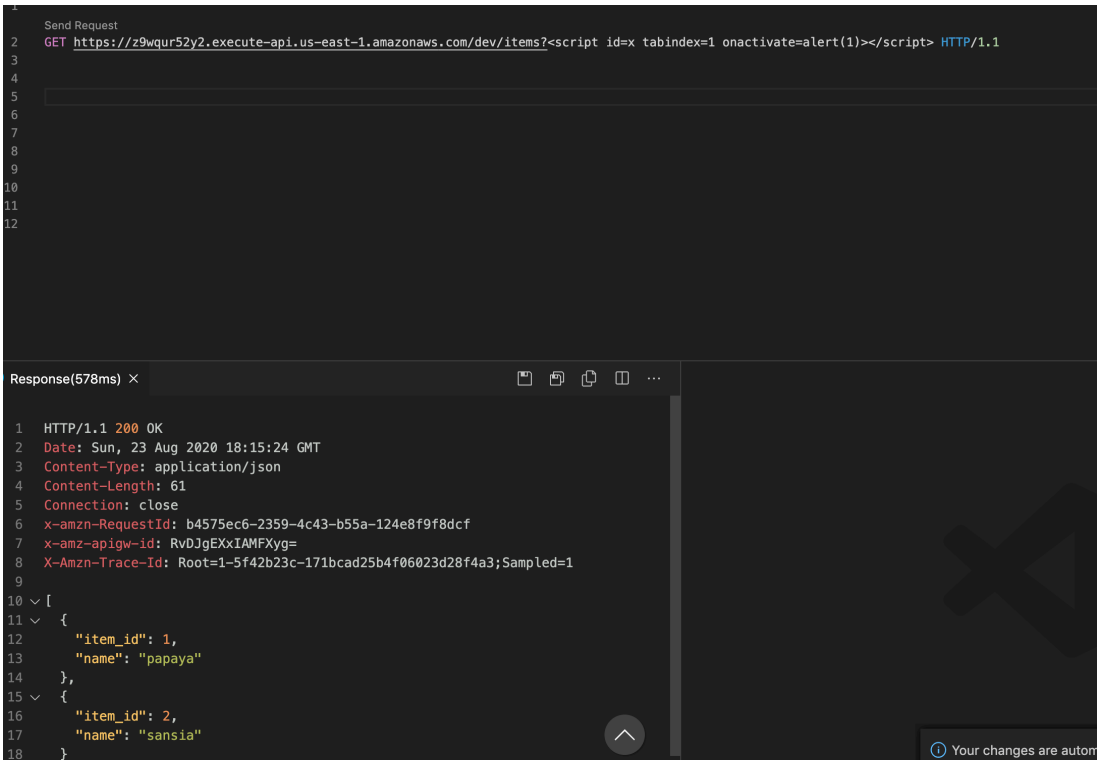


Figura 54: Resultado de una solicitud con XSS y sin control implementado en AWS

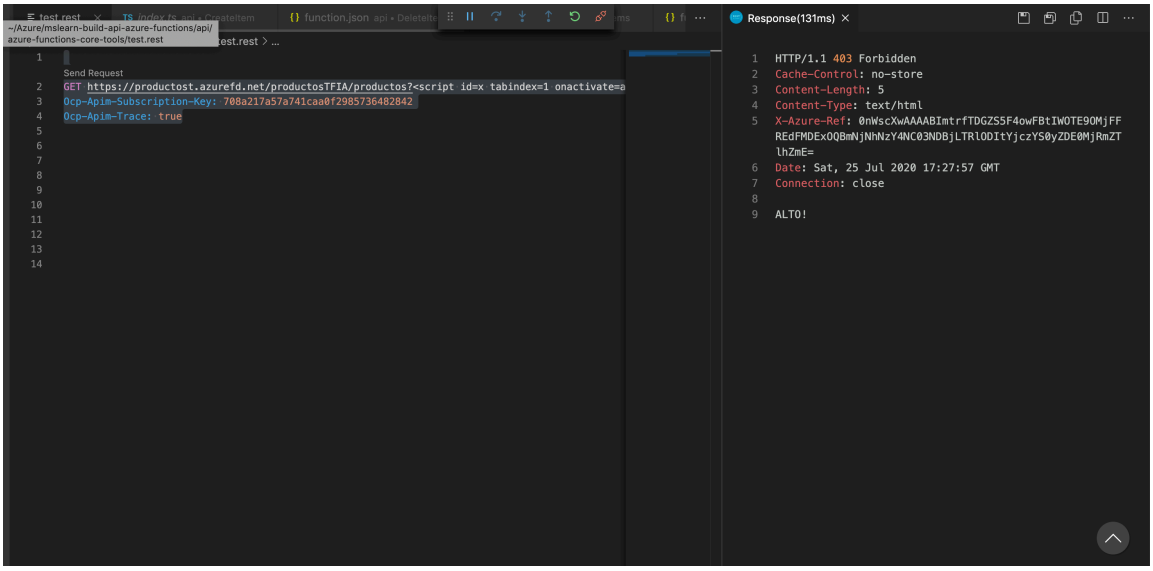


Figura 55: Resultado de una solicitud con XSS y con control implementado en Azure

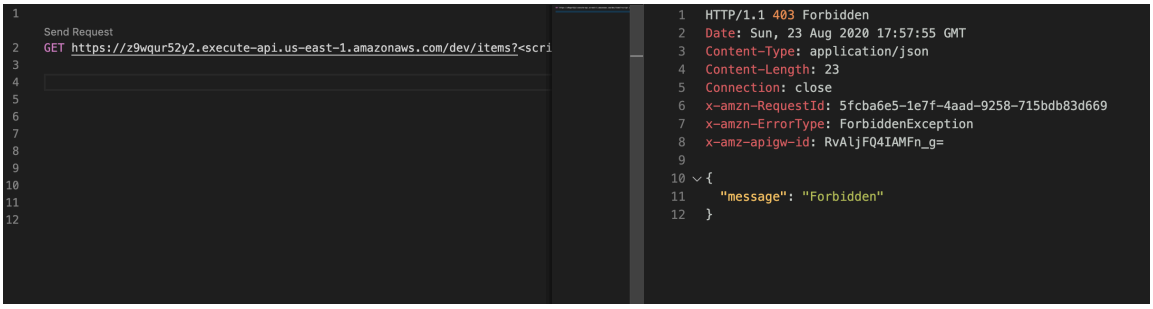


Figura 56: Resultado de una solicitud con XSS y sin control implementado en AWS

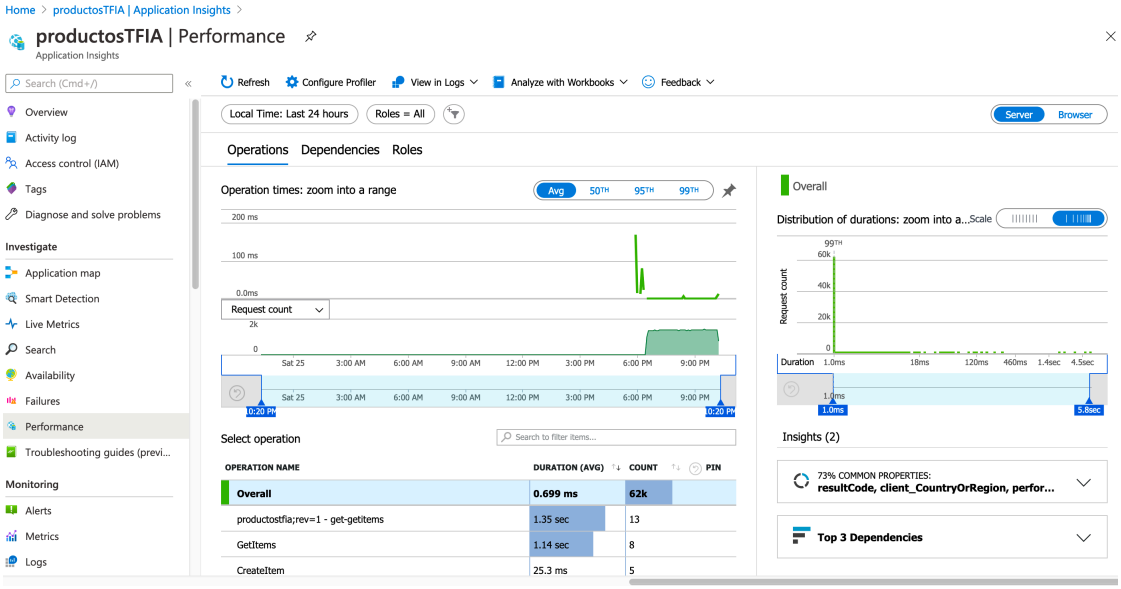


Figura 57: Datos de desempeño mostrados en el servicio de Azure de monitoreo de las función como servicio implementada

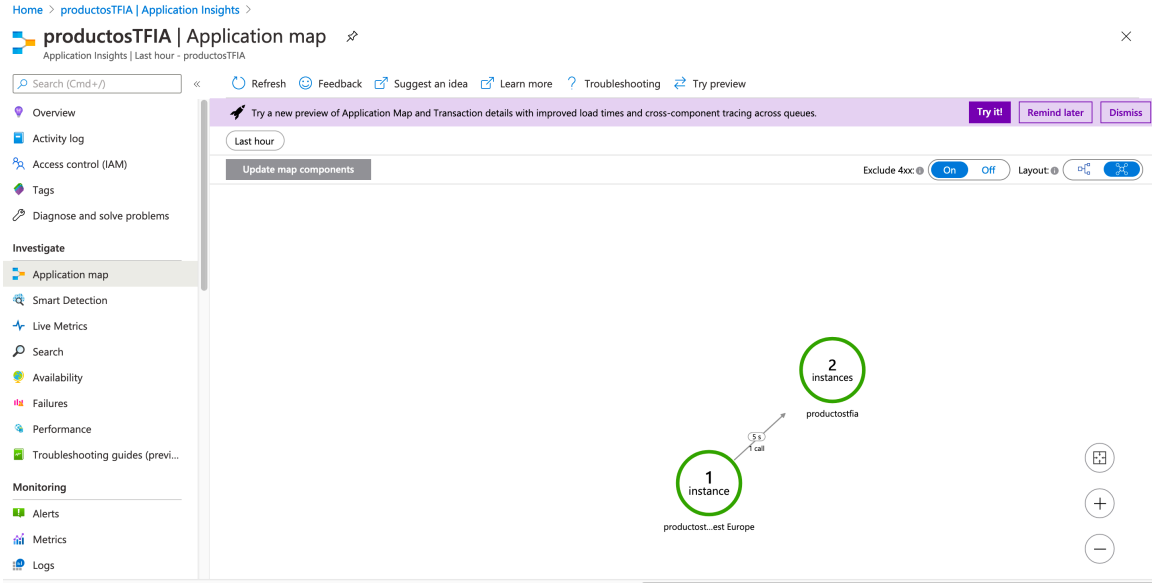


Figura 58: Datos de mapeo mostrados por el servicio de monitoreo de Azure

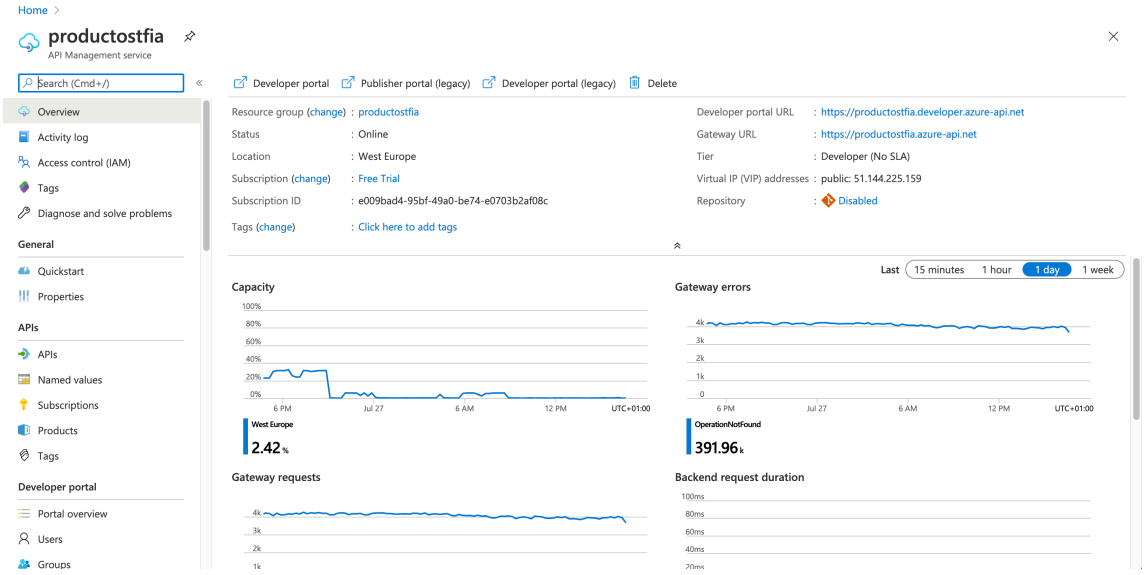


Figura 59: Datos de resumen de la actividad de la función como servicio mostrados por el servicio de monitoreo de Azure

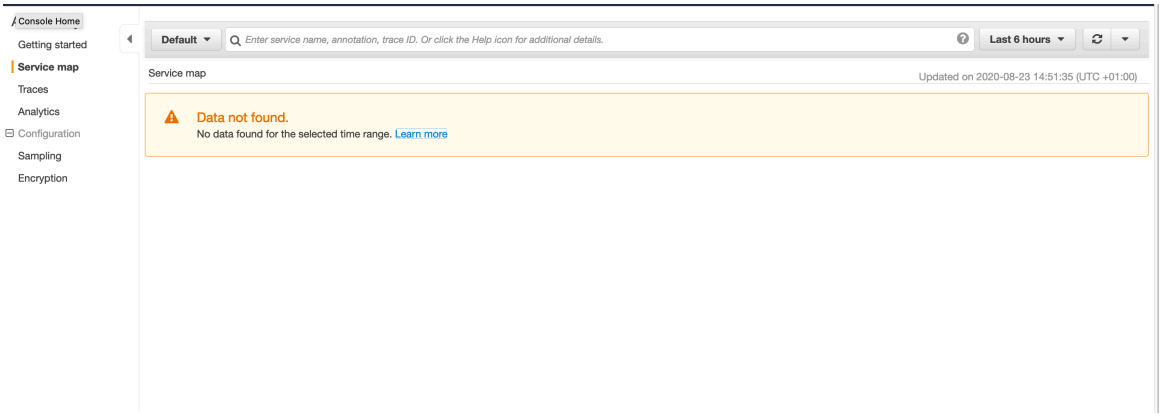


Figura 60: Muestra de la información disponible antes de la activación del control de monitoreo en AWS

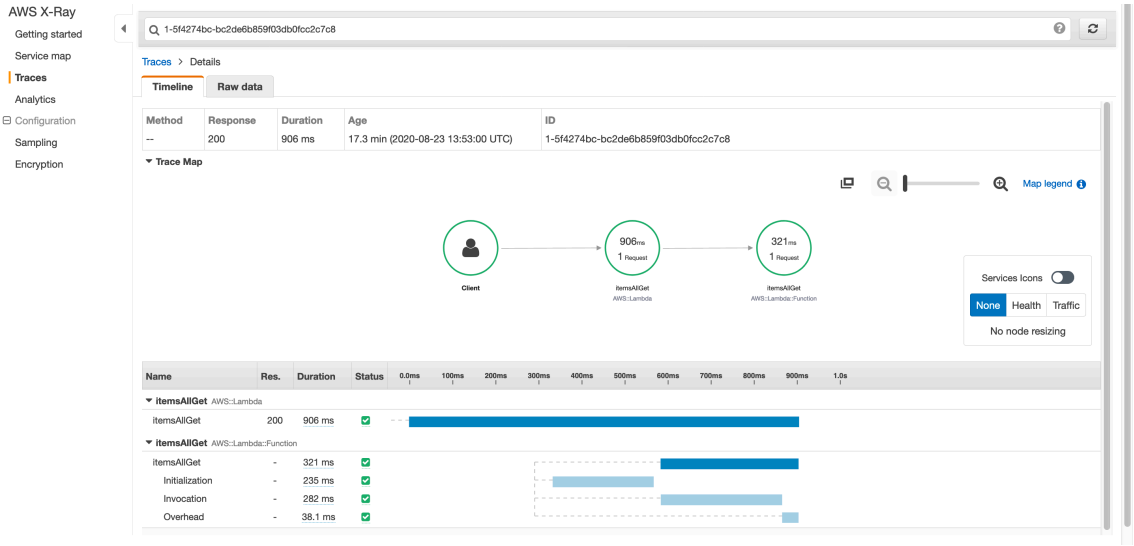


Figura 61: Información de monitoreo disponible luego de la activación del servicio de monitoreo en AWS

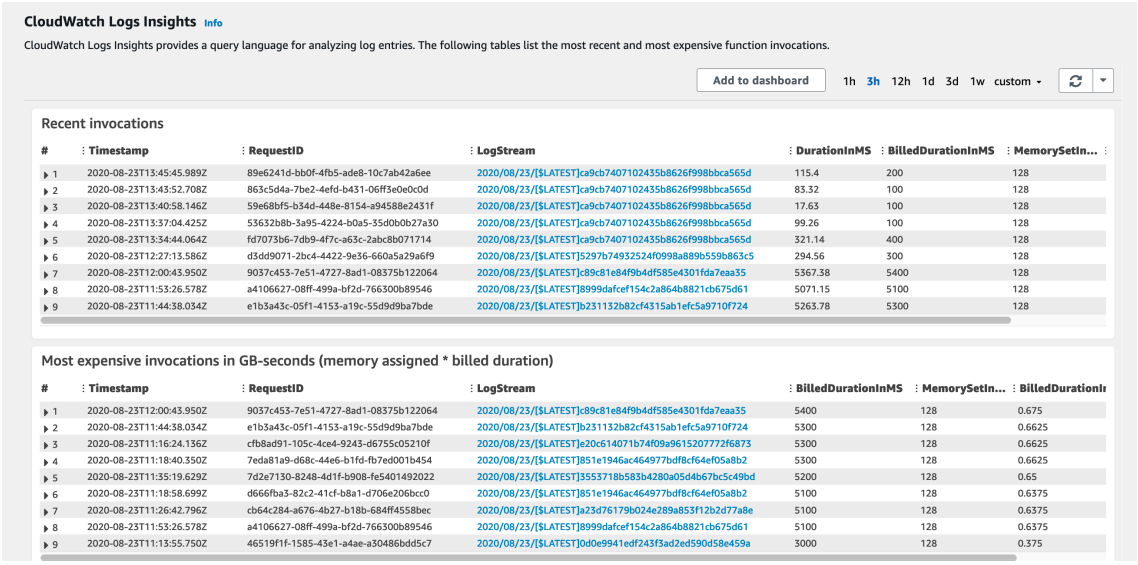


Figura 62: Información de registro adicional disponible en AWS para la función como servicio

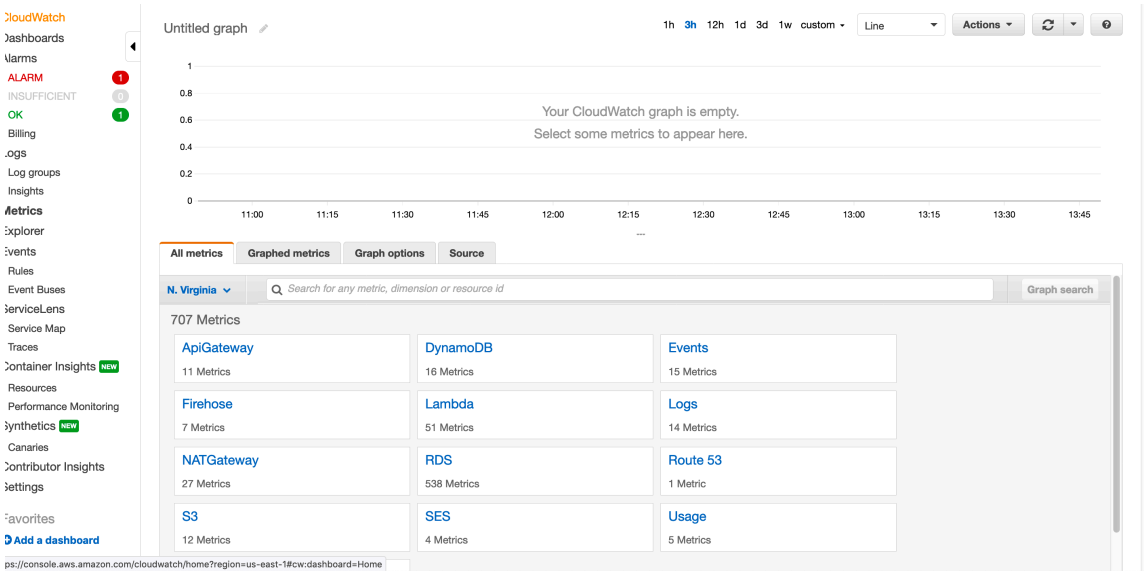


Figura 63: Información de monitoreo disponible para todos los servicios utilizados en la implementación de la aplicación y los controles en AWS