Universitat Oberta de Catalunya

uoc.edu

Autorización distribuida para aplicaciones basadas en microservicios

Alexander Granda
Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones
Sistemas de Autenticación y Autorización

Víctor Méndez Muñoz
Víctor García Font

02/06/2020


Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Autorización distribuida para aplicaciones basadas en microservicios*
Nombre del autor:	*Alexander Granda Rivera*
Nombre del consultor/a:	*Víctor Méndez Muñoz*
Nombre del PRA:	*Víctor García Font*
Fecha de entrega (mm/aaaa):	06/2020
Titulación:	*Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones*
Área del Trabajo Final:	*Sistemas de Autenticación y Autorización*
Idioma del trabajo:	*Castellano*
Palabras clave	*Autorización, microservicios*

Resumen del Trabajo:

En años recientes la arquitectura de microservicios se ha convertido en el modelo predominante para la fabricación de aplicaciones de software. Las empresas han adoptado este patrón arquitectónico gracias a su gran capacidad de adaptación y escalabilidad. Sin embargo, las organizaciones también tienen que enfrentar desafíos propios de los sistemas distribuidos, principalmente en el ámbito de la seguridad informática. OWASP en su proyecto top 10 identifica a la perdida de autenticación y control de acceso como dos de los principales riesgos a los cuales están expuestos los sistemas web.

Considerando este antecedente, el presente trabajo se enfoca en la identificación de un mecanismo de autorización que se adapte adecuadamente a las aplicaciones basadas en microservicios. Para ello se realizó una investigación sobre la arquitectura de microservicios, identificando en el proceso cuales son las principales características principal que la definen. Conjuntamente se realizó un análisis comparativo de las estrategias de autorización disponibles para este tipo de arquitectura, descubriendo que la autorización basada en tokens firmados digitalmente es el mecanismo de mayor compatibilidad con la arquitectura de microservicios.

Finalmente, mediante la implementación de una prueba de concepto sobre una aplicación web prototipo que utiliza microservicios se evaluó el funcionamiento de la estrategia de autorización identificada, con el fin de determinar su efectividad.

Abstract:

In recent years, microservices architecture has become the predominant model for the manufacture of software applications. Companies have adopted this architectural pattern thanks to its great adaptability and scalability. However, organizations also must face challenges typical of distributed systems, mainly in the field of computer security. OWASP in its top 10 project identifies the broken authentication and broken access control as two of the main risks for web applications

Considering this background, this study focuses on the identification of an authorization mechanism that fits into to microservices based applications. To achieve this goal, an investigation about microservices architecture was performed, identifying which are the main characteristics that define it.

As well, a comparative analysis of the authorization strategies available for microservices architecture was performed, discovering that authorization based on digitally signed tokens is the ideal mechanism for this kind of applications.

Finally, by the implementation of a proof of concept on a web application that uses microservices, the performance of the identified authorization strategy was evaluated, in order to determine its effectiveness.

Índice

Lista de figuras

Lista de tablas

1. Introducción

1.1 Contexto y justificación del Trabajo

La arquitectura de microservicios para aplicaciones web ha crecido en popularidad en años recientes debido a que los modelos arquitectónicos tradicionales para la fabricación de software no han logrado adaptarse a las necesidades de escalabilidad, tolerancia a fallos, pronto desarrollo y alta disponibilidad que requieren las empresas para mantener un nivel adecuado de satisfacción de sus clientes [1].

En este tipo de arquitectura cada servicio trabaja de manera independiente y se encarga de administrar sus propios recursos; sin embargo, la adopción de este patrón arquitectónico da lugar a la aparición de desafíos propios de las aplicaciones distribuidas, entre los cuales se destaca el control de acceso a los recursos.

Una inadecuada implementación en los controles de autorización durante la adopción de sistemas basados en microservicios puede ocasionar que tanto información interna de una organización, como la de terceros se vea comprometida. Estos problemas a su vez pueden originarse en el hecho de que existen varios enfoques para satisfacer los requisitos de seguridad de las aplicaciones y no todos resultan ser apropiados para sistemas distribuidos.

Si bien la migración de sistemas monolíticos elaborados con arquitecturas tradicionales hacia la arquitectura de microservicios puede representar una tarea con un considerable grado de dificultad. Cada vez son más las organizaciones que deciden acoger este patrón arquitectónico para actualizar sus sistemas, adaptándolos a las necesidades de sus clientes y aumentando su alcance publicándolos a través de la nube.

Para lograr que los usuarios se sientan cómodos utilizando este tipo de aplicaciones es necesario que los productores de software estén en la capacidad de brindarles el mayor control posible sobre su información y les garanticen niveles adecuados de seguridad. Adicionalmente, los desarrolladores de software deben procurar que las soluciones de seguridad seleccionadas se acoplen a la naturaleza de las aplicaciones, así como a los requisitos del negocio [2].

El presente trabajo final de máster pretende identificar una estrategia de autorización que permita a una aplicación web basada en microservicios administrar el control de acceso de sus recursos de una manera efectiva, respetando la autonomía de los servicios.

Para conseguir este objetivo se propone la caracterización de la arquitectura de microservicios, en conjunto con un análisis comparativo de las estrategias de autorización disponibles actualmente para este tipo de sistemas, profundizando en este análisis mediante la implementación de una prueba de concepto para el mecanismo identificado como el más adecuado y verificando su funcionamiento mediante la ejecución de pruebas que simulen escenarios reales de autorización.

1.2 Objetivos del Trabajo

El objetivo del presente trabajo de titulación es identificar una estrategia o mecanismo de autorización adecuado para aplicaciones web elaboradas utilizando la arquitectura de microservicios, para lo cual será necesario definir a esta arquitectura, identificar cuáles son sus principales características y realizar un análisis comparativo de las opciones de autorización compatibles con este tipo de sistemas.

Complementariamente, ahondar en el funcionamiento del mecanismo de autorización identificado, mediante la implementación de una prueba de concepto con el fin de evaluar su funcionamiento, permitirá confirmar o denegar los resultados teóricos.

Recapitulando, los objetivos del trabajo de fin de máster son los siguientes:

Objetivos de investigación

- Describir el estado del arte de la arquitectura de microservicios.
- Definir la arquitectura de microservicios.
- Identificar las características de la arquitectura de microservicios.
- Investigar acerca de los mecanismos de autorización existentes para aplicaciones basadas en microservicios.
- Realizar un análisis comparativo entre los diferentes enfoques de autorización para la arquitectura de microservicios.
- Identificar cual es la estrategia de autorización más adecuada a implementar en aplicaciones web elaboradas utilizando la arquitectura de microservicios.

Objetivos de implementación

- Diseñar y construir un prototipo de aplicación web basada en microservicios.
- Implementar una prueba de concepto funcional para el mecanismo de autorización identificado.
- Verificar el funcionamiento del mecanismo de autorización mediante el diseño y ejecución de casos de prueba.

1.3 Enfoque y método seguido

El presente trabajo está orientado hacia el análisis de estrategias de autorización disponibles para sistemas informáticos basados en microservicios, buscando identificar un mecanismo de autorización adecuado para aplicaciones web fabricados con esta arquitectura. Para complementar el análisis investigativo inicial se implementará una prueba de concepto que permitirá evaluar la efectividad de la estrategia de autorización seleccionada, procurando respetar las características que se identifiquen para las aplicaciones basadas en microservicios.

El proyecto se compone de dos partes:

La primera parte teórica consiste en el desarrollo de una investigación sobre la arquitectura de microservicios, esta se enfocará en contestar las siguientes preguntas de investigación.

- ¿Qué es la arquitectura de microservicios?
- ¿Cuáles son las características de la arquitectura de microservicios?
- ¿Qué estrategias de autorización se pueden implementar en esta arquitectura?
- ¿Cuál es la estrategia de autorización más adecuada para aplicaciones web basadas en esta arquitectura?

Adicionalmente, en esta etapa se establecerán las características principales de una aplicación que servirá como caso de estudio y que guiará las actividades prácticas de implementación y evaluación en la siguiente etapa del proyecto.

La segunda parte posee un enfoque práctico y permitirá ahondar en el análisis comparativo de las estrategias de autorización. En esta etapa se implementará a manera de prueba de concepto la estrategia de autorización seleccionada como la más adecuada, basándose en la aplicación definida como caso de estudio.

Durante la primera parte del proyecto centrada en la investigación y análisis de información se llevaran a cabo las siguientes actividades:

1) Recolección: se buscará en libros, revistas y artículos especializados información necesaria para poder definir la arquitectura de microservicios, identificar sus principales características, así como reconocer los esquemas de autorización que se pueden implementar en aplicaciones basadas en esta arquitectura.
2) Análisis: mediante un enfoque cuantitativo se llevará a cabo una comparación entre los esquemas de autorización identificados y su grado de compatibilidad con la arquitectura de microservicios, procurando el uso de métricas objetivas en el proceso.
3) Desenlace: a partir de los resultados del análisis comparativo previo se seleccionará la estrategia de autorización que mejor se adapte a las aplicaciones basadas en microservicios y se especificará cual va a ser el caso de estudio en el cual se evaluará este mecanismo.

En la segunda parte enfocada principalmente en la evaluación de la estrategia de autorización identificada como la más apropiada, haciendo uso de una metodología ágil de desarrollo basada en ciclos iterativos e incrementales, se llevarán a cabo las actividades de diseño e implementación de la prueba de concepto del caso de estudio.

En cada ciclo de desarrollo de llevarán a cabo las siguientes fases.

1) Definición de requerimientos: etapa en la cual se analizarán e identificarán las funcionalidades que se va a implementar durante la iteración.
2) Codificación: consiste en la implementación de las funcionalidades que forman parte del alcance de la iteración.
3) Evaluación: etapa en la que se evalúan las funcionalidades implementadas, se detectan errores y se los refina para conseguir aportaciones de valor para el producto de software que se está elaborando.
4) Finalización: etapa en la cual se da por terminada la iteración y se agregan nuevas partes funcionales al producto final.

Durante el proceso de desarrollo se dará preferencia al uso de herramientas y tecnologías de acceso gratuito y licenciamiento libre.

1.4 Planificación del Trabajo

El proyecto se compone de cuatro secciones iniciales relacionadas con la elaboración del trabajo de fin de máster cada etapa cuenta con su respectivo hito. Adicionalmente existen dos secciones que corresponden al cierre y culminación del proyecto, los hitos de estas secciones son la entrega del video con la presentación del proyecto y la defensa del trabajo final.

Durante la elaboración del cronograma de actividades, se consideraron a los domingos como días de descanso.

Debido a los problemas ocasionados por la pandemia global del virus COVID-19, fue necesario actualizar el cronograma de trabajo establecido. Viéndose afectadas principalmente la planificación correspondiente al segundo y tercer hito.

Las tareas que se desarrollarán durante el proyecto son:

	Tarea	Inicio	Fin	Duración
1	Planificación	19/02/2020	02/03/2020	
1.1	Establecer el problema a resolver	19/02/2020	20/02/2020	5 horas
1.2	Contextualizar el problema	21/02/2020	22/02/2020	4 horas
1.3	Definir los objetivos del proyecto	24/02/2020	24/02/2020	4 horas
1.4	Definir la metodología de trabajo	25/02/2020	26/02/2020	5 horas
1.5	Definir las tareas a realizar	27/02/2020	27/02/2020	3 horas
1.6	Planificar las tareas	28/02/2020	28/02/2020	4 horas
1.7	Redactar el plan de trabajo	29/02/2020	02/03/2020	8 horas
1.8	**Entregable 1: Plan de trabajo**	**03/03/2020**	**03/03/2020**	**Hito**

2	Análisis e investigación	03/03/2020	26/03/2020	
2.1	Definir la arquitectura de microservicios	03/03/2020	05/03/2020	6 horas
2.2	Describir el caso de estudio	06/03/2020	06/03/2020	4 horas
2.3	**Estrategias de autorización para microservicios**	**07/03/2020**	**20/03/2020**	
2.3.1	Investigar las estrategias de autorización para aplicaciones basadas en microservicios	07/03/2020	12/03/2020	8 horas
2.3.2	Realizar un análisis comparativo de las estrategias investigadas	13/03/2020	17/03/2020	8 horas
2.3.3	Identificar la estrategia de autorización adecuada para el caso de estudio	18/03/2020	20/03/2020	4 horas
2.4	Redactar la memoria de la investigación	27/03/2020	04/04/2020	16 horas
2.4	**Entregable 2: Avance de la parte teórica del TFM**	**04/04/2020**	**05/04/2020**	**Hito**
3	Diseño e implementación	06/04/2020	10/05/2020	
3.1	**Prototipo de aplicación basada en microservicios**	**06/04/2020**	**18/04/2020**	
3.1.1	Análisis e identificación de los recursos necesarios	06/04/2020	07/04/2020	3 horas
3.1.2	Elaborar el diseño de la aplicación	08/04/2020	10/04/2020	6 horas
3.1.2	Configurar el ambiente de desarrollo	11/04/2020	11/04/2020	3 horas
3.1.3	Implementar el prototipo de aplicación	13/04/2020	18/04/2020	15 horas
3.2	**Mecanismo de autorización**	**20/04/2020**	**04/05/2020**	
3.2.1	Analizar los requisitos de implementación	20/04/2020	21/04/2020	3 horas
3.2.2	Elaborar el diseño de la prueba de concepto	22/04/2020	25/04/2020	5 horas
3.2.3	Configurar el ambiente de desarrollo	27/04/2020	28/04/2020	4 horas
3.2.4	Implementar la prueba de concepto	29/04/2020	04/05/2020	15 horas
3.3	**Verificación del proyecto**	05/05/2020	07/05/2020	
3.3.1	Diseñar los casos de prueba	05/05/2020	05/05/2020	3 horas
3.3.2	Ejecutar las pruebas de autorización	06/05/2020	07/05/2020	6 horas
3.4	Redactar la memoria de la implementación	08/05/2020	09/05/2020	12 horas
3.5	**Entregable 3: Avance de la parte práctica del TFM**	**10/05/2020**	**10/05/2020**	**Hito**
4	Presentación y defensa del TFM	11/05/2020	15/06/2020	
4.1	**Memoria del TFM**	11/05/2020	02/06/2020	
4.1.1	Analizar los resultados de las pruebas	11/05/2020	13/05/2020	6 horas
4.1.1	Redactar las conclusiones sobre los resultados obtenidos	14/05/2020	20/05/2020	4 horas
4.1.2	Elaborar la memoria final	21/05/2020	01/06/2020	20 horas
4.1.3	**Entregable 4: Memoria final del TFM**	**02/06/2020**	**02/06/2020**	**Hito**
4.2	**Presentación y video**	02/06/2020	09/06/2020	
4.2.1	Diseñar la presentación del TFM	02/06/2020	03/06/2020	4 horas
4.2.2	Elaborar el video de presentación	04/06/2020	08/06/2020	6 horas

4.2.3	**Entregable 5: Presentación en video del TFM**	09/06/2020	09/06/2020	**Hito**
4.3	**Defensa del TFM**	15/06/2020	15/06/2020	**Hito**

Tabla 1: Planificación de actividades

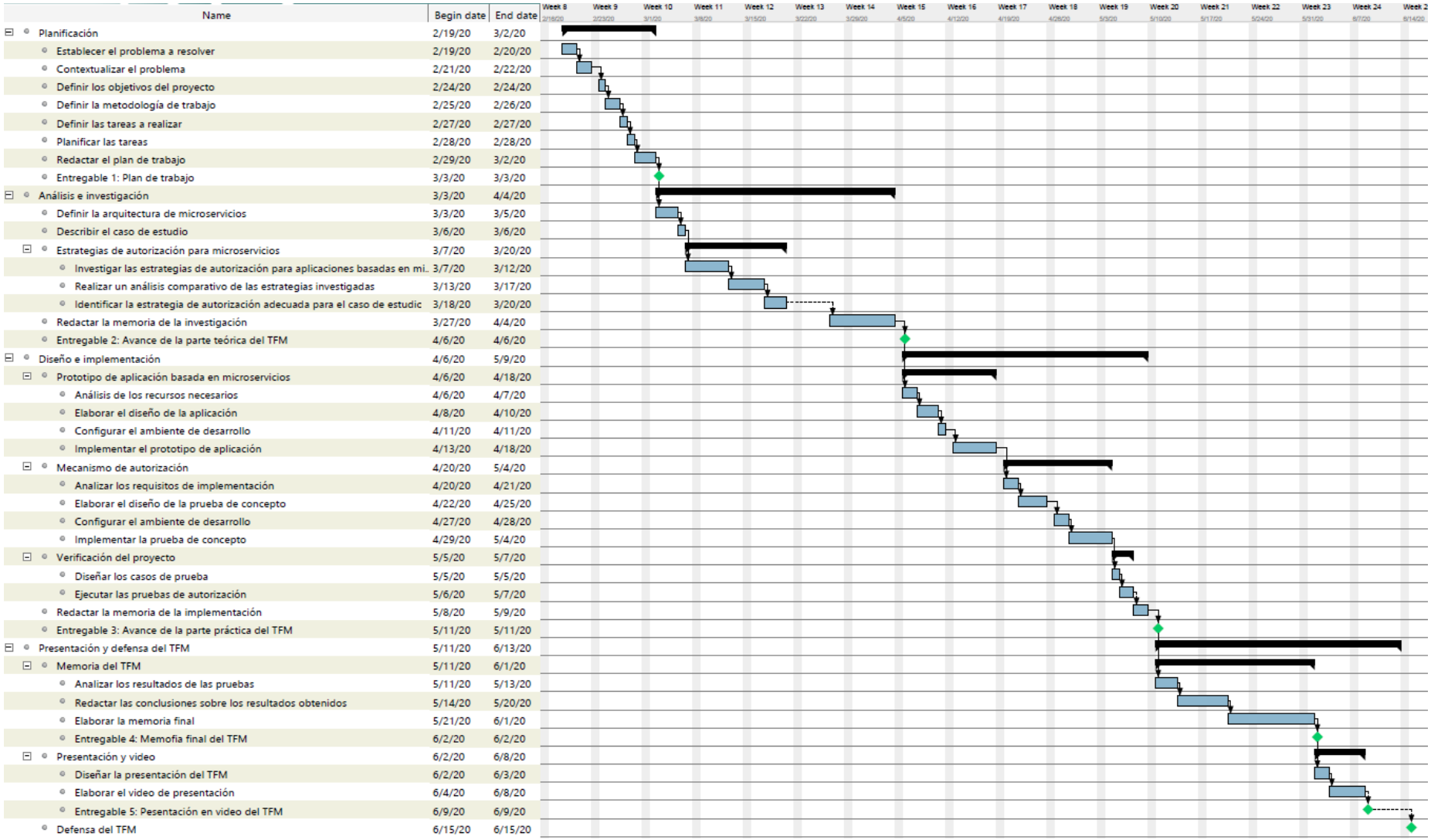

Ilustración 1: Diagrama de Gantt de la planificación de actividades

1.5 Estado del arte

Desde su aparición en la década de 1980, la arquitectura de software se ha convertido en una disciplina madura, logrando trascender desde el ámbito de la investigación académica hasta establecerse en la industria, convirtiéndose de este modo en un elemento esencial para la construcción de software [3].

Los primeros patrones para la elaboración de software estaban basados en el paradigma de la orientación a objetos y ayudaron a sistematizar la fabricación de software, un conocido ejemplo de estos patrones es el Modelo Vista Controlador (MVC). Posteriormente debido a la necesidad de una mejor gestión en el mantenimiento y evolución de los sistemas de software surgieron los conceptos de separación de responsabilidades e ingeniería de software basada en componentes, dando origen a la computación orientada a servicios (SOC). La sofisticación de este paradigma dio lugar a la arquitectura orientada a servicios (SOA) permitiendo la elaboración de aplicaciones distribuidas mediante la colaboración entre servicios. La arquitectura de microservicios es a su vez un refinamiento de SOA, en la que una aplicación está compuesta de pequeños servicios, donde cada uno de ellos funciona de manera independiente, ejecuta sus propios procesos y se comunica a través de mecanismos ligeros [3].

Acorde el ritmo de evolución de los modelos arquitectónicos, las organizaciones han logrado adaptar sus sistemas tradicionalmente monolíticos a sistemas distribuidos utilizando microservicios desplegados en el Internet, incluso sin ser necesaria la intervención humana, logrando adaptarse a los requisitos de sus clientes y proveedores en un mercado que se encuentra en constante evolución [4]. Una de las principales motivaciones para realizar este tipo de migraciones es el hecho de que las aplicaciones monolíticas con el paso del tiempo y el incremento de funcionalidad presentan serios desafíos en términos de mantenibilidad, escalabilidad y disponibilidad [5].

A pesar de los beneficios que brindan las aplicaciones basadas en microservicios, los equipos de desarrollo también deben considerar los desafíos que se presentan en este tipo de sistemas distribuidos, especialmente en términos de seguridad. La OWASP en su proyecto top 10, que provee un consenso sobre los riesgos de seguridad más críticos para las aplicaciones web indica que la perdida autenticación y control de acceso son dos de las principales vulnerabilidades en las aplicaciones actuales [6].

El alto flujo de información que circula por el Internet despertó el interés de un grupo de personas dedicadas a actividades poco éticas e ilícitas, entre estas actividades se pueden encontrar el robo de información, la suplantación de identidades, entre otras; obligando a las organizaciones a tomar más en serio la seguridad de sus sistemas informáticos. Debido a estas razones los conceptos de autenticación y autorización han adquirido una mayor relevancia [7].

Esta situación también ha obligado a los profesionales dedicados al desarrollo de software a innovar constantemente en los mecanismos de seguridad que se implementan los sistemas de información. Lamentablemente, es usual que los avances tecnológicos, los mecanismos de prevención y las amenazas de

seguridad se desarrollen de manera dispar, provocando que las aplicaciones no siempre puedan contar con resguardos adecuados para combatir las amenazas de seguridad más sofisticadas.

A causa de este inconveniente las aplicaciones basadas en microservicios presentan una amplia diversidad en estrategias de seguridad, especialmente en el ámbito del control de acceso a recursos mediante la autorización de entidades y puede que no todas sean adecuadas para para este tipo de arquitectura.

Uno de los primeros enfoques utilizados para asegurar los sistemas basados en microservicios fue la replicación de los mecanismos de autorización de las aplicaciones monolíticas tradicionales [8]. Esto ayudó a evidenciar que las estrategias de autorización basadas en el almacenamiento de las sesiones de usuarios en ambientes distribuidos requieren replicar una misma sesión a través de todos los componentes del sistema o a su vez hacer uso de una sesión del usuario centralizada, lo que ocasionaría un mayor consumo de recursos, limitando las bondades ofrecidas por la arquitectura de microservicios.

Otro esquema de seguridad aplicado a este tipo de arquitectura se basa en el uso de tokens de sesión que serán evaluados independientemente en cada servicio para la determinar la identidad de un usuario y a que recursos puede acceder [9]. Este es el enfoque mayormente adoptado en la industria ya que permite mantener un apropiado nivel de seguridad sin afectar el correcto funcionamiento de las aplicaciones basadas en microservicios [8].

Tomando en cuenta los trabajos de investigación en el área de la autorización de sistemas informáticos. El presente trabajo de fin de máster busca identificar un esquema de autorización adecuado para aplicaciones web basadas en microservicios, que permita controlar el acceso de los usuarios a los recursos de los diferentes servicios sin sacrificar la autonomía de estos.

1.6 Breve sumario de productos obtenidos

El presente trabajo de fin de máster está dividido en etapas, al finalizar cada etapa se contará con un entregable. Al culminar el proyecto el producto final estará constituido por la combinación de estos entregables.

- Entregable 1: el primer entregable abarca el plan de trabajo del proyecto, que está formado por la definición del ámbito, los objetivos que se planean alcanzar, la metodología que se va a utilizar y el cronograma de actividades a realizarse.
- Entregable 2: este entregable está relacionado con la parte teórica del proyecto, se compone de la definición de la arquitectura de microservicios, el análisis comparativo de las opciones de autorización aplicables a esta arquitectura y la identificación de un mecanismo de autorización adecuado para un sistema web basado en microservicios.
- Entregable 3: corresponde a la parte práctica del proyecto y abarca el diseño e implementación de una prueba de concepto para verificar el correcto funcionamiento del mecanismo de autorización previamente identificado.
- Entregable 4: representa la recopilación de los entregables previos, es la memoria del proyecto donde adicionalmente se documentarán el análisis de los resultados, las conclusiones del proyecto y los futuros caminos que se puedan tomar a partir del presente trabajo de fin de máster.
- Entregable 5: consiste en la elaboración de una presentación del proyecto, así como una demostración de la prueba de concepto implementada durante el desarrollo de este.

2. Investigación

En este capítulo se contestarán las preguntas de investigación planteadas con el fin de identificar una estrategia de autorización apropiada para aplicaciones basadas en microservicios, empezando por la determinar de una definición para este patrón arquitectónico de desarrollo de software.

2.1 Arquitectura de microservicios

En la literatura e investigaciones relacionadas con la arquitectura de microservicios no existe un consenso respecto a una definición formal para este término, motivo por el cual se han analizado algunas definiciones de diferentes autores para posteriormente elaborar una definición propia de este concepto.

2.1.1 Definiciones de investigadores

Para Martin Fowler y James Lewis [10]: “La arquitectura de microservicios es un enfoque para desarrollar una aplicación como un conjunto de pequeños servicios, donde cada uno de ellos ejecuta en sus propios procesos y se comunica mediante mecanismos ligeros, a menudo una API HTTP. Estos servicios se implementan de forma independiente”.

Según Dragoni et. Al [3]: la arquitectura de microservicios puede verse como una técnica para desarrollar aplicaciones de software que, heredando los principios y conceptos de la arquitectura orientada a servicios, permite estructurar aplicaciones basadas en una colección de servicios de software muy pequeños desacoplados entre sí. También puede verse como un nuevo paradigma para la programación de aplicaciones mediante la composición de pequeños servicios, cada uno de los cuales ejecuta sus propios procesos y se comunica a través de mecanismos ligeros.

Según Christudas [2]: La arquitectura de microservicios se puede definir como una forma de diseñar aplicaciones de software con el objetivo de lograr entregas agiles, implementación flexible y escalabilidad precisa, enfocándose en la elaboración de componentes pequeños pero funcionales.

Según Hilbrich [11]: La arquitectura de microservicios es un estilo arquitectónico, donde los requisitos funcionales del sistema se encapsulan en servicios que brindan la funcionalidad a los usuarios finales y no tienen dependencias de otros componentes (servicios) del sistema.

Para Hannousse y Yahiouche [12]: Es nuevo estilo arquitectónico que permite construir sistemas de software compuestos de servicios livianos que realizan funciones muy específicas y coherentes. Los microservicios son la base para esta arquitectura, un microservicio es una unidad de software que se puede crear, inicializar, duplicar y destruir independientemente de otros microservicios del mismo sistema. Además, los microservicios se pueden desplegar en plataformas de ejecución heterogéneas a través de una red.

El usuario enabled solutions [13] menciona en su artículo web que: Los microservicios comenzaron como un estilo arquitectónico mediante el cual se desarrolla un área de dominio específica como un conjunto de pequeños servicios, cada uno se comunican entre sí a través de interfaces bien definidas denominadas API.

En el artículo también se menciona que la aparición de los microservicios permitió que las organizaciones puedan enfocarse más en la satisfacción de sus clientes aprovechándose de las cualidades de autonomía y tolerancia a fallos de este paradigma, agilizando el ciclo de vida del software y a su vez incrementado las capacidades de innovación.

2.1.2 Definición propia

En base a las definiciones previamente expuestas se pudo elaborar una definición personal para la arquitectura de microservicios.

Es un estilo arquitectónico para elaborar sistemas de software compuestos por un conjunto de pequeños servicios autónomos y desacoplados entre sí, que se comunican con el resto de los componentes del sistema mediante mecanismos ligeros y bien definidos; esta arquitectura motiva las prácticas de implementación flexible, entregas ágiles y escalabilidad precisa.

2.2 Características de la arquitectura de microservicios

En contraste con el apartado anterior varios autores como: Pahl y Jamshidi [14], Amaral et. Al [15], López y Maya [16] y Djernæs [17]; concuerdan en que las características que mejor describen a la arquitectura de microservicios son las presentadas por Lewis y Fowler [10], en su artículo web “Microservicios, una definición para un nuevo término arquitectónico”. Estas características se describen a continuación.

2.2.1 Componentización a través de servicios

Los componentes en un sistema exponen su funcionalidad a través de servicios; donde, cada servicio representa una pequeña unidad de software autónoma independiente, actualizable y reemplazable. Esté principio ayuda a disminuir el acoplamiento entre los elementos que componen un sistema informático.

2.2.2 Gestión de productos, no proyectos

Un microservicio no se entrega a un departamento de mantenimiento cuando se considera implementado, sino que es propiedad del mismo equipo que lo elaboro durante toda su vida útil.

2.2.3 Autonomía

Cada microservicio debe funcionar de manera autónoma, lo que significa que no comparte recursos ni depende de otros componentes o servicios del sistema.

Esto permite implementar cada microservicio en la tecnología más adecuada para las necesidades del negocio.

Adicionalmente esto implica que un cambio realizado sobre un microservicio no debe afectar al resto de componentes del sistema.

2.2.4 Gestión descentralizada de datos

Cada microservicio tiene su propio almacén de datos, al que ningún otro componente o microservicio puede acceder directamente. Todo el acceso a los datos se logra a través de la funcionalidad expuesta por el microservicio propietario.

2.2.5 Tolerancia a fallos

Una consecuencia del uso de servicios como componentes es que las aplicaciones deben diseñarse de modo que puedan tolerar fallos en los servicios. Cualquier llamada de servicio podría fallar debido a la falta de disponibilidad; sin embargo, el cliente debe responder a esto de la mejor manera posible.

2.2.6 Automatización

Los sistemas que se elaboran con microservicios deben ser construidos de forma que puedan ser desplegados de manera automática utilizando técnicas de automatización de infraestructura (devops), tales como la integración y el despliegue continuos.

2.2.7 Mecanismos de comunicación ligeros

Los microservicios que componen las aplicaciones deben actuar como filtros en el sentido que reciben una solicitud, aplican la lógica correspondiente y producen una respuesta. Esta interacción se debe llevar a cabo de forma transparente utilizando mecanismos simples de mensajería compatibles con el resto de los componentes del sistema.

2.3 Estrategias de autorización para aplicaciones de microservicios

En la literatura referente a las investigaciones realizadas sobre autorización en aplicaciones distribuidas y basadas en servicios se evidenció una tendencia al uso de mecanismos basados en tokens [2] [9] [18] [19], siendo el estándar JSON Web Token (JWT) el más utilizado por los investigadores en sus análisis.

El estándar JWT define una forma compacta y autónoma para transmitir información de forma segura entre dos partes utilizando objetos en formato JSON. Esta información puede ser verificada debido a que cuenta con una firma digital. Los JWT pueden firmarse usando un secreto, mediante el algoritmo HMAC o un par de claves pública / privada usando RSA o ECDSA [20].

A pesar de la marcada tendencia hacia el uso de tokens para la autorización de microservicios y con el fin de ahondar en el estudio de las estrategias de

autorización compatibles con microservicios, también se analizaron los mecanismos basados en sesiones de usuarios mencionados en el artículo “Autenticación y autorización de usuarios finales en la arquitectura de microservicios” [8].

En las siguientes secciones se describirán los diferentes mecanismos de autorización, para posteriormente poder realizar un análisis comparativo entre ellos.

2.3.1 Autorización basada sesiones de usuario replicadas

En este esquema el usuario genera una sesión proporcionando sus credenciales, esta sesión contiene información sobre la identidad y los permisos del usuario, esta sesión es replicada en todos los servicios que componen la aplicación. Cuando un servicio recibe una petición verifica los permisos del usuario en la sesión antes de conceder o negar el acceso al recurso solicitado.

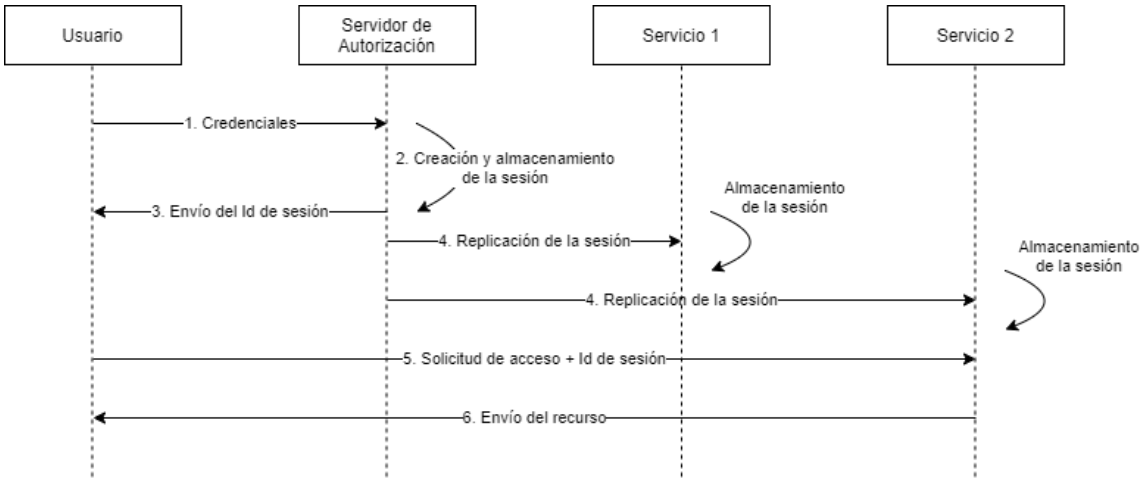

Ilustración 2: Flujo de autorización basada en sesiones replicadas

2.3.2 Autorización basada en sesiones de usuario centralizadas

En este esquema un usuario genera una sesión proporcionando sus credenciales, la sesión generada se almacena en un sistema centralizado de almacenamiento, Cuando un servicio recibe una petición, consultara al sistema de almacenamiento centralizado la información de la sesión, para verificar los permisos del usuario antes de conceder o negar el acceso al recurso solicitado.

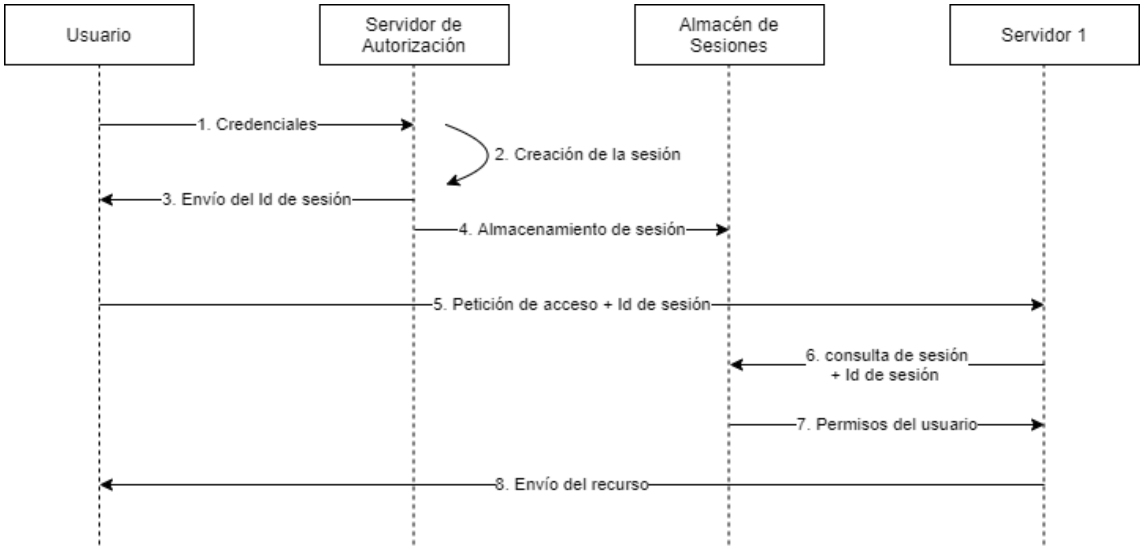

Ilustración 3: Flujo de autorización basada en sesiones centralizadas

2.3.3 Autorización basada en tokens sin firma

En este esquema el usuario solicita un token utilizando sus credenciales, el servidor autorización verifica estas credenciales, genera un token de autorización y lo envía al usuario. Cuando un servicio recibe una petición con el token del usuario solicitando el de acceso a un determinado recurso, el servicio se comunica con el servidor de autorización para verificar la validez del token proporcionado, finalmente cuando el token ha sido verificado por el servidor de autorización el servicio concede o niega el acceso al recurso solicitado.

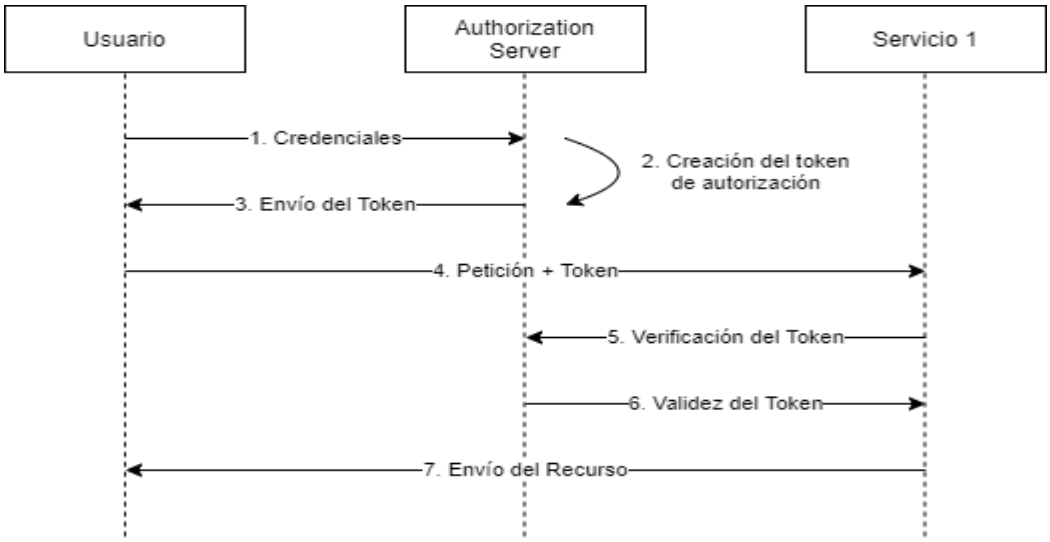

Ilustración 4: Flujo de autorización basada en tokens sin firma

2.3.4 Autorización basada en tokens firmados

En este esquema cada microservicio cuenta con un secreto que le permitirá validar los tokens generados y firmados por el servidor de autorización.

El flujo de autorización comienza cuando el usuario solicita un token utilizando sus credenciales, el servidor autorización verifica estas credenciales, genera un token de autorización y lo envía al usuario. Cuando un servicio recibe una petición con el token del usuario solicitando el de acceso a un determinado recurso, utilizando el secreto proporcionado por el servidor de autorización verifica la validez del token, finalmente si el token ha sido verificado positivamente le concede al usuario el acceso al recurso, caso contrario se lo niega.

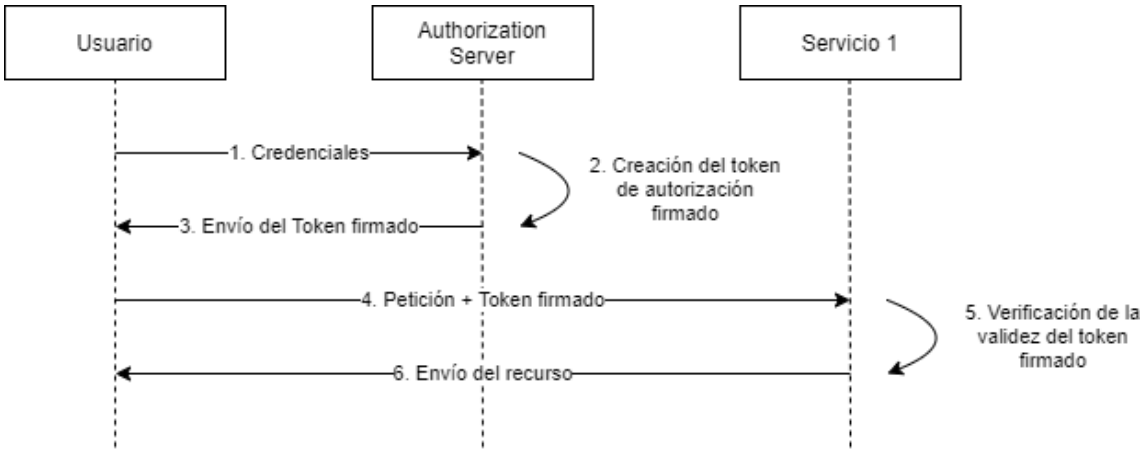

Ilustración 5: Flujo de autorización basado en tokens firmados

2.3.5 Autorización basada en tokens firmados, utilizando un API Gateway

En este esquema todas las peticiones se realizan a través de un API Gateway, el API Gateway manejará dos tipos de tokens uno para la comunicación con el usuario y otro para la comunicación con los microservicios. El token utilizado para la comunicación con los microservicios es el proporcionado por el servidor de autorización. El token utilizado para la comunicación con los usuarios es el token proporcionado por el servidor de autorización sometido a un proceso de ofuscación.

El flujo de autorización comienza cuando el usuario solicita un token utilizando sus credenciales, el API Gateway retransmite la petición al servidor autorización, después de la verificación de las credenciales proporcionadas se genera un token de autorización que es enviado al API Gateway. El API Gateway elabora un token modificado a partir el token original, el token modificado es almacenado y transmitido al cliente para que este lo utilice en futuras peticiones.

Cuando el usuario solicita el acceso a un recurso de un microservicio, el API Gateway recibe la petición con el token modificado, evalúa su validez, recupera el token original utilizando el token modificado y envía la petición del usuario al microservicio correspondiente utilizando el token original. Finalmente, e; microservicio verifica la validez del token antes de conceder o negar el acceso al recurso solicitado.

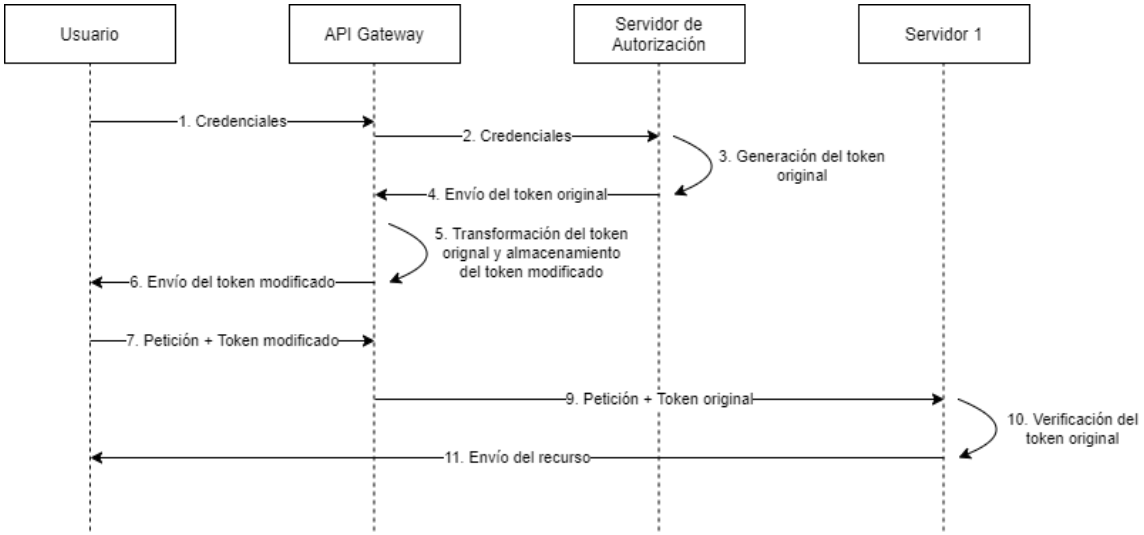

Ilustración 6: Flujo de autorización basada en tokens firmados, utilizando un API Gateway

2.4 Análisis comparativo de las estrategias de autorización

Con el fin de determinar cuál es el mecanismo de autorización que mejor se adapta a la arquitectura de microservicios se realizará un análisis de compatibilidad entre las características que definen a esta arquitectura y las estrategias de autorización descritas en el apartado anterior.

Para valorar el grado de compatibilidad entre una estrategia de autorización y una característica de la arquitectura de microservicios, se utilizará la siguiente escala de valores.

- 1 – cuando son compatibles
- 0.5 – cuando exista una compatibilidad parcial
- 0 – cuando no son compatibles

En caso de existir incertidumbre al momento de determinar la compatibilidad entre las estrategias de autorización y las características de la arquitectura estudiada, se utilizará el valor de compatibilidad parcial.

Al finalizar el análisis parcial de compatibilidad se consolidarán los resultados obtenidos con el fin de sumarlos y obtener una puntuación total final que identifique al mecanismo de autorización más adecuado para aplicaciones basadas en microservicios.

2.4.1 Compatibilidad respecto a la componentización a través de servicios

Estrategia de Autorización	Compatibilidad	Justificación
Sesiones replicadas	0	Requiere la implementación de funcionalidad destinada a la replicación y almacenamiento de sesiones además del dominio que implementa el servicio.

Sesiones centralizadas	0.5	El almacén de sesiones puede requerir un tipo de estandarización para las interacciones con el almacén de sesiones.
Tokens sin firma	1	El enfoque no afecta la componentización del sistema.
Tokens con firma	1	El enfoque no afecta la componentización del sistema.
Tokens + API Gateway	1	El enfoque no afecta la componentización del sistema.

Tabla 2: Compatibilidad entre la característica de componentización a través de servicios y las estrategias de autorización

2.4.2 Compatibilidad respecto a la gestión de productos, no proyectos

Estrategia de Autorización	Compatibilidad	Justificación
Sesiones replicadas	0.5	Para la replicación y sincronización de sesiones se requiere la colaboración todos los equipos encargados del desarrollo de los diferentes microservicios del sistema.
Sesiones centralizadas	0.5	Se requiere comunicación y colaboración entre los equipos de desarrollo del sistema para establecer la estrategia de sesión centralizada.
Tokens sin firma	0.5	Se requiere comunicación y colaboración con el equipo encargado del servicio de autorización para la validación de los tokens.
Tokens con firma	1	El enfoque no altera la dinámica de la aplicación.
Tokens + API Gateway	1	El enfoque no altera la dinámica de la aplicación.

Tabla 3: Compatibilidad entre la característica de gestión de productos, no proyectos y las estrategias de autorización

2.4.3 Compatibilidad respecto a la autonomía

Estrategia de Autorización	Compatibilidad	Justificación
Sesiones replicadas	0	Cada microservicio depende de los demás para la replicación y sincronización de sesiones.

Sesiones centralizadas	0	Los microservicios dependen de la disponibilidad del almacén de sesiones.
Tokens sin firma	0	Los microservicios dependen de la disponibilidad del servicio de autorización.
Tokens con firma	0.5	Los microservicios dependen del servidor de autorización para la obtención de los secretos necesarios para validar los tokens.
Tokens + API Gateway	0.5	Los microservicios dependen del servidor de autorización para la obtención de los secretos necesarios para validar los tokens.

Tabla 4: Compatibilidad entre la característica de autonomía y las estrategias de autorización

2.4.4 Compatibilidad respecto a la gestión descentralizada de datos

Estrategia de Autorización	Compatibilidad	Justificación
Sesiones replicadas	0	Requiere comunicarse con otros servicios para replicar y sincronizar sesiones.
Sesiones centralizadas	0	Se utiliza un repositorio común para el almacenamiento de sesiones.
Tokens sin firma	1	La implementación no se altera la gestión de almacenamiento de datos de los microservicios.
Tokens con firma	1	La implementación no se altera la gestión de almacenamiento de datos de los microservicios.
Tokens + API Gateway	1	La implementación no se altera la gestión de almacenamiento de datos de los microservicios.

Tabla 5: Compatibilidad entre la característica de gestión descentralizada de datos y las estrategias de autorización

2.4.5 Compatibilidad respecto a la tolerancia a fallos

Estrategia de Autorización	Compatibilidad	Justificación
Sesiones replicadas	0	Un fallo en un servidor puede complicar los procesos de sincronización y replicación de sesiones.

Sesiones centralizadas	0	Si el almacén centralizado de sesiones falla la aplicación limitaría significativamente su funcionalidad.
Tokens sin firma	0	El servidor de autorización falla las peticiones con control de acceso dejarían de funcionar.
Tokens con firma	1	Fallos en el servidor de autorización no limitarían la funcionalidad de los servicios que ya cuenten con secretos para validar los tokens.
Tokens + API Gateway	0	Si el API Gateway falla, el sistema deja de ofrecer sus funcionalidades.

Tabla 6: Compatibilidad entre la característica de tolerancia a fallos y las estrategias de autorización

2.4.6 Compatibilidad respecto al despliegue automatizado

Estrategia de Autorización	Compatibilidad	Justificación
Sesiones replicadas	0.5	Se pueden aplicar técnicas de despliegue automatizado, siempre y cuando no existan cambios significativos en la gestión de sesiones.
Sesiones centralizadas	1	Se pueden aplicar técnicas de despliegue automatizado en todos los componentes del sistema.
Tokens sin firma	1	Se pueden aplicar técnicas de despliegue automatizado en todos los componentes del sistema.
Tokens con firma	1	Se pueden aplicar técnicas de despliegue automatizado en todos los componentes del sistema.
Tokens + API Gateway	1	Se pueden aplicar técnicas de despliegue automatizado en todos los componentes del sistema.

Tabla 7: Compatibilidad entre la característica de despliegue automatizado y las estrategias de autorización

2.4.7 Compatibilidad respecto al uso de mecanismos de comunicación simples

Estrategia de Autorización	Compatibilidad	Justificación
Sesiones replicadas	0	Para asegurar una adecuada replicación y sincronización serían necesarios mecanismos de comunicación más sofisticados.
Sesiones centralizadas	0.5	El almacén de sesiones puede requerir un tipo especial de comunicación.
Tokens sin firma	0.5	La verificación centralizada de tokens generaría un alto flujo de datos en los canales de comunicación.
Tokens con firma	1	La comunicación entre los componentes del sistema se realizaría utilizando mecanismos simples.
Tokens + API Gateway	1	La comunicación entre los componentes del sistema se realizaría utilizando mecanismos simples.

Tabla 8: Compatibilidad entre la característica de uso de mecanismos simples de comunicación y las estrategias de autorización

2.4.8 Resultado

A continuación, para identificar la estrategia de autorización que mejor se adapta a la arquitectura de microservicios se consolidarán los resultados del análisis de compatibilidad de cada una de las características estudiadas y las diferentes estrategias de autorización en la siguiente tabla.

Mecanismo de Autorización	C1 (/1)	C2 (/1)	C3 (/1)	C4 (/1)	C5 (/1)	C6 (/1)	C7 (/1)	Compatibilidad Total (/7)
Sesiones replicadas	0	0.5	0	0	0	0.5	0	**1**
Sesiones centralizadas	0.5	0.5	0	0	0	1	0.5	**2.5**
Tokens sin firma	1	0.5	0	1	0	1	0.5	**4**
Tokens con firma	1	1	0.5	1	1	1	1	**6.5**
Tokens + API Gateway	1	1	0.5	1	0	1	1	**5.5**

Tabla 9: Resultados de compatibilidad entre las estrategias de autorización y las características de la arquitectura de microservicios

El valor de la columna “Compatibilidad Total” se obtiene al sumar los valores de compatibilidad parciales obtenidos por cada una de las estrategias de autorización, como indica la siguiente fórmula.

$$Compatibilidad\ Total = \sum_{i=1}^{n} Compatibilidad_i$$

Para facilitar la visualización de los resultados obtenidos en Tabla 9, se elaboró la siguiente grafica con los valores de compatibilidad total en términos porcentuales.

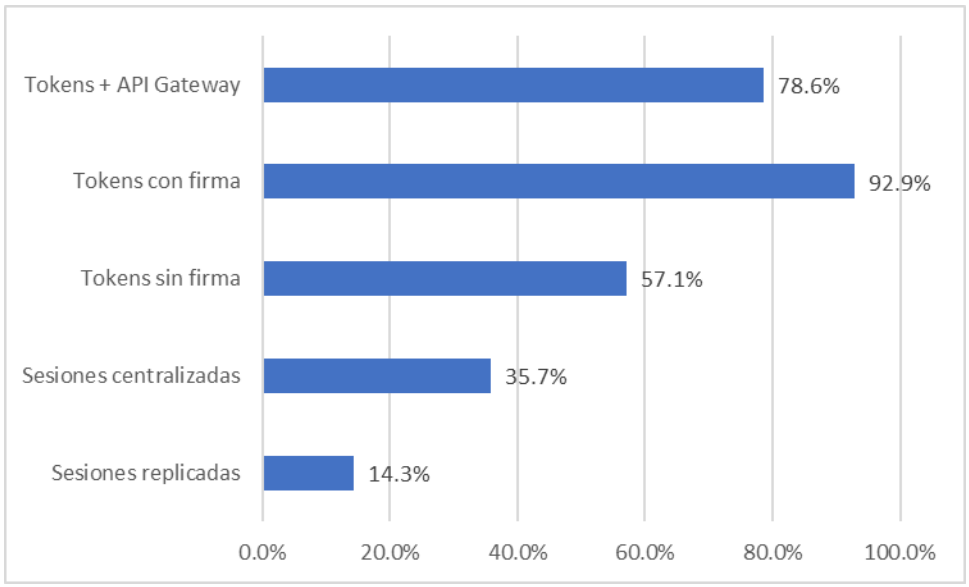

Ilustración 7: Porcentaje de compatibilidad total entre las estrategias de autorización y las características de la arquitectura de microservicios

Después de comparar los resultados de los mecanismos de autorización, se determinó que la estrategia de autorización más adecuada para aplicaciones basadas en microservicios es la utilización de tokens firmados, ya que esta opción obtuvo una puntuación de 6.5 sobre 7 que equivale a un 92.9% de compatibilidad con las principales características de la arquitectura de microservicios.

2.5 Definición del caso de estudio

Tomando en cuenta que uno de los objetivos del presente proyecto es verificar el funcionamiento de la estrategia de autorización identificada en el apartado anterior. Se propone la siguiente aplicación como caso de estudio.

2.5.1 Descripción de la aplicación

Se plantea la creación de una aplicación web para el seguimiento académico, en la cual estudiantes y sus representantes puedan visualizar sus calificaciones.

2.5.2 Requerimientos del sistema

- Todos los usuarios del sistema podrán visualizar su perfil.
- Dentro de la aplicación tanto los estudiantes, como sus representantes tendrán acceso a las calificaciones de los cursos en los que se encuentran registrados los alumnos.
- Los estudiantes y sus respectivos representantes podrán visualizar las notificaciones publicadas por el personal administrativo y docente de la institución académica.
- El personal docente de la institución podrá registrar y modificar las calificaciones de los estudiantes registrados en sus cursos.
- El personal docente podrá visualizar las notificaciones publicadas por el personal administrativo de la institución.
- El personal docente podrá crear y editar notificaciones dirigidas hacia estudiantes o representantes.
- El personal administrativo de la institución podrá crear y modificar los cursos que se imparten.
- El personal administrativo de la institución podrá crear, modificar y eliminar notificaciones dirigidas a cualquier audiencia.
- El personal administrativo de la institución podrá vincular o desvincular estudiantes y representantes.

2.6 Resumen

En este capítulo se elaboró una definición para la arquitectura de microservicios tomando como referencia definiciones propuestas por diferentes autores. Subsiguientemente se identificaron las principales características que definen a este modelo arquitectónico.

Asimismo, se identificaron los principales mecanismos de autorización para aplicaciones basadas en microservicios y a través de un análisis comparativo entre la compatibilidad de las estrategias de autorización y las características de la arquitectura basada en microservicios se identificó que el uso de tokens firmados es el mecanismo de autorización más adecuado para aplicaciones basadas en esta arquitectura.

Finalizando el capítulo se definió la aplicación que servirá como caso de estudio para implementación de la prueba de concepto correspondiente a la parte práctica del proyecto.

3. Implementación

Una vez identificada una estrategia de autorización adecuada para aplicaciones basadas en microservicios, en este capítulo se profundizará en el análisis de su funcionamiento mediante la implementación de una prueba de concepto basada en el caso de estudio propuesto.

3.1 Análisis del caso de estudio

Para definir los controles de acceso que se deben implementar en el caso de estudio es necesario identificar los roles o perfiles de usuario, los recursos y las acciones que pueden realizar los diferentes perfiles de usuario sobre los recursos dentro de la aplicación.

3.1.1 Roles

En base a los requerimientos del caso de estudio se pudieron reconocer los siguientes roles o perfiles de usuario.

- Administrador: miembros del personal administrativo de la institución académica.
- Profesor: miembros del personal docente de la institución académica.
- Estudiante: alumnos inscritos en la institución académica.
- Representante: padres de familia o representantes legales de los alumnos de la institución académica.

3.1.2 Recursos

Los recursos identificados a partir de los requerimientos del caso de estudio son los siguientes.

- Cursos
- Calificaciones
- Notificaciones
- Perfil personal

3.1.3 Acciones permitidas para los perfiles de usuario

Habiendo identificado los perfiles de usuario y los recursos disponibles en la aplicación, para establecer las acciones disponibles para cada uno de los roles se generó la siguiente tabla de control de acceso.

En las celdas correspondientes al cruce entre un recurso y un rol se listarán las acciones disponibles para dicha combinación; por ejemplo, en la celda correspondiente al cruce entre el rol estudiante y el recurso perfil personal se debería colocar la acción de visualizar.

Roles / Recursos	Cursos	Calificaciones	Notificaciones	Perfil personal
Administrador	Visualizar, crear y modificar		Visualizar, crear, modificar y eliminar	Visualizar
Profesor	Visualizar	Visualizar, crear y modificar	Visualizar, crear y modificar	Visualizar
Estudiante	Visualizar	Visualizar	Visualizar	Visualizar
Representante	Visualizar	Visualizar	Visualizar	Visualizar

Tabla 10: Control de acceso del caso de estudio

3.2 Diseño de la aplicación

Una herramienta utilizada para construir microservicios es el diseño basado en dominios (DDD), este estilo de diseño usa el principio denominado “divide y vencerás” que consiste en agrupar las funcionalidades de un sistema en base a sus recursos compartidos y relaciones en común de tal forma que se pueda construir una aplicación como un conjunto de microservicios independientes, pero colaborativos [21].

Gracias a este análisis de pudieron identificar los siguientes dominios:

- Dominio académico: aquí se agrupan las funcionalidades relacionadas con los cursos y las calificaciones, esta agrupación se constituyó considerando el hecho de que las calificaciones se otorgan a los alumnos registrados en un curso.
- Dominio informativo: este dominio contiene la funcionalidad referente a la gestión de notificaciones, se decidió separar la gestión de notificaciones del dominio anterior ya que esta funcionalidad no posee dependencias con la gestión de cursos o calificaciones.
- Dominio de identidad: en este dominio se encuentra la funcionalidad de gestión de usuarios y la gestión de los controles de autorización. Se opto por separar esta funcionalidad en un dominio independiente porque esta funcionalidad no es compatible con las tareas que se realizan en dominios académico o informativo dentro del caso de estudio que se está analizando, cabe destacar que este dominio es referenciado por el dominio académico en la relación que existe entre cursos, profesores y estudiantes; y por el dominio informativo en los destinatarios de las notificaciones.

En el siguiente gráfico se pueden visualizar los dominios identificados para el caso de estudio.

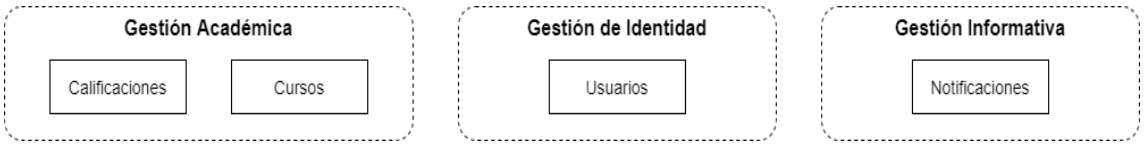

Ilustración 8: Dominios del caso de estudio

Posteriormente de haber agrupado las funcionalidades del sistema en dominios, se establecieron los componentes del sistema a implementar. A cada uno de los dominios se lo consideró como un microservicio y adicionalmente se determinó que era necesaria la implementación de un cliente web para que los usuarios del sistema puedan interactuar con los microservicios. La distribución del sistema se puede apreciar en el siguiente diagrama.

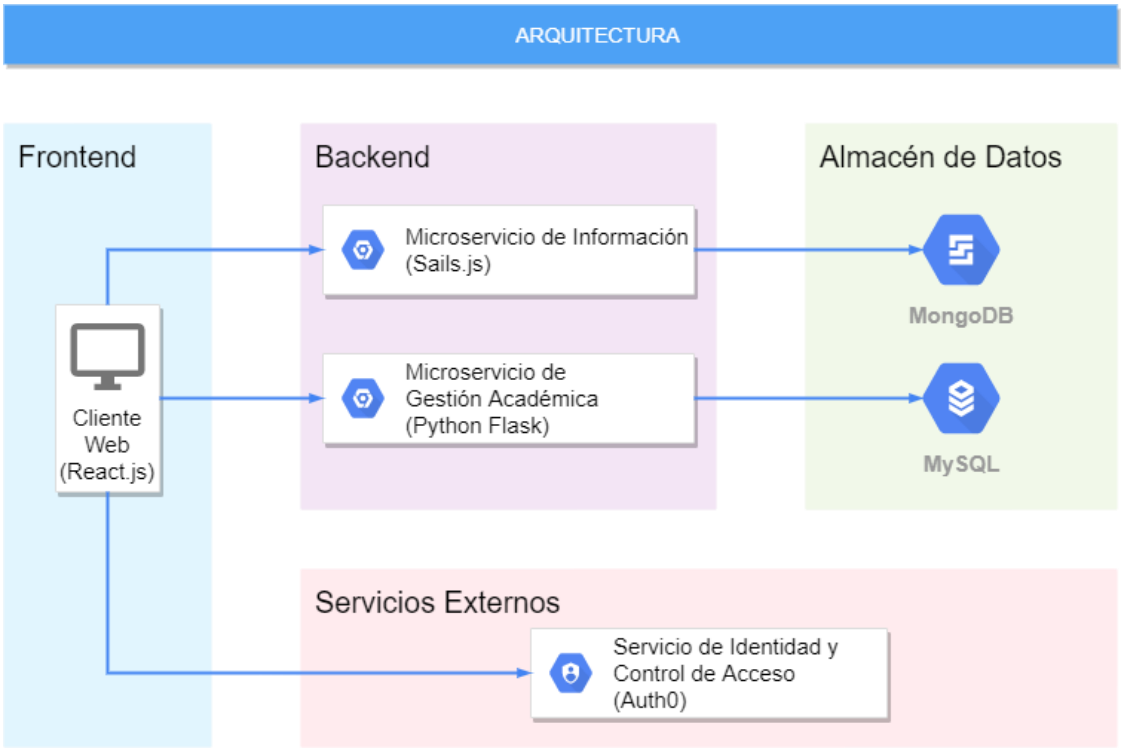

Ilustración 9: Diseño de la aplicación caso de estudio

3.3 Selección de tecnología

En este apartado se seleccionarán las tecnologías a utilizarse en la implementación de la prueba de concepto para el mecanismo de autorización basado en tokens firmados.

3.3.1 Tokens

Para los tokens de autorización se tomaron en cuenta las recomendaciones de los autores He y Yang [8], que sugieren utilizar tokens JWT firmados digitalmente utilizando un par de llaves pública y privada para el control de autorización en ambientes distribuidos. Esta tecnología permite que los tokens se puedan verificar en cada uno de los servicios mitigando la dependencia con el servicio de gestión de usuarios y al mismo tiempo se reduce el riego de que entidades no autorizadas generen tokens válidos y accedan a los recursos restringidos del sistema.

3.3.2 Cliente Web

Para el cliente web se decidió utilizar una SPA, debido a que este tipo de aplicaciones de una sola página, se pueden elaborar de una manera rápida e integrar fácilmente con varios servicios. Se seleccionó el lenguaje de programación JavaScript junto a la librería React.js para la fabricación de este componente del sistema, puesto que cuenta con una gran base documental y un amplio catálogo de herramientas lista para su uso.

3.3.3 Microservicios

En este punto para comprobar la validez del mecanismo de autorización en arquitecturas de microservicios se optó por utilizar tecnologías diferentes para implementación de los microservicios de gestión académica y de gestión informativa.

Tomando como base el artículo “Mejores lenguajes para microservicios” [23] y el grado de conocimiento de los lenguajes de programación expuestos en el artículo, las tecnologías seleccionadas para implementar los microservicios de la prueba de concepto fueron los siguientes.

Microservicio de información

- Lenguaje de programación: JavaScript
- Librerías: Sails.js
- Reportorio de datos: MongoDB

Microservicio de gestión académica

- Lenguaje de programación: Python
- Librerías: Flask
- Repositorio de datos: MySQL

3.3.4 Servicio de gestión de identidad

Para este servicio se consideró sensato utilizar una solución disponible en el mercado que cumpla con los estándares de implementación para tokens de tipo JWT, esta decisión se apoya en el hecho que el objetivo del presente trabajo no es la implementación del mecanismo de autorización, sino más bien la evaluación de su funcionamiento.

La elección de la solución a utilizar se realizó en base a un análisis comparativo entre las aplicaciones Aut0, Keycloak y FusionAuth, elegidas para este estudio por su documentación disponible y su popularidad en la comunidad de desarrolladores. Se utilizó la siguiente escala para evaluar los parámetros que debe cumplir cada uno de los sistemas analizados.

- 0 – la solución no cumple con el parámetro de evaluación
- 0.5 – la solución cumple parcialmente con el parámetro de evaluación
- 1 – la solución cumple satisfactoriamente con el parámetro de evaluación

Parámetro	Auth0	Keycloak	FusionAuth
Permite administrar usuarios	1	1	1
Permite gestionar los permisos de acceso de los usuarios	1	1	1
Cuenta con un API para tareas de gestión	1	1	1
Puede generar tokens JWT firmados con llaves privadas	1	1	1
Se puede utilizar gratuitamente	0.5	1	0.5
Cuenta con documentación ilustrativa para integraciones	1	0.5	0.5
Facilidad de despliegue	1	0.5	0.5
Total	**6.5**	**6**	**5.5**

Tabla 11: Comparativa entre soluciones de gestión de usuarios y control de acceso

En base a los resultados del análisis la aplicación Auth0, obtuvo el mayor puntaje, por lo tanto, será la solución utilizada como el servicio de gestión de usuarios y control de autorización en la prueba de concepto.

A pesar de que es una aplicación de pago cuenta con una versión gratuita que provee las funcionalidades necesarias para implementar y evaluar la prueba de concepto del presente proyecto, un factor clave que favoreció la selección de esta solución fue la amplia documentación disponible para integrar los mecanismos de verificación de los tokens en diferentes lenguajes de programación, facilitando de este modo la tarea de integrar los controles de autorización en el resto de microservicios del sistema.

3.4 Evaluación de la prueba de concepto

En este apartado se evaluará tanto el funcionamiento, como la efectividad del mecanismo de autorización basado en tokens JWT firmados con un par de llaves pública y privada, implementado sobre una aplicación web basada en microservicios.

3.4.1 Casos de prueba

Para la evaluación de la prueba de concepto se consideraron los siguientes casos.

Generar un token de autorización para cada uno de los roles de usuario, este conjunto de pruebas tiene como objetivo verificar que la asignación de permisos por parte del servicio de gestión de usuarios y control de acceso funcione correctamente.

1) Generar un token para un usuario de rol “estudiante”
2) Generar un token para un usuario de rol “representante”
3) Generar un token para un usuario de rol “profesor”
4) Generar un token para un usuario de rol “administrador”

Realizar una consulta a un servicio sin proporcionar el token de autorización, esta prueba tiene como objetivo comprobar que el microservicio evaluado está comprobando que el usuario o el sistema haya proporcionado los permisos necesarios para efectuar una operación antes de ejecutarla.

5) Realizar una consulta al servicio de gestión académica sin proporcionar el token de autorización

Realizar una consulta a un servicio proporcionando un token de autorización expirado, esta prueba pretende comprobar que además de verificar que el usuario o el sistema han proporcionado los permisos necesarios para efectuar una operación estos posean un estado válido antes de efectuar la operación solicitada.

6) Realizar una consulta al servicio de gestión de información proporcionando un token de autorización expirado

Realizar una petición de edición utilizando un token alterado, esta prueba tiene como propósito comprobar que los algoritmos de verificación de los tokens implementados en los microservicios evalúan los permisos proporcionados por el usuario o el sistema no han sido alterados con el fin de evadir los controles de acceso de la aplicación.

7) Realizar una consulta al servicio de gestión de información proporcionando un token de autorización modificado

Realizar una petición a un microservicio utilizando un token sin los privilegios necesarios para realizar la acción solicitada, esta prueba pretende identificar si el microservicio evaluado está verificando los permisos del usuario además de la validez del token de autorización proporcionado, previo a la ejecución de la acción.

8) Realizar una petición para editar una calificación utilizando un token correspondiente a un usuario con rol de “estudiante”
9) Realizar una petición para crear de un nuevo curso utilizando un token de autorización correspondiente a un usuario con rol “profesor”
10) Realizar una petición para editar una notificación utilizando un token de autorización correspondiente a un usuario con rol “representante”

3.4.2 Ejecución de las pruebas

A continuación, se muestran los resultados de la ejecución de los casos de prueba definidos en el apartado anterior.

1) Caso 1: Generar un token para un usuario de rol “estudiante”

Descripción
Generar un token de autorización para un usuario con el rol de estudiante.
Pasos por seguir 1. Ingresar las credenciales de acceso de un usuario con rol de estudiante. 2. Dar clic al botón ingresar para generar el token 3. Verificar el contenido del token generado
Datos de prueba - Usuario: cnoble@tfm.com - Contraseña: Student2020
Resultado esperado Un token que contenga el únicamente el rol de estudiante (student) y los permisos: - read:grades - read:notifications - read:courses

Resultado obtenido
 	
{
  "alg": "RS256",
  "typ": "JWT",
  "kid": "vjhdPiq2lQfs7-wcp7NJ5"
}
{
  "userinfo": {
  "email": "cnoble@tfm.com",
  "roles": ["student"]
  },
  "scope": "openid profile email read:notifications
  read:grades read:courses"
}

Estado de la prueba | Aprobado

Tabla 12: Resultado del primer caso de prueba

2) Caso 2: Generar un token para un usuario de rol “representante”

Descripción

Generar un token de autorización para un usuario con el rol de representante.

Pasos por seguir

1. Ingresar las credenciales de acceso de un usuario con rol de representante.
2. Dar clic al botón ingresar para generar el token
3. Verificar el contenido del token generado

Datos de prueba

- Usuario: jcharles@tfm.com
- Contraseña: Tutor2020

Resultado esperado

Un token que contenga el únicamente el rol de representante o padre de familia, un listado de los alumnos a los que represente y los permisos:

- read:grades
- read:notifications
- read:courses

Resultado obtenido


```
{
  "alg": "RS256",
  "typ": "JWT",
  "kid": "vjhdPiq2lQfs7-wcp7NJ5"
}
{
  "userinfo": {
    "email": "jcharles@tfm.com",
    "roles": ["tutor"],
    "students": ["auth0|5eb8331454b14c0c128cf847"]
  },
  "scope": "openid profile email read:notifications read:grades
            read:courses "
}
```


Estado de la prueba	Aprobado

Tabla 13: Resultado del segundo caso de prueba

3) Caso 3: Generar un token para un usuario de rol “profesor”

Descripción

Generar un token de autorización para un usuario con el rol de profesor.

Pasos por seguir

1. Ingresar las credenciales de acceso de un usuario con rol de profesor.
2. Dar clic al botón ingresar para generar el token
3. Verificar el contenido del token generado

Datos de prueba

- Usuario: jstuart@tfm.com
- Contraseña: Teacher2020

Resultado esperado

Un token que contenga el únicamente el rol de profesor (teacher) y los permisos:

- create:grades
- edit:grades
- read:grades
- read:courses
- create:notifications
- edit:notifications
- read:notifications

Resultado obtenido


```
{
  "alg": "RS256",
  "typ": "JWT",
  "kid": "vjhdPiq2lQfs7-wcp7NJ5"
}
{
  "userinfo": {
    "email": "jstuart@tfm.com",
    "roles": ["teacher"],
  },
  "scope": "openid profile email read:notifications read:courses
            edit:notifications create:notifications create:grades
            edit:grades read:grades"
}
```


Estado de la prueba	Aprobado

Tabla 14: Resultado del tercer caso de prueba

4) Caso 4: Generar un token para un usuario de rol “administrador”

Descripción
Generar un token de autorización para un usuario con el rol de administrador.
Pasos por seguir 1. Ingresar las credenciales de acceso de un usuario con rol de representante. 2. Dar clic al botón ingresar para generar el token 3. Verificar el contenido del token generado
Datos de prueba - Usuario: agrandar@tfm.com - Contraseña: Admin2020
Resultado esperado Un token que contenga el únicamente el rol de administrador (sys-admin) y los permisos: - create:courses - edit:courses - read:courses - create:notifications - edit:notifications - read:notifications - delete:notifications

Resultado obtenido
 	
{
"alg": "RS256",
"typ": "JWT",
"kid": "vjhdPiq2lQfs7-wcp7NJ5"
}
{
"userinfo": {
"email": "agrandar@tfm.com",
"roles": ["sys-admin"]
},
"scope": "openid profile read:notifications create:notifications
delete:notifications edit:notifications
create:courses edit:courses read:courses"
}

Estado de la prueba | Aprobado

Tabla 15: Resultado del cuarto caso de prueba

5) Caso 5: Realizar una consulta al servicio de gestión académica sin proporcionar el token de autorización

Descripción
Realizar una consulta de los cursos existentes, sin utilizar un token de acceso.
Pasos por seguir 1. Realizar una petición de consulta al servicio de gestión académica, sin utilizar un token de acceso 2. Verificar la respuesta del servicio
Datos de prueba - Servicio de gestión de académica, endpoint de consulta de cursos - Método GET
Resultado esperado El servicio debe responder con el código de error 401 y el mensaje no se encontró el token de acceso para la petición
Resultado obtenido `{ "message": "No se encontró el token de acceso para la petición", "status": 401 }`

Estado de la prueba	Aprobado

Tabla 16: Resultado del quinto caso de prueba

6) Caso 6: Realizar una consulta al servicio de gestión de información proporcionando un token de autorización expirado

Descripción

Utilizar un token expirado para la consulta de notificaciones.

Pasos por seguir

1. Generar un token de acceso para cualquier rol de usuario
2. Esperar a que se el tiempo de validez del token expire
3. Realizar una petición de consulta al servicio de notificaciones, con el token expirado
4. Verificar la respuesta del servicio

Datos de prueba

- Token expirado
- Servicio de información, endpoint de consulta de notificaciones
- Método GET

Resultado esperado

El servicio debe responder con el código de error 401 y el mensaje token expirado

Resultado obtenido

```
{
  "message": "Token expirado",
  "status": 401
}
```

Estado de la prueba	Aprobado

Tabla 17: Resultado del sexto caso de prueba

7) Caso 7: Realizar una consulta al servicio de gestión de información proporcionando un token de autorización modificado

Descripción
Utilizar un token de autorización modificado con el fin de editar una notificación.
Pasos por seguir 1. Generar un token de acceso para cualquier rol de usuario excepto el rol administrador 2. Modificar el token para agregar el permiso edit:notifications 3. Realizar una petición de edición al servicio de información, utilizando el token modificado 4. Verificar la respuesta del servicio
Datos de prueba - Token modificado - Servicio de información, endpoint de edición de notificaciones - Método PUT
Resultado esperado El servicio debe responder con el código de error 401 y el mensaje el token proporcionado no es válido.
Resultado obtenido `{ "message": "El token proporcionado no es válido", "status": 401 }`
Estado de la prueba | Aprobado

Tabla 18: Resultado del séptimo caso de prueba

8) Caso 8: Realizar una petición para editar una calificación utilizando un token correspondiente a un usuario con rol de “estudiante”

Descripción Utilizar un token de estudiante para editar una calificación.	
Pasos por seguir 1. Generar un token de un usuario con rol de estudiante 2. Hacer una petición al servicio de gestión académica utilizando el token generado 3. Verificar la respuesta del servicio	
Datos de prueba - Token de un usuario con rol de estudiante - Servicio de gestión académica, endpoint de edición de calificaciones - Método PUT	
Resultado esperado El servicio debe responder con el código de error 403 y el mensaje petición no autorizada.	

Resultado obtenido
 	
<pre>{
  "message": "Petición no autorizada",
  "status": 403
}</pre>	

Estado de la prueba	Aprobado

Tabla 19: Resultado del octavo caso de prueba

9) Caso 9: Realizar una petición de creación de un nuevo curso utilizando un token de autorización correspondiente a un usuario de rol “profesor”

Descripción
Utilizar un token de un profesor para crear un nuevo curso.
Pasos por seguir 1. Generar un token de un usuario con rol de profesor 2. Hacer una petición al servicio de gestión académica utilizando el token generado 3. Verificar la respuesta del servicio
Datos de prueba - Token de un usuario con rol de profesor - Servicio de gestión académica, endpoint de creación de cursos - Método POST
Resultado esperado Que el servicio responda con el código 403 y el mensaje petición no autorizada.

Resultado obtenido
 	
{
"message": "Petición no autorizada",
"status": 403
}

Estado de la prueba | Aprobado

Tabla 20: Resultado del noveno caso de prueba

10) Caso 10: Realizar una petición para editar una notificación utilizando un token de autorización correspondiente a un usuario con rol de “representante”

Descripción
Utilizar un token de un usuario representante para editar una notificación
Pasos por seguir
1. Generar un token de un usuario con rol de representante 2. Hacer una petición al servicio de información utilizando el token generado 3. Verificar la respuesta del servicio
Datos de prueba
- Token de un usuario con el rol de representante - Servicio de información, endpoint de edición de notificaciones - Método PUT
Resultado esperado
Que el servicio responda con el código 403 y el mensaje petición no autorizada.
Resultado obtenido

{
"message": "Petición no autorizada",
"status": 403
}

Estado de la prueba | Aprobado

Tabla 21: Resultado del décimo caso de prueba

3.5 Análisis de resultados

Después de haber concluido las pruebas del funcionamiento del mecanismo de autorización, en este apartado se agruparán los resultados obtenidos, para facilitar su revisión.

Caso de prueba	Estado
Generar un token para un usuario de rol “estudiante”	Aprobado
Generar un token para un usuario de rol “representante”	Aprobado
Generar un token para un usuario de rol “profesor”	Aprobado
Generar un token para un usuario de rol “administrador”	Aprobado
Realizar una consulta al servicio de gestión académica sin proporcionar el token de autorización	Aprobado
Realizar una consulta al servicio de gestión de información proporcionando un token de autorización expirado	Aprobado
Realizar una consulta al servicio de gestión de información proporcionando un token de autorización modificado	Aprobado
Realizar una petición para editar una calificación utilizando un token correspondiente a un usuario con rol de “estudiante”	Aprobado
Realizar una petición para crear de un nuevo curso utilizando un token de autorización correspondiente a un usuario con rol “profesor”	Aprobado
Realizar una petición para editar una notificación utilizando un token de autorización correspondiente a un usuario con rol “representante”	Aprobado

Tabla 22: Resultados de la ejecución de los casos de prueba sobre el mecanismo de autorización basado en tokens firmados

Los resultados obtenidos confirman los hallazgos conseguidos en la parte teórica del presente trabajo de fin de máster, con lo cual se puede ratificar que un mecanismo de autorización basado en tokens firmados es compatible y adecuado para gestionar el control de acceso a los recursos de las aplicaciones web basadas en microservicios.

3.6 Resumen

En este capítulo se realizó el diseño de un prototipo de aplicación web basada en microservicios, que permitió la implementación de una prueba de concepto para un mecanismo de autorización basado en tokens firmados digitalmente. La prueba de concepto se basa en los requerimientos planteados en la definición del caso de estudio del segundo capítulo del proyecto.

Se realizó un breve análisis de tecnologías disponibles, con el objetivo de establecer cuales se utilizarían en la implementación de la prueba de concepto, obteniendo como resultado que para los tokens se utilizaría la tecnología JWT con firmas digitales realizadas con un par de llaves privada y pública. En lo que respecta al servicio de autorización y gestión de usuarios se decidió utilizar la aplicación Auth0 existente en el mercado, ya que cumple con la referencia RFC 7519 para tokens JWT y cuenta con una amplia documentación para integrar los controles de acceso en servicios y aplicaciones codificados en varios lenguajes de programación como Python y JavaScript.

Complementariamente se decidió utilizar diferentes tecnologías para la implementación de los microservicios de gestión informativa y de gestión académica, con el propósito de validar si el servicio de gestión de usuarios y control de acceso realmente es compatible con microservicios codificados en diferentes tecnologías. El último componente del sistema es un cliente web para el cual se decidió utilizar el lenguaje JavaScript con la librería React.js

Finalmente, se puso a prueba el funcionamiento y la efectividad del mecanismo de autorización mediante el diseño y ejecución de casos de prueba, simulando escenarios donde se intentan burlar los controles de acceso a los recursos protegidos.

4. Conclusiones

En el presente proyecto se lograron contestar las preguntas de investigación planteadas, gracias a la recolección y análisis de información realizado inicialmente, permitiendo de esta forma cumplir los objetivos de la parte teórica del trabajo.

Inicialmente se logró evidenciar como la investigación académica en el área del desarrollo de software ha permitido que la industria pueda adaptar sus sistemas informáticos acorde a las necesidades de sus clientes, dejando atrás modelos arquitectónicos monolíticos y adoptando patrones de computación distribuida. Sin embargo, esta constante innovación ha ocasionado que los mecanismos de seguridad aplicados a este tipo de arquitecturas modernas como la basada en microservicios presenten deficiencias por no ser del todo compatibles con el paradigma de desarrollo de software que se está utilizando.

Subsecuentemente, mediante el estudio de las definiciones propuestas por diferentes investigadores se concluyó que la arquitectura de microservicios es un estilo arquitectónico para la construcción de software compuesto por un conjunto de servicios pequeños y autónomos, que pueden colaborar y comunicarse utilizando mecanismos ligeros de comunicación.

Para completar los objetivos de la primera parte del proyecto, a partir de la caracterización de la arquitectura de microservicios y la identificación de los mecanismos de autorización disponibles para sistemas distribuidos, se consiguió elaborar un análisis de compatibilidad para identificar cual es la estrategia de autorización más indicada para aplicaciones basadas en este tipo de arquitecturas, encontrando que el uso de mecanismos de autorización basados en tokens firmados, se adaptan mejor a las características de la arquitectura de microservicios.

En relación con los objetivos planteados en la parte práctica del proyecto, el análisis de los requerimientos del caso de estudio estableció un formidable punto de partida para identificar los componentes de la aplicación prototipo y como estos están relacionados. A través de la estrategia tomada del arte de la guerra “divide y vencerás” y presente también en el diseño basado en dominios, fue posible distribuir la funcionalidad del caso de estudio en microservicios, facilitando el proceso de su implementación.

Mediante la integración del servicio de gestión de usuarios y control de acceso a recursos, se evaluó el funcionamiento de los tokens JWT como mecanismo de autorización, utilizando un conjunto de casos de pruebas orientados a valorar el correcto funcionamiento de este mecanismo y su efectividad en la tarea de prevenir el acceso no autorizado a los recursos protegidos de la aplicación, permitiendo concluir que este mecanismo de autorización no solamente es compatible con la arquitectura de microservicios, sino que también es efectivo en la protección de información privada o confidencial.

4.1 Lecciones aprendidas

Durante la elaboración del proyecto se presentaron desafíos que pudieron solventarse mediante experimentación de tipo ensayo y error, o una investigación más profunda, el propósito de esta sección es compartir con el lector la experiencia adquirida para solventar este tipo de inconvenientes.

- En ocasiones resulta complicado encontrar un concepto o definición formal para un término ampliamente difundido, por lo cual es muy recomendable recolectar y analizar diferentes puntos de vista para clarificar las dudas que se pueden tener respecto a dicho tópico, en el presente proyecto esto se pudo evidenciar en la búsqueda de una definición para la arquitectura de microservicios.

- Para implementar un esquema adecuado de verificación de firmas y tokens JWT, es fundamental especificar explícitamente el algoritmo con el cual están siendo firmados los tokens, la razón de esto es que existen librerías que al no conocer específicamente el algoritmo de firma del token de autorización aceptar como válidos tokens alterados, comprometiendo de esta forma la información de la organización y de sus usuarios.

4.2 Trabajo futuro

El proyecto se centró en la identificación de un mecanismo de autorización adecuado para aplicaciones basadas en microservicios, sin embargo, existen interrogantes que surgieron durante el desarrollo de este trabajo que pueden explorarse en trabajos futuros.

1) Evaluar si mediante el uso de tokens JWT firmados, como estrategia de autorización se pueden solventar los requisitos de seguridad en el control de acceso a recursos documentados en guías de buenas prácticas como la NIST 800-95.
2) Realizar un análisis comparativo más detallado de las estrategias de autorización identificadas en el proyecto, mediante la implementación de cada una de ellas y estableciendo de métricas de rendimiento, funcionamiento, efectividad, entre otras. Con el propósito comprobar si los resultados obtenidos concuerdan con el análisis basado en compatibilidad desarrollado en este proyecto.
3) Analizar la viabilidad de implementar un sistema de detección de intrusos o un mecanismo similar, basándose en el registro de eventos sospechosos detectados y registrados durante la evaluación de los flujos de autorización. Por ejemplo, identificar si un usuario ha intentado acceder a información privada utilizando tokens expirados de otros usuarios o tokens manipulados para escalar privilegios.

5. Glosario

- **OWASP:** Siglas en inglés de Open Web Application Security Project, es una fundación sin fines de lucro que trabaja para mejorar la seguridad del software.
- **MVC:** Siglas de Modelo Vista Controlador, es un patrón de diseño de aplicaciones que consta de tres partes interconectadas: el modelo (los datos), la vista (la interfaz de usuario) y el controlador (los procesos que operan información de la aplicación).
- **SOC:** Siglas en inglés de Service-Oriented Computing, es paradigma en el cual los programas se denominan servicios y se comunican con otros componentes del sistema mediante mensajes. Los servicios desacoplan sus interfaces de comunicación de su implementación [3].
- **SOA:** Siglas en inglés de Service-Oriented Architecture, es un modelo de desarrollo de software que permite que servicios se comuniquen a través de diferentes plataformas e idiomas para formar aplicaciones. En SOA, un servicio es una unidad de software autónoma diseñada para completar una tarea específica.
- **Monolito:** Una aplicación de software cuyos módulos no se pueden ejecutar de forma independiente [3].
- **DDD:** Siglas en inglés de Domain Driven Design, es una filosofía de desarrollo de software centrada en dominios o esferas de conocimiento. El enfoque permite el desarrollo de software se centre en agrupar requisitos comunes y no desperdiciar esfuerzo en requisitos fuera del campo de acción de un dominio [24].
- **SPA:** Siglas en inglés de Simple Page Application, es una aplicación web que se presenta al usuario a través de una sola página HTML para que sea más receptiva y para replicar más de cerca el comportamiento de una aplicación de escritorio o una aplicación nativa [25].

6. Bibliografía

[1] F. Ponce, G. Márquez y H. Astudillo, «Migrating from monolithic architecture to microservices: A Rapid Review,» *38th International Conference of the Chilean Computer Science Society (SCCC),* pp. 1-7, 2019.

[2] B. Christudas, Practical Microservices Architectural Patterns, Trivandrum: Apress, 2019, pp. 29-34.

[3] N. Dragoni, S. Giallorenzo, A. Lluch Lafuente, M. Mazzara, F. Montesi, R. Mustafin y L. Safina, «Microservices: yesterday, today, and tomorrow,» *Present and Ulterior Software Engineering,* pp. 195-216, 2017.

[4] A. Bánáti, E. Kail, K. Karóczkai y M. Kozlovszky, «Authentication and authorization orchestrator for microservice-based software architectures,» *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO),* pp. 1180-1184, 2018.

[5] J. Fritzsch, J. Bogner, S. Wagner y A. Zimmermann, «Microservices Migration in Industry: Intentions, Strategies, and Challenges,» *2019 IEEE International Conference on Software Maintenance and Evolution,* pp. 481-490, 2019.

[6] The OWASP Foundation, «OWASP Top Ten,» The OWASP Foundation, 2017. [En línea]. Available: https://owasp.org/www-project-top-ten/. [Último acceso: marzo 2020].

[7] S. Echeverría, G. A. Lewis, D. Klinedinst y L. Seitz, «Authentication and Authorization for IoT Devices in Disadvantaged Environments,» *IEEE 5th World Forum on Internet of Things (WF-IoT),* pp. 368-373, 2019.

[8] X. He y X. Yang, «Authentication and Authorization of End User in Microservice Architecture,» *Journal of Physics: Conference Series,* vol. 910, nº 012060, pp. 1-8, 2017.

[9] A. Nehme, V. Jesus, K. Mahbub y A. Abdallah, «Fine-Grained Access Control for Microservices,» *Foundations and Practice of Security,* vol. 11358, pp. 285-287, 2019.

[10] J. Lewis y M. Fowler, «Microservices,» 25 marzo 2014. [En línea]. Available: https://martinfowler.com/articles/microservices.html. [Último acceso: marzo 2020].

[11] M. Hilbrich, «In Microservices We Trust — Do Microservices Solve Resilience Challenges?,» *Tagungsband des FB-SYS Herbsttreffens 2019,* pp. 1-2, 2019.

[12] A. Hannousse y S. Yahiouche, «Securing Microservices and Microservice Architectures: A Systematic Mapping Study,» 2020.

[13] Enabled Solutions, «Why Microservices Are The New Innovation Enablers For Enterprises,» Hackernoon, 13 agosto 2018. [En línea]. Available: https://hackernoon.com/why-microservices-are-the-new-innovation-enablers-for-enterprises-6a4c637fd901. [Último acceso: abril 2020].

[14] C. Pahl y P. Jamshidi, «Microservices: A Systematic Mapping Study,» *6th International Conference on Cloud Computing and Services Science,* vol. I, p. 137–146, 2016.

[15] M. Amaral, J. Polo, D. Carrera, I. Mohomed, M. Unuvar y M. Steinder, «Performance Evaluation of Microservices Architectures Using Containers,» *IEEE 14th International Symposium on Network Computing and Applications (NCA),* p. 27–34, 2015.

[16] D. López y E. Maya, «Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web,» *Séptima Conferencia de Directores de Tecnología de Información, TICAL 2017 Gestión de las TICs para la Investigación y la Colaboración,* pp. 1-12, 2017.

[17] C. Djernæs Nielsen, «Investigate availability and maintainability within a microservice architecture,» junio 2015. [En línea]. Available: https://dokumen.tips/documents/investigate-availability-and-maintainability-within-a-microservice-.html. [Último acceso: marzo 2020].

[18] D. Preuveneers y W. Joosen, «Towards multi-party policy-based access control in federations of cloud and edge microservices,» *IEEE European Symposium on Security and Privacy Workshops,* pp. 29-38, 2019.

[19] A. Bánáti, E. Kail, M. Kozlovszky y K. Karoczkai, «Authentication and authorization orchestrator for microservice-based software architectures,» *41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO),* pp. 1180-1184, 2018.

[20] auth0.com, «JSON Web Tokens Introduction,» auth0, 219. [En línea]. Available: https://jwt.io/introduction/. [Último acceso: marzo 2020].

[21] S. Newman, Building Microservices, 1st ed., O'Reilly Media, Inc., 2015.

[22] S. Parker, «Best Languages for Microservices,» Medium, 17 septiembre 2019. [En línea]. Available: https://blog.bitsrc.io/best-languages-for-microservices-fc721f10d8a0. [Último acceso: mayo 2020].

[23] M. Haughn, «Domain-driven design (DDD),» TechTarget Network, enero 2018. [En línea]. Available: https://whatis.techtarget.com/definition/domain-driven-design. [Último acceso: abril 2020].

[24] M. Rouse, «Single-page application (SPA),» TechTarget Network, diciembre 2016. [En línea]. Available: https://whatis.techtarget.com/definition/single-page-application-SPA. [Último acceso: abril 2020].

[25] R. Martin, Clean Code, Prentice Hall, 2008.