UNIVERSIDAD NACIONAL DE SAN CRISTÓBAL DE HUAMANGA

FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS:

Protocolo OpenID Connect para gestionar la seguridad en las APIs expuestas por Swagger, 2024.

Para optar el título profesional de:

INGENIERA DE SISTEMAS

PRESENTADO POR:

Bach. Yovana RONDINEL PEREZ

ASESOR:

Ing. Elinar CARRILLO RIVEROS

AYACUCHO - PERÚ

2024

PÁGINA DE CONFORMIDAD

Dedicatoria

A Dios por permitirme levantarme con vida y fortaleza cada día, para seguir luchando y lograr mis metas.

A mis padres quienes están en cada etapa de mi vida demostrándome su apoyo incondicional, confiando en mí a pesar de mis adversidades.

A mí a pesar de las dificultades que hay, siempre busqué lo mejor para mí.

Agradecimiento

Agradezco a mis padres y hermanos por ser el pilar fundamental sin el apoyo y la comprensión la presentación de este trabajo no sería posible.

Agradezco a la Universidad Nacional de San Cristóbal de Huamanga, mi alma mater, por acogerme en sus aulas y permitirme ser una excelente profesional.

A la Escuela Profesional de Ingeniería de Sistemas y a los docentes que con su enseñanza han contribuido en el desarrollo de mi formación profesional y personal.

Agradezco la asesoría de la Ing. Elinar Carrillo Riveros, que con gran generosidad ha sido mi guía con su experiencia y conocimiento para culminar la presente investigación y lograr mi objetivo.

Resumen

En la actualidad el crecimiento en la utilización de las interfaces de programación de aplicaciones (APIs) y el interés de las compañías en adoptar soluciones de gestión de estas APIs han provocado un notable incremento en la cantidad de datos procesados por estas APIs, lo que a su vez ha generado un aumento en la vulnerabilidad a ataques de seguridad. Por lo cual es necesario implementar protocolos de seguridad para salvaguardar la información de las empresas y organizaciones para que estas informaciones se encuentren seguras y protegidas, evitando así el acceso a usuarios y personas no autorizadas a dicha información.

El propósito de este estudio fue implementar el protocolo openid connect para gestionar la seguridad en las apis expuestas por swagger, con el fin de fortalecer la seguridad de los microservicios, la investigación se realizó en la Dirección Regional de Salud Ayacucho, Perú 2024.

Se realizó una investigación de tipo aplicada, porque abordó un problema práctico con el objetivo de generar un nuevo conocimiento, no se manipuló ni hubo intervención directamente en las variables, por consiguiente, es una investigación con un diseño no experimental.

En este estudio se aplicó el marco de trabajo Scrum para planificar la implementación del protocolo openid connect para gestionar la seguridad en las apis expuestas por swagger y para la recolección de datos se aplicó análisis de datos.

Palabras Clave: Api, Openid Connect, Swagger

Abstract

Currently, the growth in the use of application programming interfaces (APIs) and the interest of companies in adopting management solutions for these APIs has led to a significant increase in the amount of data processed by these APIs, which in turn has generated an increase in vulnerability to security attacks. It is therefore necessary to implement security protocols to safeguard the information of companies and organizations so that this information is safe and protected, this preventing unauthorized users and individuals from accessing this information.

The purpose of this study was to implement the openid connect protocol to manage security in the apis exposed by swagger, in order to strengthen the security of microservices, the research was conducted in the Regional Health Directorate of Ayacucho, Perú 2024.

This was an applied type of research, because it addressed a practical problem with the objective of generating new knowledge, there was no manipulation or direct intervention in the variables, therefore, it is a research with a non-experimental design.

In this study, the Scrum framework was applied to planning the implementation of the openid connect protocol to manage security in the apis exposed by swagger and data analysis was applied for data collection.

Keywords: Api, Openid Connect, Swagger

Índice

Lista de Tablas

Lista de Figuras

CAPÍTULO I

INTRODUCCIÓN

1.1. PLANTEAMIENTO DEL PROBLEMA

Actualmente, se tiene un incremento notable en el uso de APIs y el interés de las empresas por implementar plataformas de gestión de APIs. Esto ha llevado a un significativo incremento en la cantidad de información que se procesa a través de estas APIs, lo que, a su vez, ha dado lugar a un incremento en los ataques dirigidos a estas APIs (Noelia Martín, 2018).

La APIs son fundamentales para la arquitectura de microservicios, estos microservicios son la descomposición de las aplicaciones monolíticas y la comunicación entre ellos es a través de las APIs, las cuales emplean diversos protocolos (http, grpc, graphql, web-socket), las cuales son vulnerables a ataques de seguridad y por ellos es necesario implementar medidas de seguridad para proteger la información que contienen estos microservicios.

El propósito del estudio es encontrar una manera más efectiva de proteger la información en las APIs expuestas por Swagger en una organización, utilizando OpenID Connect. Para lograr esto, se deben considerar diversos aspectos, Cómo validar la identidad de los usuarios de las APIs y asignar un grado de confianza único a cada uno de ellos.

El problema existente en la Dirección Regional de Salud Ayacucho es la deficiente administración y control de seguridad de las APIs existentes, estas APIs están expuestas sin medida de seguridad y ello implica que cualquier usuario con poca experiencia pueda manipular la información, como modificar, eliminar o agregar registros, así mismo pueda acceder a la data sensible para su propio beneficio, la data

custodiada es sensible y por ello debe ser protegida mediante el empleo de los protocolos de seguridad, al exponer las APIs sin medidas de seguridad afectan la integridad de la información, la confidencialidad de los datos sensibles y, por ende, la reputación de la Dirección Regional de Salud Ayacucho, es importante establecer políticas de acceso adecuadas para garantizar que solo usuarios autorizados tengan acceso a la información, reduciendo así el riesgo de manipulación indebida.

Por lo tanto, no existe una manera eficiente de proteger las APIs y la información sensible en la Dirección Regional de Salud de Ayacucho.

1.2. FORMULACIÓN DEL PROBLEMA

1.2.1. Problema general

¿Cómo el protocolo Openid Connect gestiona la seguridad en las Apis expuestas por Swagger, 2024?

1.2.2. Problemas específicos

a. ¿De qué manera usar el Access Token para gestionar la seguridad en las Apis expuestas por Swagger?
b. ¿Cómo generar el Refresh Token para gestionar la seguridad en las Apis expuestas por Swagger?
c. ¿Cómo el Redirect_uri gestiona la seguridad en las Apis expuestas por Swagger?

1.3. JUSTIFICACIÓN DE LA INVESTIGACIÓN

La motivación de este trabajo surge por comprender, desarrollar y mejorar la seguridad en la API de la (DIRESA). Esto tiene como objetivo contribuir a satisfacer una necesidad primordial y brindar beneficios tanto a los actores directos como indirectos de la institución. Además, busco lograr esto utilizando los recursos tecnológicos disponibles de una manera óptima, asegurando la calidad, continuidad, cobertura, tiempo y costos razonables.

Es de vital importancia para las organizaciones garantizar que solo las aplicaciones y usuarios autorizadas, tanto internas como externas, tengan acceso a la información protegida.

De la misma manera, el de OpenID Connect es crucial para establecer una gestión de seguridad robusta y escalable en entornos de aplicaciones distribuidas. Proporciona una capa de autenticación segura, define estándares para la comunicación y ofrece flexibilidad para integrar aplicaciones y servicios de manera efectiva.

Se pretende implementar mecanismos de seguridad (Token de tipo openid connect) a las APIS (applications program interface) que están expuestas por SWAGGER, estas APIS por lo general no implementan ninguna seguridad y son vulnerables a ataques de DOS (denegación de servicio) y cualquier persona con poco conocimiento en tecnologías de información podría manipular la información mediante esta herramienta.

La propuesta consiste en utilizar el protocolo OpenID Connect para implementar mecanismos de seguridad que habilita el control del acceso de cada aplicación a los recursos del sistema. Este enfoque resulta altamente beneficioso, ya que este marco ofrece varias secuencias de autorización según el tipo de acceso que se requiere para cada solicitud de API.

1.4. LIMITACIONES DE LA INVESTIGACIÓN

Se propone llevar a cabo un estudio sobre la seguridad de APIs expuestas a través de Swagger, dirigido a los Microservicios de la Dirección Regional de Salud Ayacucho.

1.5. OBJETIVOS

1.5.1. Objetivo general

Implementar el protocolo Openid Connect para gestionar la seguridad en las Apis expuestas por Swagger, 2024.

1.5.2. Objetivo específico

a. Definir el alcance de la autorización del Access Token para gestionar la seguridad en las Apis expuestas por Swagger.
b. Configuración del Refresh Token para gestionar la seguridad en las Apis expuestas por Swagger.
c. Determinar el alcance de Redirect_uri para gestionar la seguridad en las Apis expuestas por Swagger.

CAPÍTULO II

MARCO TEÓRICO

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

Según Sujanani & Vinod (2018) en su artículo “Implementación de OpenId connect y OAuth 2.0 para crear SSO para institutos educativos”, publicado en la revista Internacional de Ingeniería y Tecnología, el objetivo fue establecer las distinciones entre el sistema convencional y la propuesta de inicio de sesión mediante openid connect, y, llegaron a la conclusión de OpenId connect permitió un acceso más eficiente a las diferentes aplicaciones, ya que los usuarios necesitaron autenticarse solo una vez para acceder a múltiples servicios de estas aplicaciones.

De acuerdo a Rushdy et al. (2021) en su artículo “Marco para proteger token web OAuth 2.0 y JSON para la API”, publicado en la revista de Tecnología de Información Teórica y Aplicada, el objetivo de esta investigación fue plantear una estrategia de JWT (Json Web Tokens) con OAuth 2.0 en una API REST, con el fin de posibilitar que la aplicación pueda validar el token de acceso OAuth2, reduciendo la necesidad de interactuar para lograr un nuevo token de acceso OAuth2 y concluyó que los resultados obtenidos demuestran que el empleo de encriptación RS256 no da un resultado favorable a comparación con la encriptación RS256 en cuanto a la evaluación de los algoritmos propuestos.

Así mismo Li & Mitchell (2020) en su artículo “Privacidad de acceso de usuarios en OAuth 2.0 y OpenID Connect”, publicado por el Instituto de Ingenieros Eléctricos y Electrónicos Inc., el objetivo fue hacer un análisis sistemático las características de privacidad relacionadas con acceso de los usuarios a los sistemas OAuth 2.0 y OpenID Connect. Además, se describe la facilidad con la que un proveedor de identidades puede rastrear los accesos realizados por los usuarios, concluyeron que para garantizar una verdadera protección de la privacidad es necesario realizar modificaciones sustanciales

en el funcionamiento de los protocolos OpenId Connect.

Además Singh & Chaudhary (2022) en su artículo “OAuth 2.0: Diseño arquitectónico para mitigar vulnerabilidades de seguridades comunes”, publicado en la revista de Seguridad de la Información y Aplicaciones, el objetivo fue analizar el diseño arquitectónico y mejorar la eficacia general de la seguridad del protocolo OAuth 2.0. Concluyeron que incorporando doble factor de autenticación (2FA) como característica para la autenticación de aplicaciones cliente de terceros mejoró significativamente la seguridad de las aplicaciones.

2.2. MARCO CONCEPTUAL

2.2.1. OpenId Connect

Según Rasiwasia (2018) el OpenIdI Connect es una forma de autenticación sencillo en OAuth 2.0 que autoriza a las aplicaciones validar la información del usuario a través del proceso de autenticación hecho por los proveedores de OpenID, y también menciona que hay más de 500 millones de cuentas de usuario basadas en OIDC, como PayPal, Microsoft y Google. Además Copete (2019) afirma que OpenID Connect se fundamenta en estándares abiertos y aprovecha OAuth para preservar los servicios y recursos. Mediante la solicitud de un código (token de acceso, Refresh Token, Redirect_uri), OpenID Connect proporciona información sobre la identidad del solicitante y el alcance de acceso a los recursos. Por otra parte Duran Castañeda (2020) afirma que OpenID Connect es un protocolo de código abierto ampliamente adoptado, lanzado por copOpenID Foundation en febrero de 2014, que establece una manera compatibilidad colaborativa de emplear OAuth 2.0 para autenticar a usuarios.

De otra manera Copete (2019) define que el OpenID Connect es un servicio de identidad, que utilizaremos para autenticar a los interesados finales que puede ser utilizado en dos direcciones: la primera dirección como proveedores del servicio, garantizamos la identidad del usuario y proporcionar a terceros una interfaz de estándar abierto; y la segunda como usuario del servicio, no tenemos que mantener y asegurar un perfil de un usuario y su contraseña centrándonos en ofrecer la funcionalidad de nuestro producto.

De la misma manera Copete (2019) afirma las ventajas de OpenID Connect con las siguientes secuencias : brinda un protocolo estándar para autenticar la identidad de un usuario final, se fundamenta en protocolos de código abierto como OAuth y es

independiente de la aplicación permitiendo realizar implementaciones e integraciones de manera más sencilla y con menos esfuerzo, posibilita la utilización de diversos mecanismos de seguridad, facilita el intercambio de información del perfil del usuario, siempre y cuando haya obtenido el consentimiento explícito del usuario final, la cantidad de información compartida dependerá de lo que previamente el usuario haya autorizado. De la misma forma Paradigma (2018) nos menciona las ventajas de OpenID Connect de la siguiente manera: permite autenticar a los usuarios sin la necesidad de almacenar ni gestionar contraseñas en aplicaciones, facilita la administración de una instancia OAuth propia que permita el acceso a la información.

Por otra parte OpenID (2020) menciona el funcionamiento del OpenID Connect con el siguiente orden: primero el usuario accede a un sitio web o navegador web, segundo el usuario ingresa a la opción "iniciar sesión" proporciona nombre y contraseña de usuario, tercero el Cliente (RP) envía una petición al Proveedor OpenID (OP), cuarto el Proveedor OpenID verifica la identidad del Usuario y obtiene la autorización necesaria, quinto el Proveedor OpenID responde proporcionando un token de identidad, sexto el Cliente tiene la opción de enviar una solicitud al dispositivo del Usuario junto con el Token de Acceso, y por último el endpoint de información del usuario devuelve datos o reclamos sobre el usuario final. De otra manera Guzmán et al. (2018) afirma sobre el funcionamiento del OpenID Connect de la siguiente manera: la contraseña única se requiere al iniciar sesión en un recurso protegido y se almacena por un período específico de tiempo para esa sesión en particular, esto permite que el usuario pueda navegar por los recursos protegidos sin que se le solicite volver a ingresar una nueva contraseña generada por el dispositivo.

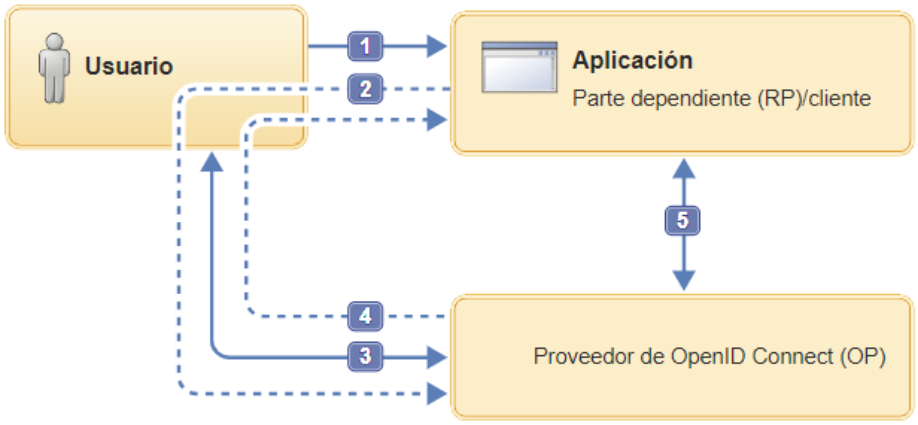
Además OpenID (2020) menciona sobre la relación de OAuth 2.0 con OpenID Connect, OAuth 2.0 es un conjunto de reglas definidas por el IETF en los RFC 6750 y 6749, con el objetivo de facilitar la creación de protocolos de autorización y autenticación, este marco ofrece varios flujos de mensajes estandarizados basados en HTTP y JSON, y OpenID Connect los aprovecha para brindar servicios de identidad. De otra manera Paradigma (2018) menciona la relación que tiene OAuth con OIDC, este es un protocolo de código abierto que posibilita la autenticación de usuarios a través de solicitudes HTTP en formato JSON y un protocolo de autenticación de usuarios que se fundamenta en OpenID y amplía OAuth 2.0, proporcionándole una capa adicional para autenticar usuarios sin tener que almacenar ni gestionar contraseñas. Así mismo Rafael Neto (2023) también menciona sobre la relación que tiene OAuth con OpenId Connect, OAuth 2.0 es para la autorización que consiste en otorgar acceso a las API y obtener datos de

usuario en otros sistemas, así mismo, utilizar OpenID Connect para la autenticación que permite registrar al usuario y hacer que esté disponible en otros sistemas.

Así mismo IBM (2023) afirma que OpenID Connect es un protocolo basado en OAuth 2.0 que ofrece una forma sencilla y estándar de identificación para permitir a las aplicaciones cliente confiar en la autenticación proporcionada por OpenID Connect, de esta manera identifica al usuario. Se describe el flujo de OpenID Connect en el siguiente diagrama (Figura 1).

Figura 1

Flujo de Autorización de OpenID Connect



Nota. Adaptado de Servidor de *aplicaciones Liberty de WebSphere*, IBM (2023)

2.2.1.1. Access Token

Allen & Baier (2022) afirma que un token de acceso es utilizado para adquirir acceso a un recurso de la API, los clientes solicitan y envían tokens de acceso a la API, estos tokens contienen información sobre el cliente, y si corresponde al usuario asociado, la API utiliza esta información para autorizar el acceso a sus datos. Dicho de otra manera, un token de acceso se utiliza en la autenticación basada en token para que la aplicación pueda ingresar a la API, el sistema recibe el token de acceso después de que el usuario se autentique con éxito, este token de acceso es utilizado para llamar a la API destino y realizar una acción especifica (auth, 2023). Además Feria (2018) Indica que un token de acceso es un elemento que proporciona detalles sobre el entorno de protección de un procedimiento. Un token de acceso guarda recurso de identidad y autorizaciones de una cuenta de usuario.

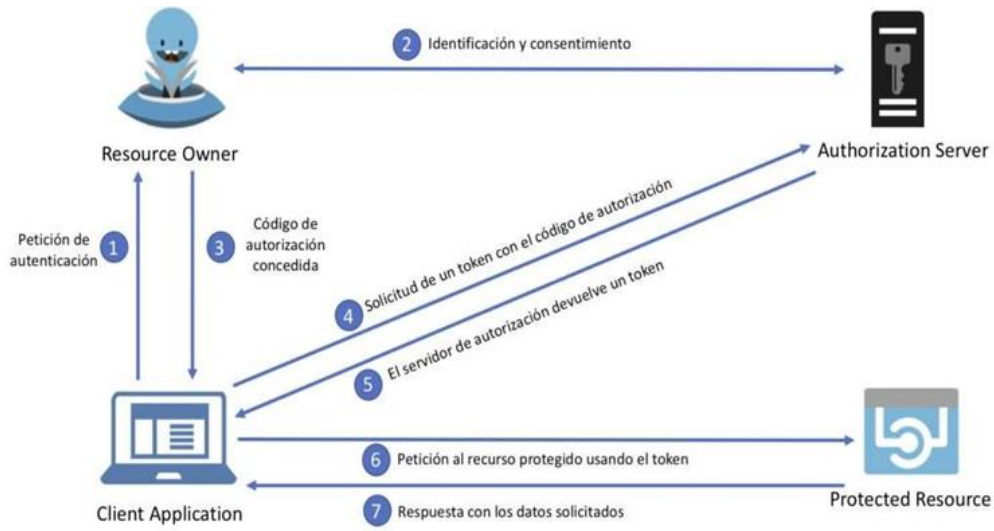
Por otra parte Okta (2023) afirma la funcionalidad de los tokens de acceso con el siguiente orden: inicio de sesión utilizando las credenciales del usuario previamente establecidos con el fin de verificar su identidad; verificación, el servidor valida la información y genera un token de autenticación; almacenamiento, el token es transferido y guardado en su navegador; comunicación, en cada ocasión en que usted accede a un nuevo elemento en el servidor se realiza una nueva verificación de su token; por último la eliminación, al concluir su sesión el token es descartado. De la misma manera Francisco Sánchez (2018) describe el funcionamiento del token de acceso con el siguiente orden: el usuario ingresa su nombre y contraseña en la aplicación; el usuario ingresa al sistema con contraseña y nombre de usuario; el servidor revisa credenciales del usuario, si son válidas crea y encripta un Access token este responde el token de acceso en el cuerpo del payload; el navegador o aplicación cliente genera un recurso como petición e inserta el token en el encabezado de la petición; el token es enviado en el encabezado de autorización del recurso generado; el servidor revisa las claves del token de acceso y obtiene la información del usuario del payload; el servidor responde con el recurso solicitado y por último el recurso es visualizado por la aplicación cliente.

También Okta (2023) afirma que para la petición o solicitud del token de acceso se incluirá estos parámetros: grant_type (requerido), este parámetro debe ser establecido como refresh_token; refresh_token (requerido), el token de actualización o renovación generado anteriormente para el cliente; Scope (opcional), el ámbito solicitado no debe contener ámbitos adicionales que no hayan sido incluidos en el token de acceso original; Autenticación del cliente, los tokens de actualización se emplean únicamente con clientes que requieren confidencialidad. De la misma forma García Delgado et al. (2018) afirma los siguientes parámetros: grant_type como tipo de autorización; el refresh_token para obtener un nuevo token de acceso; y finalmente el State, es utilizado por el cliente para rastrear la solicitud y la respuesta en el tiempo.

Según Torres (2019) presenta esta propuesta de solicitud de un Access Token, como evidencia se muestra en la figura (Figura 2).

Figura 2

Solicitud de Token de Acceso



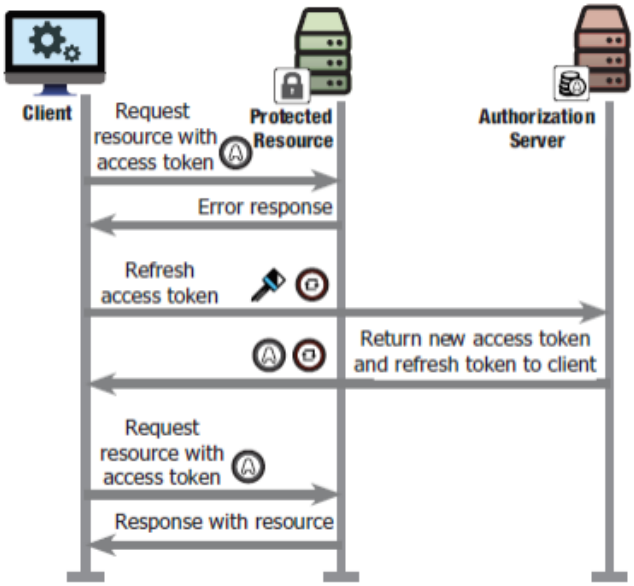
Nota. Adaptado de returngis, Torres (2019)

2.2.1.2. Refresh token

Duran Castañeda (2020) afirma que un Refresh Token en OAuth es comparable al Access token en términos de concepto, cuando es generado por el servidor de consentimiento al cliente, y, sin embargo, el cliente no tiene ni conocimiento ni el interés en el contenido del token. Sin embargo, la diferencia radica cunado el token de actualización nunca se envía al recurso resguardado, en su lugar se utiliza el Refresh Token para gestionar un nuevo tokens de acceso sin la necesidad de implicar al propietario del recurso.También se describe el Procedimiento de cómo renovar el token de acceso mediante el uso del Refresh Token como evidencia se muestra en la figura (Figura 3). Así mismo Feria (2018) afirma que el Refresh Token comparte una percepción similar con el Access Token, este es usado para adquirir un nuevo Access Token, evitando así la necesidad de solicitar usuario y contraseña nuevamente. De otra manera Hugo Pagola (2018) menciona que el Refresh Token es una cadena entregada por el servidor de autorización a la aplicación, que le permite solicitar nuevos Access Tokens una vez que los actuales hayan expirado dentro de un intervalo de tiempo ajustable. De esta manera, se evita que el sistema tenga que pedir nuevamente las credenciales al usuario.

Figura 3

Procedimiento de cómo renovar el Access Token mediante el uso del Refresh Token



Fuente: Duran Castañeda (2020, P.16)

Para Spasovski (2019) un token de actualización es un tipo de token generada por el servidor de autorización, así mismo existe un contraste con el acceso, la presencia de este tipo de token es opcional dentro del proceso de gestión de autorización, se aclara que este token tiene la finalidad de brindar a los clientes la oportunidad de expandir la aprobación preservada creando una nueva nota de acceso, es vital salvaguardar estos tokens (custodiar), ya que su exposición representa un riesgo de seguridad. También Dan Arias y Sam Bellen (2021) afirman que los tokens de acceso son válidos por poco tiempo por razones de seguridad. Cuando caducan, las aplicaciones de los clientes pueden "actualizar" el token de acceso utilizando token de actualización. En otras palabras, token de actualización es un artefacto de una calificación que permite a las aplicaciones de los clientes obtener un nuevo token de acceso sin solicitar al usuario volverse a conectarse.

Además Curity (2023) menciona sobre el funcionamiento y la caducidad de los tokens de actualización con las siguientes secuencias: los tokens de actualización podrían considerarse como una especie de contraseña, cuando accedes a un sitio web y aún no tienes una sesión activa, inicias sesión proporcionando algún tipo de credencial, como una contraseña; un token de actualización se puede utilizar para obtener otros tokens, conocidos como tokens de acceso, que tienen una duración más limitada; es relevante

porque una aplicación cliente a menudo deseará extender el período de acceso de un usuario sin requerir que este inicie sesión nuevamente, mediante el uso de un token de actualización, el cliente puede adquirir un token de acceso nuevo sin que el usuario intervenga, mejorando la experiencia del usuario (UX); la vida útil de los tokens de actualización se establece en el servidor de identidad para cada aplicación cliente, no hay un tiempo fijo, pero suele representar el período antes de que un usuario deba autenticarse nuevamente, para los tokens de actualización con información de alto valor generalmente se establece un tiempo más reducido.

Finalmente, el token de actualización lo utiliza el Cliente cuando el actual ha sido invalidado para solicitar un nuevo Access token.

2.2.1.3. Redirect_uri

Datatracker (2020) afirma que un URI de redirección ayuda a detectar clientes malintencionados y evita ataques de tipo phishing que intentan engañar al usuario para que crea que el phisher es el cliente o usuario, el valor del URI de redireccionamiento real utilizado en la solicitud de autorización debe presentarse y verificarse cuando se intercambia un "código" de autorización por tokens. Esto ayuda a prevenir ataques en los que el "código" de autorización se revela a través de redireccionadores y clientes de aplicaciones web falsificados. El servidor de autorización debe exigir a los clientes públicos y confidenciales que utilicen el tipo de concesión implícita que registren previamente sus URI de redireccionamiento y validen con el URI de redireccionamiento registrado en la solicitud de autorización. Dicho de otra manera Authorization (2023) afirma que el servidor de la API redirecciona al usuario después de que completa el flujo de autorización, el valor debe coincidir exactamente con uno de los URI de redireccionamiento autorizados para el cliente OAuth 2.0, que configuraste en la consola de registro de las aplicaciones cliente.

De acuerdo con Microsoft (2023) un URI de redireccionamiento, también denominado URL de respuesta, es la dirección a la cual el servidor de autorización dirige al usuario una vez que la aplicación ha sido autorizada con éxito y se le ha otorgado una clave de autorización. Es fundamental que la ubicación correcta sea registrada durante el proceso de registro de la aplicación, ya que es en el URI de redireccionamiento donde el servidor de autorización envía el código o token.

Finalmente, Redirect_uri es la redirección de retorno a la aplicación cliente autorizada

después del proceso de iniciar sesión.

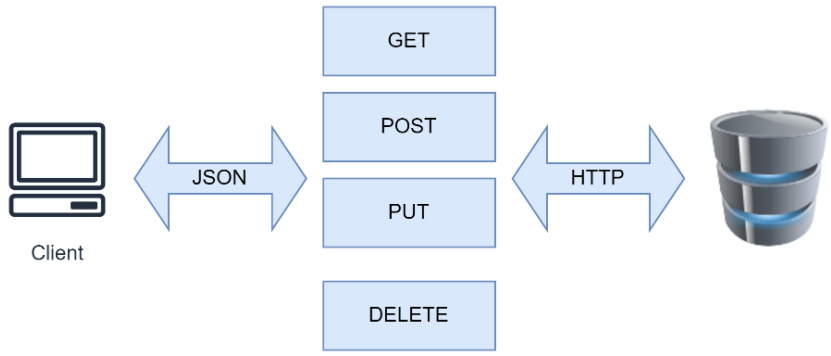
2.2.2. API

Según De la Fuente García (2020) una API es un mecanismo que agiliza la comunicación y la reciprocidad de datos entre aplicaciones, en el ámbito de los Datos Abiertos, este concepto suele hacer referencia a las APIs web, también denominadas API web, las cuales posibilitan la reciprocidad o el intercambio de información tanto dentro de una organización como entre diferentes organizaciones. De la misma manera Casal et al. (2019) afirma que una API es un mecanismo que acelera el intercambio y la comunicación de datos entre las aplicaciones en el área de datos abiertos, este concepto generalmente se refiere a Web -APIS, que también se conoce como API -WEB y permite el intercambio de información tanto dentro de una organización como entre diferentes organizaciones. Además, en HTTP los métodos más utilizados se denominan como: POST, GET, PUT y DELETE. Corresponden a operaciones de creación, consultar, actualización y eliminación, estos métodos son utilizados por estas APIs de servicio (Avior.API, 2022).

De acuerdo con la propuesta de Alejandra y Jonathan (2021) se presenta la arquitectura de los métodos HTTP de las APIs, que se evidencia en la (Figura 4).

Figura 4

Arquitectura de métodos http de las Apis



Fuente: Alejandra y Jonathan (2021, P. 21)

AWS (2023) afirma sobre la funcionalidad de las APIs, se describen en términos de cliente y servidor, el cliente es la aplicación que realiza la solicitud, mientras que el servidor es la que envía la respuesta a las peticiones realizadas por el cliente. Las APIS pueden operar de diversas formas, dependiendo del momento y el propósito de su desarrollo, existen tres categorías principales y estas son: la primera categoría API

REST, estas API se conocen como llamadas a procedimientos remotos en este tipo de interacción, el cliente solicita la ejecución cierta lógica o negocio en el servidor luego el servidor retorna el resultado al cliente solicitante; la segunda categoría es SOAP, este tipo de servicios hacen uso del protocolo Simple Object Access, en este caso el cliente y el servidor se comunican intercambiando mensajes en formato XML; y el tercero es WebSocket, estos son una innovación moderna en el ámbito de las API web donde se utilizan objetos JSON para la transmisión de datos permitiendo una comunicación bidireccional entre las aplicaciones cliente y el servidor.

Además SYDLE (2022) afirma sus principales ventajas de una API mencionadas a continuación: como la eficiencia, los datos se pueden distribuir de manera más eficiente al tener contenido que se publica automáticamente y se hace disponible en varios canales al mismo tiempo adaptabilidad, otra ventaja de la API es su capacidad para adaptarse a los cambios mediante la migración de datos y la flexibilidad de los servicios; alcance, a través de ellas es posible crear capas de aplicaciones con el objetivo de distribuir información a diferentes audiencias; aplicaciones, el uso de las APIs proporciona una mayor flexibilidad en los procesos de transferencia de información; y la personalización, pueden funcionar como una solución para crear experiencias personalizadas para los usuarios, al permitir adaptar protocolos, funciones y comandos según los requisitos específicos.

En pocas palabras las APIs son una parte del código que permite compartir información y comunicarse entre diferentes aplicaciones.

2.2.2.1. Get

Oscar Blancarte (2018) afirma que se utiliza exclusivamente para solicitar datos a los servidores, similar al ejecutar una consulta SELECT en la base de datos. No permite la transmisión del payload. Dicho de otra manera López & Lucía (2018) afirman que GET se utiliza para adquirir la representación de una información sin realizar modificaciones o alteraciones en él. Una respuesta satisfactoria normalmente en formato JSON o XML, con el código 200 que indica "OK". Si ocurre alguna falla, se devuelve el código 404 (no encontrado) o 400 (solicitud incorrecta), el método GET es una operación segura e idempotente.

Por otra parte IONOS (2020) afirma que GET se utiliza para obtener la representación de un recurso sin realizar modificaciones en él. Una respuesta satisfactoria en formato

JSON o XML, con el código 200 que indica "OK". Si ocurre alguna falla, se devuelve el código 404 (no encontrado) o 400 (solicitud incorrecta), el método GET es una operación segura e idempotente.

Según IONOS (2020) el método GET presenta algunas desventajas importantes. En primer lugar, su protección de datos es limitada y los parámetros URL enviados son visibles en la url del navegador. En segundo lugar, su capacidad restringida, dado que, dependiendo del servidor y el navegador, no se puede ingresar más de 2000 caracteres. Además, los parámetros URL solo admiten caracteres ASCII (signos, números, letras) y no permiten datos binarios, como archivos de audio o imágenes.

Además Williams Morales (2023) afirma que el método GET como método HTTP efectúa una solicitud a un endpoint, a diferencia de otros métodos, no posibilita el envío de datos, salvo que sean incluidos como parámetros en la URL de la solicitud; esta solicitud devolverá tanto la cabecera como el contenido de la respuesta. Además, el método GET puede devolver la respuesta en diversos formatos, como HTML, JSON, XML, imágenes o JavaScript, su propósito principal es consultar información, similar a una instrucción SELECT en una base de datos, los datos enviados a través de la URL pueden utilizarse para filtrar datos durante la consulta. También describe la respuesta del Método Get en HTTP en el siguiente cuadro (Tabla 1) y las características del Método Get en HTTP en el siguiente cuadro (Tabla 2).

Tabla 1

Respuesta del Método Get en HTTP

200	Respuesta correcta
400	Petición incorrecta
404	Recurso no encontrado

Fuente: (Williams Morales, 2023)

Tabla 2

Características del Método Get en HTTP

Petición con cuerpo	No
Respuesta con cuerpo	Si
Seguro	Si
Idempotente	Si
Cacheable	Si
Permitido en HTMLforms	Si

Fuente: (Williams Morales, 2023)

Así mismo GET, lee y consulta un recurso, recupera una representación de un recurso SEGURO – IDEMPOTENTE, varias peticiones idénticas devuelven el mismo resultado (Avior.API, 2022).

2.2.2.2. Post

Oscar Blancarte (2018) afirma que se utiliza para solicitar la generación de un registro nuevo, es decir, algo que no estaba presente anteriormente, similar a realizar una inserción en la base de datos. Permite el envío del payload. Dicho otra manera López & Lucía (2018) afirma que es utilizado para crear recursos que están subordinados a otros. En otras palabras, se asigna una identificación al nuevo recurso que lo vincula con su componente principal. La respuesta devuelve el código 201 para indicar que se ha efectuado una creación exitosa. A diferencia de otros métodos, POST no es idempotente ni seguro, ya que el envío repetido de la petición puede conllevar la creación de múltiples recursos.

Por otra parte IONOS (2020) resalta la ventaja del método POST en la transferencia de datos, especialmente al completar formularios que contienen información confidencial como nombres de usuario y contraseñas. El método POST ofrece un alto nivel de confidencialidad, asegurando que los datos no queden visibles en la caché del navegador ni en su historial. Asimismo, la versatilidad del método POST es altamente beneficiosa, ya que no se limita solo al envío de textos breves, sino que también permite transferir otros tipos de información, como vídeos o fotos.

También Williams Morales (2023) afirma el método POST en HTTP posibilita la transmisión de datos hacia el servidor a través del cuerpo de solicitud, a diferencia del método GET que utiliza la URL, se utiliza semánticamente para registrar información, al igual que un comando INSERT en una base de datos, sin embargo, es importante tener en cuenta que se puede utilizar de manera forzada para otras acciones como actualización, eliminación de registros aunque se emplea para acciones que no implican registro de información, es importante resaltar que el método POST no es idempotente, para enviar los datos, el POST utiliza formularios, formato JSON, entre otros. También describe la respuesta del Método Get en HTTP en el siguiente cuadro (Tabla 3) y las características del Método Get en HTTP en el siguiente cuadro (Tabla 4).

Tabla 3

Respuesta del Método Post en HTTP

201	Creado
400	Petición incorrecta
404	Recurso no encontrado

Fuente: (Williams Morales, 2023)

Tabla 4

Características del Método Post en HTTP

Petición con cuerpo	Si
Respuesta con cuerpo	Si
Seguro	Si
Idempotente	No
Cacheable	Solo si incluye nueva información
Permitido en HTML forms	Si

Fuente: (Williams Morales, 2023)

Así mismo POST crea un recurso, es decir utiliza para crear nuevos recursos NO ES SEGURO - NO ES IDEMPOTENTE, hacer 2 POST idénticos duplicará el recurso (Avior.API, 2022).

2.2.2.3. Put

Oscar Blancarte (2018) afirma que se utiliza para llevar a cabo una actualización total de un dato existente, es decir, es similar a efectuar una actualización en la base de datos. Permite el envío del payload. Así mismo López & Lucía (2018) afirman que se utiliza para actualizar recursos, La información que alterará el recurso original se incluye en el contenido del mensaje. Algunos de estos servicios utilizan el PUT para generar un recurso nuevo si la identificación brindada por el usuario no coincide con los datos existentes y no es recomendable ya que puede generar desorden con POST, como también el contenido en la respuesta de la actualización no necesariamente es obligatoria, este método es IDEMPOTENTE (produce los mismos resultados) que significa que enviar la misma solicitud repetidamente no produce cambios adicionales, no obstante, no es seguro ya que altera el estado del recurso.

Además Williams Morales (2023) afirma que el método de solicitud PUT comparte similitudes con el método POST, aunque su principal diferencia radica en que PUT es idempotente, lo que implica que puede ejecutarse varias veces con el mismo resultado, a diferencia del método POST. Cada vez que se ejecuta el método PUT, se lleva a cabo una actualización de la información existente, similar a un comando UPDATE en una base de datos. En este sentido, PUT se utiliza semánticamente para modificar datos ya existentes. Por lo general, las solicitudes PUT envían datos en formato JSON, entre otras opciones. También describe la respuesta del Método Get en HTTP en el siguiente cuadro (Tabla 5) y las características del Método Get en HTTP en el siguiente cuadro (Tabla 6).

Tabla 5

Respuesta del Método Put en HTTP

201	Creado
204	No responde, si el servidor no devuelve ningún contenido
400	Petición incorrecta
404	Recurso no encontrado

Fuente: (Williams Morales, 2023)

Tabla 6

Características del Método Put en HTTP

Petición con cuerpo	Si
Respuesta con cuerpo	Si
Seguro	No
Idempotente	Si
Cacheable	No
Permiso en THML forms	No

Fuente: (Williams Morales, 2023)

Así mismo PUT, se entiende que Actualiza o modifica un recurso, NO SEGURO - IDEMPOTENTE Múltiples peticiones idénticas actualizarán el mismo recurso (Avior.API, 2022).

2.2.2.4. Delete

Oscar Blancarte (2018) afirma que este método se emplea para suprimir un registro, es análogo a la operación DELETE en una base de datos. No permite el envío del payload. De la misma manera López & Lucía (2018) afirman que se utiliza para eliminar o descartar un recurso identificado por su URI. Si la eliminación es exitosa, se devuelve el código 200 acompañado del cuerpo de la respuesta o con el código 204 si la respuesta carece de contenido. Sin embargo, el método DELETE no siempre se considera idempotente.

Además Williams Morales (2023) dice que este tipo de solicitud permite la eliminación de un recurso específico, también es idempotente, lo que implica que se puede ejecutar varias veces con el mismo resultado, al igual que el método PUT y GET, y su significado se enfoca en la eliminación de información existente, de forma similar a un comando DELETE en una base de datos. También describe la respuesta del Método Get en HTTP en el siguiente cuadro (Tabla 7) y las características del Método Get en HTTP en el siguiente cuadro (Tabla 8).

Tabla 7

Respuesta del Método Delete en HTTP

202	La acción fue realizada correctamente.
204	Respuesta correcta

Fuente: (Williams Morales, 2023)

Tabla 8

Características del método Delete en HTTP

Petición con cuerpo	Tal vez
Respuesta con cuerpo	Si
Seguro	No
Idempotente	Si
Cacheable	No
Permitido en HTML forms	No

Fuente: (Williams Morales, 2023)

Así mismo DELETE, elimina un recurso, NO SEGURO - IDEMPOTENTE Varias solicitudes idénticas no eliminarán el mismo recurso ya que la primera petición realizó la eliminación y las siguientes peticiones lanzarían error 404 (recurso no encontrado) (Avior.API, 2022).

2.3. MARCO REFERENCIAL

2.3.1. API REST

Una API REST sigue los principios de REST, permitiendo la creación de aplicaciones y servicios accesibles mediante el protocolo HTTP. Se basa en la arquitectura de transferencia de estado representacional y comúnmente se denomina API RESTful.

2.3.2. Swagger

Swagger es una plataforma online que facilita la creación de documentación detallada para APIs, ofreciendo un conjunto de herramientas de apoyo al desarrollador. Facilita el acceso, visualización de la estructura y datos necesarios para los puntos finales de la API, permitiendo proporciona pruebas directas en su interfaz.

2.3.3. Seguridad

La seguridad es la capacidad de los sistemas de información para oponer resistencia ataques con una confianza significativa, es decir, específicamente al manejo de datos para prevenir su manipulación por parte de personas sin autorización, previniendo los incidentes o acciones maliciosas.

2.3.4. Autenticación

La autenticación consiste en verificar la identidad de un usuario mediante contraseñas, tokens u otros datos. El inicio de sesión único (single-sign-on) es un protocolo que permite la autenticación en varios sistemas con un único proceso de inicio de sesión.

2.3.5. Autorización

La autorización permite el acceso a ciertos recursos a un usuario que se haya autenticado previamente, es decir, determina las acciones que se te permiten realizar una vez que te has identificado. Está estrechamente relacionada con la autenticación y se utiliza para prevenir el uso no autorizado de recursos.

2.3.6. Microservicios

Los microservicios son una arquitectura de desarrollo que utiliza servicios autónomos comunicados a través de APIs definidas, descomponiendo aplicaciones en componentes independientes.

2.3.7. Control de Acceso

El control de acceso protege la información permitiendo su uso solo a usuarios autorizados con los permisos adecuados, asegurando la protección contra accesos no autorizados. Sus políticas se basan en técnicas como la verificación de identidad y la asignación de permisos para aplicar medidas de seguridad.

2.3.8. OAuth 2.0

OAuth 2.0 es un protocolo que proporciona la autorización permitiendo a las aplicaciones tener acceso limitado a cuentas de usuarios de forma segura, estándar y simple por parte de una aplicación hacia un usuario, mediante el uso de una API sin necesidad de que el usuario comparta sus credenciales de autenticación con la aplicación cliente.

2.3.9. JSON Web Token (JWT)

JSON Web Token (JWT) es una forma segura de representar peticiones entre dos partes a través de una URL, con afirmaciones codificadas en formato JSON. Utilizado en escenarios como el inicio de sesión único (SSO), cada token JWT contiene datos para la autenticación, siendo una estándar abierto ampliamente adoptado que permite la firma digital y su uso en navegadores web.

2.3.10. Client Credentials

Las credenciales del cliente se solicita para utilizar el Access token para ingresar a sus propios recursos, permitiendo que un sistema obtenga un token de acceso así ingresar a la información que pertenece al usuario, esta forma de autorización es adecuado para sistemas que requieren ingresar a las APIs como el servicio de almacenamiento, en nombre de la propia aplicación y no en nombre de dicho cliente.

2.3.11. Authorization Code

El código de autorización es una clave temporal que el cliente intercambiará por un token de acceso, este código se obtiene del servidor de autorización, donde el usuario tiene la oportunidad de ver la información que el cliente solicita y decidir si aprueba o rechaza la solicitud, en especial es adecuado para aplicaciones web en el servidor.

2.3.12. Servicios Web

Es una aplicación que tiene como propósito establecer comunicación con otras aplicaciones mediante la definición de reglas que incluyen direcciones para localizar recursos, acciones permitidas y la estructura del intercambio de información entre las aplicaciones.

2.3.13. Alcance

Un alcance se define como una representación de permisos para acceder a un recurso protegido, compuesta por cadenas combinadas en una lista separada por comas; estos desempeñan un papel crucial al restringir el acceso del cliente, siendo definidos por el recurso protegido según sus APIs asociadas.

CAPÍTULO III

MATERIALES Y MÉTODOS

3.1. TIPO Y NIVEL DE INVESTIGACIÓN

3.1.1. Tipo de investigación

Según Teodoro & Nieto (2018) La investigación aplicada se centra en mejorar, perfeccionar u optimizar el funcionamiento de sistemas, procedimientos, normas y reglas tecnológicas existentes a la luz de los avances en ciencia y tecnología. Por lo tanto, este tipo de investigación no se evalúa en términos de verdadero, falso o probable, sino en términos de eficiencia, deficiencia, ineficiencia, eficacia o ineficacia.

De acuerdo a las variables en el estudio la investigación fue Aplicada, porque las variables solo sirvieron para caracterizar fenómenos sin manipular variables y no están relacionadas.

3.1.2. Nivel de investigación

Supo et al. (2014) expone que una investigación de carácter descriptivo se basa en el método de análisis para caracterizar de manera detallada una situación específica, resaltando sus características y atributos. Su propósito fundamental radica en interpretar y describir la situación actual de las cosas. Asimismo, mediante el empleo de criterios de clasificación, facilita la organización, agrupación o sistematización de los elementos implicados en la investigación. Este tipo de investigación se enfoca en los hechos reales y busca presentar una interpretación precisa de los mismos.

El nivel de la presente investigación es descriptivo, porque los cambios que se realizaron en la implementación del protocolo Openid Connect para gestionar la seguridad en las Apis expuestas por Swagger, solo se describieron.

3.2. DISEÑO DE LA INVESTIGACIÓN

Según lo expresado por Marisela Dszul (2021) plantea que, en una investigación con diseño no experimental, no se lleva a cabo la manipulación intencional de variables. Su enfoque principal se dirige hacia la observación directa de fenómenos en su contexto natural, con el objetivo de someterlos a análisis posteriormente.

La investigación adopta un enfoque no experimental, ya que no se llevaron a cabo modificaciones deliberadas de las variables, solo sirvió para observar fenómenos en situaciones específicas.

3.3. VARIABLES

3.3.1. Definición conceptual de variables

Primera Variable de interés

Protocolo OpenId Connect: Es un marco de autenticación y autorización que posibilita a los usuarios acceder a sitios web utilizando credenciales de otros servicios, como Google o Facebook, asegurando la identidad y el acceso seguro.

Variables descriptivas (Dimensiones)

Access Token: Es un token de seguridad que se obtiene tras una autenticación exitosa y permite acceder a recursos protegidos en aplicaciones o servicios en línea, mejorando la seguridad y la autorización.

Refresh Token: Es una clave de actualización que se obtiene junto con un Access Token, permitiendo renovar o intercambiar el Access Token expirado sin requerir nuevas credenciales, mejorando la seguridad y la experiencia del usuario.

Redirect_uri: Es una URL a la que un servicio de autenticación redirige al usuario después de autorizar el acceso a una aplicación, permitiendo que esta reciba códigos de autorización o tokens para completar el proceso de autenticación.

Segunda Variable de interés

Apis: Las APIs son conjunto de normas y protocolos que posibilitan la comunicación y el intercambio de datos y capacidades entre diversas aplicaciones o sistemas, facilitando la integración y el desarrollo de software.

Variables descriptivas (Dimensiones)

Get: El método "GET" es una solicitud en protocolos HTTP/HTTPS utilizada para obtener información del servidor, permitiendo recuperar recursos como páginas web o datos de una API.

Post: El método "POST" es una solicitud en protocolos HTTP/HTTPS que envía datos desde el cliente al servidor, útil para enviar información como formularios o cargar archivos, sin mostrar los datos en la URL.

Put: El método "PUT" es una solicitud en protocolos HTTP/HTTPS que se emplea para actualizar o crear recursos en el servidor, permitiendo enviar datos para modificar o crear contenido en una base de datos.

Delete: El método "DELETE" es una solicitud en protocolos HTTP/HTTPS que solicita la eliminación de un recurso en el servidor, posibilitando borrar información, archivos u objetos almacenados en una aplicación o sitio web.

3.3.2. Definición operacional de variables

Primera Variable de interés

Protocolo OpenId Connect: Se implementa en una aplicación web mediante la configuración de flujos de autenticación y autorización, utilizando bibliotecas de software y servicios de proveedores como Google o Microsoft para habilitar la autenticación de usuarios a través de cuentas de terceros.

Variables descriptivas (Dimensiones)

Access Token: Se genera mediante el intercambio exitoso de credenciales de usuario con un proveedor de autenticación (como Google), y se incluye en las cabeceras de las solicitudes HTTP a recursos protegidos en una aplicación web, otorgando acceso autorizado a dichos recursos.

Refresh Token: Se adquiere junto con un Access Token durante la autenticación, y se almacena de manera segura en el cliente. Cuando el Access Token expira, el Refresh Token se utiliza para obtener un nuevo Access Token sin requerir que el usuario proporcione credenciales nuevamente.

Redirect_uri: Se configura en la aplicación como la URL a la que el proveedor de

autenticación redirige al usuario después de autorizar el acceso. Esta URL contiene información necesaria para que la aplicación reciba y procese los códigos de autorización o tokens.

Segunda Variable de interés

Apis: Las APIs se implementan mediante la exposición de endpoints (puntos finales) en un servidor web o servicio, cada uno asociado a un método HTTP específico (GET, POST, PUT, DELETE), que permite a las aplicaciones realizar operaciones y acceder a recursos específicos a través de solicitudes y respuestas HTTP.

Variables descriptivas (Dimensiones)

Get: Se aplica al método HTTP "GET" en una API, donde una solicitud "GET" realiza una petición de datos desde un recurso específico, como un servidor, y recibe una respuesta con los datos solicitados.

Post: Se aplica al método HTTP "POST" en una API, donde una solicitud "POST" envía datos desde el cliente al servidor, permitiendo la creación o modificación de recursos en el servidor.

Put: Se aplica al método HTTP "PUT" en una API, donde una solicitud "PUT" actualiza un recurso específico en el servidor con los datos proporcionados en la solicitud.

Delete: Se aplica al método HTTP "DELETE" en una API, donde una solicitud "DELETE" solicita la eliminación de un recurso específico en el servidor.

3.4. POBLACIÓN Y MUESTRA

3.4.1. Población

Para Marisela Dszul (2021) la población es el total de todas las unidades de estudio (sujetos u objetos) con características observables o respuestas que son objeto de interés para el estudio se denomina población. Es de vital importancia definir claramente las poblaciones en cuanto a su contenido, ubicación y período temporal, lo cual se denomina marco muestral.

La población está conformada por 10 Microservicios de la DIRESA en la Región de Ayacucho, 2024.

3.4.2. Muestra

Para Pedro Luis López (2014) El muestreo no probabilístico implica seleccionar elementos de una población sin utilizar un proceso aleatorio basándose en criterios específicos del investigador en lugar de asignar probabilidades conocidas, se utiliza cuando es complicado realizar un muestreo probabilístico, sin embargo, la falta de aleatoriedad en la selección puede generar distorsiones en la representación de la población.

Se tomó una muestra por conveniencia no probabilística de un Microservicio de la DIRESA (Dirección Regional de Salud Ayacucho).

3.5. OPERACIONALIZACIÓN DE LAS VARIABLES

Tabla 9

Operacionalización de variables

Variable	Dimensión	Indicador	Pregunta
protocolo OpenID Connect	Access Token	Llave para acceder a un recurso	¿El access token es utilizado como clave para acceder a los recursos protegidos?
			¿Se requiere enviar un Access token válido para obtener acceso a recursos específicos?
	Refresh Token	Actualiza el token	¿El refresh token de qué manera es utilizado para renovar el access token una vez haya expirado?
			¿El proceso de actualización del token garantiza la identidad y seguridad del usuario de manera efectiva?
	Redirect_uri	Permite definir el alcance de las aplicaciones	¿Permite el redirect_uri a las aplicaciones especificar los permisos y alcances necesarios durante la autenticación?
			¿El redirect_uri asegura que solo se activen los permisos

			solicitados durante la autenticación?
	Get	Obtener recurso	¿Las solicitudes GET son utilizadas para recuperar información o recursos específicos?
			¿Existe alguna convención o estándar específico que se siga al crear recursos de tipo GET para asegurar su coherencia y claridad?
	Post	Crear recurso	¿Las solicitudes POST se envían en el cuerpo de la petición para crear recursos?
			¿Las respuestas de las solicitudes POST contienen datos precisos y relevantes?
Apis Expuestas por Swagger	Put	Edita recurso	¿Las solicitudes PUT son empleadas para actualizar los datos de recursos existentes de manera segura y eficiente?
			¿Las solicitudes PUT siguen un proceso seguro para la actualización de datos?
	Delete	Elimina recurso	¿Las solicitudes DELETE son utilizadas para eliminar recursos de manera segura?
			¿Se implementa algún tipo de confirmación o medida de seguridad antes de proceder con la eliminación para evitar acciones no deseadas?

Fuente: Elaboración propia.

3.6. TÉCNICAS E INSTRUMENTOS DE LA INVESTIGACIÓN

3.6.1. Técnicas

De acuerdo con Bermeo et al. (2016) el análisis documental comprende la exploración,

elección, estructuración y examen de un conjunto de documentos escritos con el objetivo de abordar una o varias interrogantes vinculadas a un tema específico.

En el estudio se utilizó el Análisis Documental, la cual fue aplicado a expertos profesionales, tales como arquitectos de software y desarrolladores senior; con el propósito de recopilar tanto la información de entrada como la información de salida de manera exhaustiva y detallada.

3.6.2. Instrumentos

El instrumento es un registro de datos, con el cual se obtuvo información detallada del estado de la implementación del OpenId Connect en APIs.

3.6.3. Validez del instrumento

El estudio se llevó a cabo empleando el criterio de validez de contenido.

Según Urrutia Egaña et al. (2014) la validez de contenido describe a un juicio de carácter lógico sobre cómo concuerda el contenido del aprendizaje del evaluado con lo presentado en el examen. El propósito de este tipo de validez es determinar si los ítems o preguntas planteadas reflejan de forma apropiada el ámbito de contenido (destrezas conocimientos, habilidades) que se intenta evaluar. Para lograr este propósito de forma efectiva, es esencial obtener pruebas que respalden la calidad y relevancia técnica de la prueba, asegurándose de su representatividad mediante el uso de fuentes confiables, como la consulta de literatura especializada, la opinión de expertos o la participación de una población relevante.

Se llevó a cabo la validación de contenido en la investigación, donde las preguntas formuladas fueron sometidas a evaluación mediante el coeficiente de Aiken para asegurar su adecuada formulación.

En el anexo 3 se muestra el certificado que acredita la validez de contenido, mientras que en el anexo 4 se presentan los datos evaluados por los jueces y el cálculo del coeficiente V de Aiken. La validez del contenido del registro instrumental es 0.95 evidenciando un valor de relevancia.

3.6.4. Confiabilidad del instrumento

No se procedió a evaluar la confiabilidad del instrumento, el cual se basa en un análisis

documental.

3.7. PROCEDIMIENTOS

3.7.1. Estrategia de prueba de hipótesis

Dentro del marco de esta investigación, no se realiza ninguna prueba de hipótesis debido a que se trata de un estudio de nivel descriptivo.

3.7.2. Técnicas de procesamiento de datos

A. Técnicas para procesamiento de datos con el marco de trabajo Scrum

El enfoque de trabajo Scrum se empleó para la planificación de las historias de usuario para la implementación de OpenId Connect en swagger, las etapas se detallan en las tablas proporcionadas a continuación.

Tabla 10

Procesos en la Fase de Análisis

PROCESO	ENTRADA	HERRAMIENTA	SALIDA
Definición del producto	Visión del producto. Requerimientos iniciales.	Talleres de definición del producto. Entrevistas con stakeholders.	Documento de visión del producto. Lista inicial de necesidades.
Creación del Backlog del producto.	Documento de visión del producto. Reuniones con Stakeholders. Retroalimentación del equipo.	Reuniones de priorización. Herramientas de gestión de producto Backlog.	Product Backlog priorizado. Historias de usuario definidas.
Análisis de Historias de Usuario	Historias de usuario. Requisitos detallados. Conversaciones con el producto Owner.	Discusiones entre el equipo y el producto Owner.	Definición clara de Historias de Usuario. Criterios de aceptación.

Estimación de Historias de Usuario	Historias de usuario definidas. Equipo de desarrollo.	Reuniones de estimación. Técnicas de estimación.	Estimación para las historias de Usuario. Product Backlog refinado.

Fuente: Elaboración propia.

Tabla 11

Procesos en la Fase de planificación

PROCESO	ENTRADA	HERRAMIENTA	SALIDA
Planificación del Sprint	Lista de producto Backlog. Objetivos del Sprint. Capacidad del equipo.	Reunión de planificación del Sprint. Sprint Backlog.	Sprint Backlog actualizado. Objetivo del Sprint definido.
Definición de tareas	Sprint Backlog. Historias de usuario y sus criterios de aceptación.	Reuniones de planificación de sprint. Herramientas de gestión de tareas.	Lista de tareas detalladas. Estimaciones de tiempo para las tareas.
Estimación	Lista de tareas estimadas. Capacidad del equipo.	Reuniones de planificación del sprint. Criterios de estimación.	Tareas estimadas.
Compromiso	Tareas estimadas y priorizadas. Capacidad de equipo.	Reuniones de compromiso. Herramientas de planificación.	Compromiso del equipo para completar el Sprint. Sprint Backlog definitivo.

Fuente: Elaboración propia

Tabla 12

Procesos en la Fase de implementación

PROCESO	ENTRADA	HERRAMIENTA	SALIDA
Desarrollo Iterativo	Sprint Backlog. Historia de usuario definidas.	Herramientas de desarrollo. Herramientas de colaboración.	Código fuente. Artefactos desarrollados. Incremento del producto potencialmente entregable.
Reuniones Diarias (Daily Scrums)	Equipo de desarrollo. Avance de tareas y obstáculos.	Reuniones diarias breves (máximo 15 minutos). Tablero de tareas.	Comprensión comparativa del progreso. Obstáculos identificados y resueltos.
Revisión del Sprint	Incremento del producto. Historias de usuario completadas.	Reunión de revisión del Sprint. Herramientas de presentación.	Producto demostrado y revisado. Retroalimentación del Product Owner y Stakeholders.
Retrospectiva del Sprint	Resultados del Sprint. Obstáculos y mejoras identificados.	Reunión de retrospectiva del Sprint. Técnicas de retrospectiva.	Acciones de mejora definidas. Plan para implementar mejoras.

Fuente: Elaboración propia

Tabla 13

Procesos en la Fase de revisión y retrospectiva

PROCESO	ENTRADA	HERRAMIENTA	SALIDA
Revisión del sprint	Incremento del producto potencialmente entregable. Historias de usuarios completadas.	Reunión de revisión del Sprint. Herramientas de presentación.	Producto demostrado y revisado. Retroalimentación del producto Owner y Stakeholders.
Retrospectiva del sprint	Resultados del Sprint. Obstáculos y mejoras identificados.	Reunión de retrospectiva del Sprint. Técnicas de retrospectiva.	Acciones de mejora definidas. Plan para implementar mejoras.

Fuente: Elaboración propia

Tabla 14

Procesos en la Fase de lanzamiento

PROCESO	ENTRADA	HERRAMIENTA	SALIDA
Preparación para el lanzamiento	Incremento del producto potencialmente entregable. Historias de usuario completadas. Pruebas de calidad y aceptación.	Evaluación de calidad. Herramientas de gestión de versiones.	Producto listo para el lanzamiento. Lista de verificación del lanzamiento completada.
Lanzamiento	Producto listo para el lanzamiento. Lista de verificación del lanzamiento.	Herramientas de implementación y despliegue. Comunicación con Stakeholders.	Producto lanzado al entorno de producción. Notificación a los Stakeholders.
Revisión Post-Lanzamiento	Producto lanzado. Retroalimentación de los usuarios y Stakeholders.	Análisis de métricas. Reuniones de revisión post-lanzamiento.	Evaluación del lanzamiento y sus resultados. Identificación de oportunidades de

	mejora.

Fuente: Elaboración propia

B. Herramientas para el procesamiento de datos

A continuación, se presentan las herramientas que fueron empleadas:

Tabla 15

Herramientas para el procesamiento de datos

SOFTWARE	FABRICANTE	SERVICIO
WINDOWS 10 HOME	Microsoft Coporation	Edición Windows 10 Home del sistema operativo, ofrece variedad de características y funcionalidades para instalar las herramientas de desarrollo.
LENGUAJE JAVA	Oracle	Es un lenguaje de programación utilizado en el desarrollo de aplicaciones y software.
QUARKUS	RedHat	Es un framework de desarrollo diseñado específicamente para crear aplicaciones Java.
SERVIDOR JBOOS	RedHat	Es el servidor de aplicaciones que facilita la implementación y administración.
POSTMAN	Postman	Es una herramienta que permite probar y depurar APIs al facilitar el envío de solicitudes HTTP y la visualización de respuestas, simplificando así el desarrollo y prueba de aplicaciones que se comunican a través de APIs.
MARIADB	Oracle	MariaDB proporciona una base de datos de código abierto de alta calidad y rendimiento.
KEYCLOAK	RedHat	Es una plataforma de gestión de identidad y acceso que permite la administración y el control de la seguridad en aplicaciones y servicios.
JIRA	Atlassian	Es una de las herramientas más potentes para la gestión de proyectos con Scrum, la cual te ayudará a desarrollar rápidamente tus proyectos

	de software, como sitios y aplicaciones Web, Apps para dispositivos móviles o ya sea cualquier proyecto que utilice un marco de trabajo ágil. Además, el costo de uso y mantenimiento de esta herramienta es uno de los más bajos del mercado.

Fuente: Elaboración propia.

C. Técnicas de análisis de datos

La información recopilada mediante los instrumentos, como el análisis documental y la base de datos, son sometidos a análisis mediante el marco de trabajo Scrum. Este enfoque tiene como objetivo es elaborar los procedimientos en estudio que que brindarán datos relacionados con elementos como el algoritmo a utilizar, los protocolos de comunicación y las formas de autenticación y autorización.

3.7.3. Diseño estadístico

Dado el enfoque de investigación es descriptivo, no se introduce el diseño para realizar pruebas de hipótesis.

3.7.4. Análisis e interpretación de datos

La obtención de datos se realizó a través del análisis de documentos, abarcando dimensiones como Access Token, Refresh Token, Redirect_uri, y los métodos HTTP Get, Post, Put y Delete. En cuanto a los datos extraídos mediante la herramienta, se revisaron y reformularon las preguntas de los ítems 2, 4 y 13 debido a comentarios sobre su claridad, presentando una calificación intermedia; los demás ítems se mantienen en los valores previstos.

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

4.1. Análisis de los datos recolectados

Tabla 16

Análisis de los datos recolectados

Componente	Descripción	Requerimiento	Observación
Utilización del Access token como clave para acceder a los recursos protegidos.	Para la utilización como clave para acceder a los recursos protegidos se identifica que el acceso seguro es el 30%, autenticación eficiente es 15%, autorización eficiente es 30% y el tiempo de duración es limitada al 25%	Integrar la autenticación	Se requiere tener un entorno de desarrollo configurado con las herramientas necesarias para poder comenzar a trabajar en la implementación de la seguridad.
Obtención del acceso a recursos específicos mediante un access token válido.	Para la utilización el tipo de recurso del Access token se identifica que las APIs restringidas es 30%, servicios web es 25%, microservicios es 20% y los recursos confidenciales es 25%	Implementar la emisión del token de acceso	Se necesita implementar el mecanismo de emisión de tokens de acceso basados en el protocolo OpenID Connect.
Utilización del refresh token para renovar el access token una vez que haya expirado.	Para la utilización del refresh token en la renovación del Access token expirado se identifica que la renovación segura es 30%, actualización automática es	Implementar la renovación del token	Se necesita implementar la renovación del access token mediante resfresh token basados en el

	30%, extensión acceso es 25% y la continuidad autorización es 25%.		protocolo OpenID Connect.
Garantia del proceso de actualización del refresh token para la identidad y seguridad del usuario.	Para la utilización del proceso de actualización del token que garantiza la seguridad del usuario se identifica que refuerza confianza es 25%, mantiene protección es 30%, mejora control 25%, asegura acceso 25%.	Implementar la obtención del Access token	Se necesita implementar el refresh token como mecanismo de actualización del Access token basados en el protocolo OpenID Connect.
Especificación de los permisos y alcances a aplicaciones mediante redirect_uri durante la autenticación	Para la utilización de los alcances y permisos en las aplicaciones durante la autenticación se identifica que el control autenticación es 30%, permisos adaptados es 20%, acceso dirigido es 25% y alcances personalizados es 25%.	Implementar el manejo de errores de autenticación	Se necesita implementar un manejo adecuado de los errores de autenticación.
Asegurar la activación de los permisos solicitados en redirect_uri durante la autenticación	Para la utilización segura durante la autenticación de permisos solicitados estén activados se identifica que el control de redireccionamiento es 30%, permisos validados es 30%, flujo autorización es 20% y los accesos específicos es 20%.	Implementar el manejo de errores de autenticación	Se necesita implementar un manejo adecuado de los errores de autenticación, de manera que reciban los permisos solicitados sean claros en el proceso de autenticación.
Utilización de las solicitudes GET para recuperar información o recursos específicos.	Para la utilización del GET en la recuperación de recursos se identifica que la recuperación datos es 35%, obtención recursos es 20%, información específica es 25% y datos solicitados es 20%.	Integrar OpenID Connect en la API	Se necesita integrar el OpenID Connect en la API de Swagger
Seguimiento de	Para la utilización en la	Integrar OpenID	Se necesita integrar

convenciones o estándares específicos al crear recursos de tipo GET para asegurar su coherencia y claridad	creación de recursos GET sigue estándares específicos se identifica el uso de convenciones (practicas o patrones) es 25%, estándares y normas seguidos es 30%, prácticas recomendadas 20%, conformidad normas y directrices es 25%.	Connect en la API de Swagger	el OpenID Connect en la API de Swagger
Envío en el cuerpo de la petición en las solicitudes POST para crear recurso	Para la utilización en las solicitudes POST envían en el cuerpo de la petición se identifica que envío cuerpo es 20%, datos incluidos es 25%, cuerpo petición es 25% y contenido POST 30%.	Integrar OpenID Connect en la API de Swagger	Se necesita integrar el OpenID Connect en la API de Swagger
Obtención de las respuestas de las solicitudes POST contienen datos precisos y relevantes	Para la utilización de datos relevantes de respuestas de las solicitudes POST se identifica que los datos específicos es 30 %, contenido relevante es 25%, respuestas detalladas son 25%, información puntual es 20%.	Integrar OpenID Connect en la API de Swagger	Se necesita integrar el OpenID Connect en la API de Swagger
Actualización de datos de recursos existentes de manera segura y eficiente mediante las solicitudes PUT	Para la utilización de datos actualizados con las solicitudes PUT se identifica que la actualización recursos son 30%, modificación datos es 20%, cambios existentes es 25%, Datos actualizados es 25%.	Integrar OpenID Connect en la API	Se necesita integrar el OpenID Connect en la API de Swagger
Seguimiento del proceso seguro en la actualización de datos en las	Para la utilización de un proceso seguro que siguen las solicitudes PUT se identifica que proceso seguro es 30%, seguridad	Integrar OpenID Connect en la API	Se necesita integrar el OpenID Connect en la API de Swagger

solicitudes PUT	aplicada es 20%, garantía protección es 25%, cumple estándares es 25%.		
Utilización de las solicitudes DELETE para para eliminar recursos de manera segura	Para la utilización de solicitud DELETE para eliminar recurso de manera segura se identifica que la eliminación segura es 25%, borrado confiable es 25%, proceso protegido es 25% y datos removidos es 25%.	Integrar OpenID Connect en la API	Se necesita integrar el OpenID Connect en la API de Swagger
Implementación de algún tipo de confirmación o medida de seguridad antes de proceder con la eliminación para evitar acciones no deseadas	Para la utilización del tipo de mediada de seguridad antes del proceso de eliminación se identifica que confirmación previa es 30%, medidas seguridad es 20%, protección eliminación es 25%, proceso seguro es 25%.	Integrar OpenID Connect en la API	Se necesita integrar el OpenID Connect en la API de Swagger

Fuente: Elaboración propia

4.2 RESULTADOS

4.2.1. Fase de análisis

Según las cuatro etapas descritas en la tabla 10 del tercer capítulo, en la fase I de análisis de la metodología scrum, se obtienen las historias de usuario, la arquitectura inicial y el plan de alto nivel, los cuales se exponen en las siguientes tablas.

Tabla 17

Historias de usuario

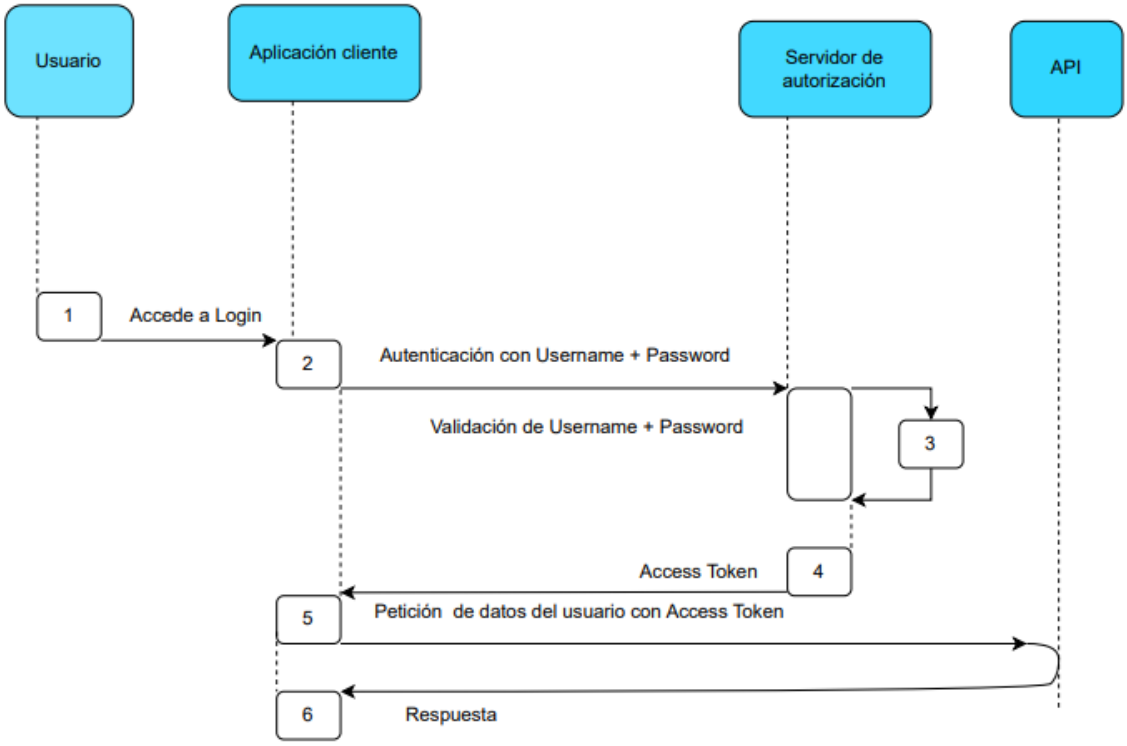
N°	Historia de usuario	Descripción
1	Configuración inicial del entorno de desarrollo	Como desarrollador, quiero tener un entorno de desarrollo configurado con las herramientas necesarias para poder comenzar a trabajar en la implementación de la seguridad.

2	Integrar la autenticación OpenID Connect en la API de Swagger	Como desarrollador, deseo integrar el flujo de autenticación OpenID Connect en la API de Swagger, de manera que los usuarios puedan autenticarse correctamente al acceder a los endpoints protegidos.
3	Implementar la emisión de tokens de acceso	Como desarrollador, necesito implementar el mecanismo de emisión de tokens de acceso basados en el protocolo OpenID Connect, para que los usuarios autenticados puedan acceder a los recursos protegidos.
4	Configurar el proveedor de identidad OpenID Connect	Como administrador del sistema, quiero poder configurar el proveedor de identidad OpenID Connect de acuerdo con los requisitos específicos de la aplicación, para garantizar la seguridad adecuada en la autenticación y autorización de usuarios.
5	Definir roles y permisos basados en tokens	Como desarrollador, deseo definir roles y permisos basados en los tokens de acceso emitidos por el proveedor de identidad, para que los usuarios tengan acceso solo a los recursos para los cuales tienen autorización.
6	Implementar el manejo de errores de autenticación	Como desarrollador, necesito implementar un manejo adecuado de los errores de autenticación, de manera que los usuarios reciban mensajes claros y útiles en caso de fallos en el proceso de autenticación.
7	Establecer la renovación de tokens de acceso	Como desarrollador, quiero implementar la renovación automática de tokens de acceso antes de que expiren, para que los usuarios no sean interrumpidos mientras usan la aplicación.
8	Auditar eventos de seguridad	Como administrador del sistema, necesito establecer la capacidad de auditar y registrar eventos de seguridad relacionados con la autenticación y autorización de usuarios a través del protocolo OpenID Connect.
9	Implementar pruebas de integración	Como desarrollador, deseo implementar pruebas de integración exhaustivas para asegurar que la implementación del Protocolo OpenID Connect en las APIs de Swagger funcione correctamente en diferentes escenarios.

Fuente: Elaboración propia

Figura 5

Arquitectura técnica inicial



Fuente: Elaboración propia

Tabla 18

Historia de usuario. Configuración inicial del entorno de desarrollo

Historia de Usuario	
ID	POIDC-01
Nombre	Configuración inicial del entorno de desarrollo
Prioridad	Alta
Riesgo	Bajo
Descripción	Configurar entorno de desarrollo con las herramientas necesarias para poder comenzar a trabajar en la implementación de la seguridad
Validación	- Las herramientas necesarias están instaladas correctamente. - Se puede compilar y ejecutar el proyecto de ejemplo.

Fuente: Elaboración propia

Tabla 19

Historia de usuario. Integrar la autenticación OpenID Connect en la API de Swagger

Historia de Usuario	
ID	POIDC-02
Nombre	Integrar la autenticación OpenID Connect en la API de Swagger
Prioridad	Alta
Riesgo	Medio
Descripción	Integrar el flujo de autenticación OpenID Connect en la API de Swagger, de manera que los usuarios puedan autenticarse correctamente al acceder a los endpoints protegidos.
Validación	- Los usuarios pueden autenticarse utilizando OpenID Connect. - Los endpoints protegidos restringen el acceso a usuarios no autenticados.

Fuente: Elaboración propia

Tabla 20

Historia de usuario, Implementar la emisión de tokens de acceso

Historia de Usuario	
ID	POIDC-03
Nombre	Implementar la emisión de tokens de acceso
Prioridad	Alta
Riesgo	Alta
Descripción	Implementar el mecanismo de emisión de tokens de acceso basados en el protocolo OpenID Connect, para que los usuarios autenticados puedan acceder a los recursos protegidos.
Validación	- Los tokens de acceso son generados correctamente para usuarios autenticados. - Los tokens son válidos por un período definido.

Fuente: Elaboración propia

Tabla 21

Historia de usuario. Configurar el proveedor de identidad OpenID Connect

Historia de Usuario	
ID	POIDC-04
Nombre	Configurar el proveedor de identidad OpenID Connect
Prioridad	Alta
Riesgo	Medio
Descripción	Configurar el proveedor de identidad OpenID Connect de acuerdo con los requisitos específicos de la aplicación, para garantizar la seguridad adecuada en la autenticación y autorización de usuarios.
Validación	- El proveedor de identidad está configurado con los parámetros requeridos. - Las políticas de autenticación y autorización son aplicadas correctamente

Fuente: Elaboración propia

Tabla 22

Historia de usuario. Definir roles y permisos basados en tokens

Historia de Usuario	
ID	POIDC-05
Nombre	Definir roles y permisos basados en tokens
Prioridad	Media
Riesgo	Medio
Descripción	Definir roles y permisos basados en los tokens de acceso emitidos por el proveedor de identidad, para que los usuarios tengan acceso solo a los recursos para los cuales tienen autorización.
Validación	- Los roles y permisos son definidos en el token de acceso. - Los recursos restringidos son accesibles solo para usuarios autorizados.

Fuente: Elaboración propia

Tabla 23

Historia de usuario. Implementar el manejo de errores de autenticación

Historia de Usuario	
ID	POIDC-06

Nombre	Implementar el manejo de errores de autenticación
Prioridad	Media
Riesgo	Bajo
Descripción	Implementar un manejo adecuado de los errores de autenticación, de manera que los usuarios reciban mensajes claros y útiles en caso de fallos en el proceso de autenticación.
Validación	- Los errores de autenticación generan mensajes de error claros. - Los usuarios son redirigidos a páginas de error personalizadas.

Fuente: Elaboración propia

Tabla 24

Historia de usuario. Establecer la renovación de tokens de acceso

Historia de Usuario	
ID	POIDC-07
Nombre	Establecer la renovación de tokens de acceso
Prioridad	Alta
Riesgo	Medio
Descripción	Implementar la renovación automática de tokens de acceso antes de que expiren, para que los usuarios no sean interrumpidos mientras usan la aplicación.
Validación	- Los tokens de acceso pueden ser renovados antes de la expiración. - Los usuarios no necesitan autenticarse nuevamente durante la sesión.

Fuente: Elaboración propia

Tabla 25

Historia de usuario. Auditar eventos de seguridad

Historia de Usuario	
ID	POIDC-08
Nombre	Auditar eventos de seguridad
Prioridad	Media
Riesgo	Alto
Descripción	Establecer la capacidad de auditar y registrar eventos de seguridad relacionados con la autenticación y autorización de usuarios a través

	del protocolo OpenID Connect.
Validación	- Los eventos de autenticación y autorización son registrados en un sistema de auditoría. - Los registros son accesibles para fines de revisión.

Fuente: Elaboración propia

Tabla 26

Historia de usuario. Implementar pruebas de integración

Historia de Usuario	
ID	POIDC-09
Nombre	Implementar pruebas de integración
Prioridad	Alto
Riesgo	Medio
Descripción	Implementar pruebas de integración exhaustivas para asegurar que la implementación del Protocolo OpenID Connect en las APIs de Swagger funcione correctamente en diferentes escenarios.
Validación	- Las pruebas de integración cubren los flujos de autenticación y autorización. - Los resultados de las pruebas son consistentes y predecibles.

Fuente: Elaboración propia

Tareas de las historias de usuario

Se detallan a continuación las tareas que componen las historias de usuario.

Tabla 27

Tarea. Instalación de herramientas

Tarea	POIDC-T01
Historia de Usuario	POIDC-01
Estado	Completada
Descripción	Instalar las herramientas de desarrollo necesarias, como el entorno de desarrollo integrado (IDE) y las utilidades de línea de comandos requeridas para el proyecto.

Fuente: Elaboración propia

Tabla 28

Tarea. Configuración de dependencias

Tarea	POIDC-T02
Historia de Usuario	POIDC-01
Estado	Completada
Descripción	Configurar y gestionar las dependencias y bibliotecas necesarias para la implementación del Protocolo OpenID Connect, utilizando un sistema de gestión de paquetes.

Fuente: Elaboración propia

Tabla 29

Tarea. Configuración de variables de entorno

Tarea	POIDC-T03
Historia de Usuario	POIDC-01
Estado	Completada
Descripción	Definir y configurar las variables de entorno necesarias para la implementación del protocolo y las credenciales de autenticación con el proveedor de identidad OpenID Connect.

Fuente: Elaboración propia

Tabla 30

Tarea. Configuración del servidor local

Tarea	POIDC-T04
Historia de Usuario	POIDC-01
Estado	Completada
Descripción	Asegurar que el servidor de desarrollo local esté correctamente configurado para admitir el flujo de autenticación y las redirecciones del Protocolo OpenID Connect.

Fuente: Elaboración propia

Tabla 31

Tarea. Creación de instancia de prueba

Tarea	POIDC-T05

Historia de Usuario	POIDC-01
Estado	Completada
Descripción	Configurar una instancia de prueba de la API expuesta por Swagger para que esté disponible en el entorno de desarrollo y permita la interacción inicial.

Fuente: Elaboración propia

Tabla 32

Tarea. Integración de bibliotecas OpenID Connect

Tarea	POIDC-T06
Historia de Usuario	POIDC-02
Estado	Completada
Descripción	Incorporar las bibliotecas y componentes necesarios en el proyecto para permitir la integración del flujo de autenticación OpenID Connect en la API de Swagger.

Fuente: Elaboración propia

Tabla 33

Tarea. Identificación de endpoints protegidos

Tarea	POIDC-T07
Historia de Usuario	POIDC-02
Estado	Completada
Descripción	Analizar e identificar los endpoints específicos de la API de Swagger que deben ser protegidos por autenticación OpenID Connect.

Fuente: Elaboración propia

Tabla 34

Tarea. Configuración de políticas de seguridad

Tarea	POIDC-T08
Historia de Usuario	POIDC-02
Estado	Completada

Descripción	Configurar las políticas de seguridad para los endpoints protegidos, definiendo los requisitos de autenticación y autorización necesarios.

Fuente: Elaboración propia

Tabla 35

Tarea. Implementación de flujo de autenticación

Tarea	POIDC-T09
Historia de Usuario	POIDC-02
Estado	Completada
Descripción	Desarrollar el flujo de autenticación basado en el Protocolo OpenID Connect, asegurando que los usuarios sean redirigidos al proveedor de identidad y puedan regresar a la API con tokens válidos.

Fuente: Elaboración propia

Tabla 36

Tarea. Validación de tokens de acceso

Tarea	POIDC-T10
Historia de Usuario	POIDC-02
Estado	Completada
Descripción	Implementar la validación de los tokens de acceso recibidos de proveedor de identidad para garantizar su autenticidad y vigencia antes de permitir el acceso a los recursos protegidos.

Fuente: Elaboración propia

Tabla 37

Tarea. Pruebas de flujo de autenticación

Tarea	POIDC-T11
Historia de Usuario	POIDC-02
Estado	Completada
Descripción	Realizar pruebas exhaustivas de los flujos de autenticación en diferentes situaciones, como autenticación exitosa y fallos de autenticación, para verificar la robustez y el comportamiento

	esperado.

Fuente: Elaboración propia

Tabla 38

Tarea. Gestión de excepciones de autenticación

Tarea	POIDC-T12
Historia de Usuario	POIDC-02
Estado	Completada
Descripción	Implementar un manejo adecuado de las excepciones que puedan ocurrir durante el flujo de autenticación, como fallos de comunicación con el proveedor de identidad o errores en la respuesta del proveedor.

Fuente: Elaboración propia

Tabla 39

Tarea. Integración de mensajes de error

Tarea	POIDC-T13
Historia de Usuario	POIDC-02
Estado	Completada
Descripción	Incorporar mensajes de error descriptivos y amigables para los usuarios en caso de fallos de autenticación, brindando orientación sobre los pasos a seguir para solucionar los problemas.

Fuente: Elaboración propia

Tabla 40

Tarea. Diseño de estructura de tokens

Tarea	POIDC-T14
Historia de Usuario	POIDC-03
Estado	Completada
Descripción	Definir la estructura de los tokens de acceso a ser emitidos por el proveedor de identidad, incluyendo los campos necesarios para la autenticación y autorización.

Fuente: Elaboración propia

Tabla 41

Tarea. Configuración de emisión de tokens

Tarea	POIDC-T15
Historia de Usuario	POIDC-03
Estado	Completada
Descripción	Configurar el proveedor de identidad para generar y emitir tokens de acceso válidos y seguros, aplicando las directrices del Protocolo OpenID Connect.

Fuente: Elaboración propia

Tabla 42

Tarea. Integración de emisión de tokens

Tarea	POIDC-T16
Historia de Usuario	POIDC-03
Estado	Completada
Descripción	Integrar la lógica de emisión de tokens de acceso en el flujo de autenticación, de manera que los tokens sean generados y entregados a los usuarios autenticados.

Fuente: Elaboración propia

Tabla 43

Tarea. Establecimiento de tiempo de vida de tokens

Tarea	POIDC-T17
Historia de Usuario	POIDC-03
Estado	Completada
Descripción	Definir el período de tiempo durante el cual los tokens de acceso serán válidos antes de expirar, ajustando este valor de acuerdo a los requisitos de seguridad y usabilidad.

Fuente: Elaboración propia

Tabla 44

Tarea. Validación de tokens recibidos

Tarea	POIDC-T18
Historia de Usuario	POIDC-03
Estado	Completada
Descripción	Implementar la validación de los tokens de acceso recibidos por la API de Swagger, asegurándose de que los tokens sean válidos, no estén manipulados y no hayan expirado.

Fuente: Elaboración propia

Tabla 45

Tarea. Actualización de tokens

Tarea	POIDC-T19
Historia de Usuario	POIDC-03
Estado	Completada
Descripción	Desarrollar la funcionalidad para permitir a los usuarios renovar sus tokens de acceso antes de que expiren, manteniendo una experiencia de sesión continua.

Fuente: Elaboración propia

Tabla 46

Tarea. Pruebas de emisión y validación de tokens

Tarea	POIDC-T20
Historia de Usuario	POIDC-03
Estado	Completada
Descripción	Realizar pruebas exhaustivas para verificar la correcta emisión, validación y renovación de los tokens de acceso, asegurando su funcionalidad y seguridad en distintos escenarios.

Fuente: Elaboración propia

Tabla 47

Tarea. Investigación de requisitos

Tarea	POIDC-T21
Historia de Usuario	POIDC-04
Estado	Completada
Descripción	Analizar los requisitos específicos de la aplicación y el entorno para determinar la configuración necesaria del proveedor de identidad OpenID Connect.

Fuente: Elaboración propia

Tabla 48

Tarea. Configuración del proveedor de identidad

Tarea	POIDC-T22
Historia de Usuario	POIDC-04
Estado	Completada
Descripción	Configurar el proveedor de identidad OpenID Connect de acuerdo con los resultados de la investigación, estableciendo las políticas y parámetros de seguridad requeridos.

Fuente: Elaboración propia

Tabla 49

Tarea. Integración de la configuración en la API

Tarea	POIDC-T23
Historia de Usuario	POIDC-04
Estado	Completada
Descripción	Incorporar la configuración del proveedor de identidad OpenID Connect en la configuración de la API de Swagger, asegurando que la autenticación y autorización se realicen según lo definido.

Fuente: Elaboración propia

Tabla 50

Tarea. Configuración de alcances y permisos

Tarea	POIDC-T24
Historia de Usuario	POIDC-04
Estado	Completada
Descripción	Definir los alcances y permisos necesarios para el flujo de autenticación, permitiendo que los usuarios accedan a recursos específicos según sus roles y autorizaciones.

Fuente: Elaboración propia

Tabla 51

Tarea. Pruebas de configuración del proveedor

Tarea	POIDC-T25
Historia de Usuario	POIDC-04
Estado	Completada
Descripción	Realizar pruebas para verificar que la configuración del proveedor de identidad se ajusta a los requisitos, asegurando que los flujos de autenticación y autorización sean correctos.

Fuente: Elaboración propia

Tabla 52

Tarea. Ajustes de seguridad y tokens

Tarea	POIDC-T26
Historia de Usuario	POIDC-04
Estado	Completada
Descripción	Asegurarse de que la configuración del proveedor de identidad y la emisión de tokens se alineen con los estándares de seguridad recomendados por OpenID Connect y OAuth 2.0.

Fuente: Elaboración propia

Tabla 53

Tarea. Pruebas de extremo a extremo

Tarea	POIDC-T27
Historia de Usuario	POIDC-04
Estado	Completada
Descripción	Realizar pruebas de extremo a extremo para verificar que la configuración del proveedor de identidad y la integración en la API de Swagger funcionan de manera coherente y segura.

Fuente: Elaboración propia

Tabla 54

Tarea. Definición de roles y permisos

Tarea	POIDC-T28
Historia de Usuario	POIDC-05
Estado	Completada
Descripción	Identificar los roles y permisos necesarios para el sistema, considerando los recursos y acciones que cada rol puede acceder y realizar.

Fuente: Elaboración propia

Tabla 55

Tarea. Integración de roles y permisos en tokens

Tarea	POIDC-T29
Historia de Usuario	POIDC-05
Estado	Completada
Descripción	Incorporar la información de roles y permisos en los tokens de acceso generados por el proveedor de identidad, de manera que la autorización se base en esta información.

Fuente: Elaboración propia

Tabla 56

Tarea. Configuración de políticas de autorización

Tarea	POIDC-T30
Historia de Usuario	POIDC-05
Estado	Completada
Descripción	Establecer políticas de autorización en la API de Swagger que utilicen los roles y permisos presentes en los tokens para permitir o restringir el acceso a recursos específicos.

Fuente: Elaboración propia

Tabla 57

Tarea. Pruebas de autorización

Tarea	POIDC-T31
Historia de Usuario	POIDC-05
Estado	Completada
Descripción	Realizar pruebas para verificar que los usuarios con diferentes roles tengan acceso a los recursos adecuados y sean rechazados en los recursos restringidos.

Fuente: Elaboración propia

Tabla 58

Tarea. Validación de autorización en endpoints

Tarea	POIDC-T32
Historia de Usuario	POIDC-05
Estado	Completada
Descripción	Verificar que los endpoints protegidos por autorización OpenID Connect funcionen correctamente, permitiendo o denegando el acceso según los roles y permisos de los usuarios.

Fuente: Elaboración propia

Tabla 59

Tarea. Identificación de escenarios de error

Tarea	POIDC-T33
Historia de Usuario	POIDC-06
Estado	Completada
Descripción	Identificar los posibles escenarios en los que pueden ocurrir errores durante el proceso de autenticación y autorización.

Fuente: Elaboración propia

Tabla 60

Tarea. Implementación de páginas de error personalizadas

Tarea	POIDC-T34
Historia de Usuario	POIDC-06
Estado	Completada
Descripción	Desarrollar páginas de error personalizadas que muestren los mensajes de error de manera coherente y que brinden una experiencia de usuario amigable.

Fuente: Elaboración propia

Tabla 61

Tarea. Pruebas de manejo de errores

Tarea	POIDC-T35
Historia de Usuario	POIDC-06
Estado	Completada
Descripción	Realizar pruebas exhaustivas para verificar que los mensajes de error se muestren correctamente y que los usuarios sean redirigidos a las páginas de error correspondientes.

Fuente: Elaboración propia

Tabla 62

Tarea. Implementación de lógica de renovación

Tarea	POIDC-T36
Historia de Usuario	POIDC-07
Estado	Completada
Descripción	Desarrollar la lógica que permitirá a los usuarios renovar automáticamente sus tokens de acceso antes de que expiren.

Fuente: Elaboración propia

Tabla 63

Tarea. Integración de renovación en el flujo

Tarea	POIDC-T37
Historia de Usuario	POIDC-07
Estado	Completada
Descripción	Incorporar la opción de renovación automática de tokens en los flujos de autenticación, para que los usuarios tengan una experiencia de sesión continua.

Fuente: Elaboración propia

Tabla 64

Tarea. Pruebas de renovación automática

Tarea	POIDC-T38
Historia de Usuario	POIDC-07
Estado	Completada
Descripción	Realizar pruebas exhaustivas para verificar que los tokens se renuevan automáticamente antes de expirar, garantizando que los usuarios no sean interrumpidos en su actividad.

Fuente: Elaboración propia

Tabla 65

Tarea. Integración de registro de eventos

Tarea	POIDC-T39
Historia de Usuario	POIDC-08
Estado	Completada
Descripción	Incorporar la lógica de registro de eventos en los flujos de autenticación y autorización, para registrar los eventos relevantes en el sistema.

Fuente: Elaboración propia

Tabla 66

Tarea. Configuración de almacenamiento de registros

Tarea	POIDC-T40
Historia de Usuario	POIDC-08
Estado	Completada
Descripción	Configurar un sistema de almacenamiento adecuado para los registros de auditoría, asegurando que los eventos se guarden de manera segura y persistente.

Fuente: Elaboración propia

Tabla 67

Tarea. Pruebas de registro de eventos

Tarea	POIDC-T41
Historia de Usuario	POIDC-08
Estado	Completada
Descripción	Realizar pruebas para verificar que los eventos de seguridad se registran correctamente y que los registros pueden ser accedidos para su revisión.

Fuente: Elaboración propia

Tabla 68

Tarea. Identificación de casos de prueba

Tarea	POIDC-T42
Historia de Usuario	POIDC-09
Estado	Completada
Descripción	Identificar los flujos de autenticación y autorización que deben ser cubiertos por las pruebas de integración, considerando diferentes escenarios.

Fuente: Elaboración propia

Tabla 69

Tarea. Desarrollo de casos de prueba

Tarea	POIDC-T43
Historia de Usuario	POIDC-09
Estado	Completada
Descripción	Crear casos de prueba que cubran los flujos de autenticación y autorización, incluyendo combinaciones de roles y permisos, diferentes resultados de autenticación y posibles errores.

Fuente: Elaboración propia

Tabla 70

Configuración del entorno de prueba

Tarea	POIDC-T44
Historia de Usuario	POIDC-09
Estado	Completada
Descripción	Preparar un entorno de prueba separado donde se puedan ejecutar las pruebas de integración sin afectar al entorno de producción.

Fuente: Elaboración propia

Tabla 71

Tarea. Ejecución de pruebas de integración

Tarea	POIDC-T45

Historia de Usuario	POIDC-09
Estado	Completada
Descripción	Ejecutar los casos de prueba de integración para verificar que los flujos de autenticación y autorización funcionan correctamente en diferentes situaciones.

Fuente: Elaboración propia

Tabla 72

Tarea. Registro y seguimiento de problemas

Tarea	POIDC-T46
Historia de Usuario	POIDC-09
Estado	Completada
Descripción	Identificar y registrar cualquier problema o anomalía encontrada durante las pruebas de integración, para que puedan ser corregidos antes de la implementación final.

Fuente: Elaboración propia

4.2.2. Fase de planificación

En las etapas de scrum descritos en la tabla 11 del tercer capítulo, en la fase II de planificación de la metodología Scrum, se establecen las responsabilidades de los participantes, después se lleva a cabo la estimación y priorización de todas las historias de usuario.

En el proyecto actual, se emplea la metodología de estimación de historias de usuario que se fundamenta en la consideración del tiempo y la experiencia del usuario. Se utilizan los números 0, 1, 2, 3, 5, 8, etc, como medidas de esfuerzo. Cuando se asigna el valor 0 a una historia, se está indicando que el esfuerzo requerido es mínimo o de baja complejidad. En contraste, un valor de 21 sugiere un esfuerzo extremadamente alto, incluso llegando al punto de ser poco realista en la situación actual.

Tabla 73

Historias de usuario priorizados

Historias de	Miembro	Estimación	Prioridad

Usuario	A	B	C	media (hrs)	
POIDC-01	12			12	Alta
POIDC-02	18			18	Alta
POIDC-03	15			15	Alta
POIDC-04	10			10	Alta
POIDC-05	8			8	Media
POIDC-06	6			6	Media
POIDC-07	10			10	Alta
POIDC-08	12			12	Media
POIDC-09	20			20	Alta

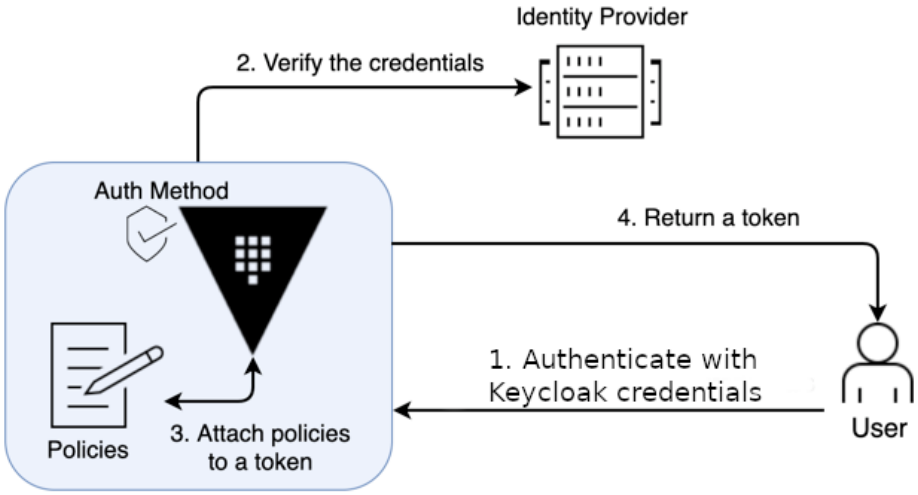
Fuente: Elaboración Propia

4.1.1. Fase de implementación

En las etapas de scrum descritos en la tabla 12 del tercer capítulo, la fase III de implementación de la metodología Scrum, facilita la generación del diagrama de componentes, la ejecución de las tareas de usuario, la creación del código fuente para las clases de entidad y la entrega de elementos funcionales llamados incrementos, los cuales corresponden a la primera iteración. Luego, se llevará a cabo una evaluación en la reunión del equipo ágil posteriormente serán validados por el product owner.

Figura 6

Diagrama de componentes

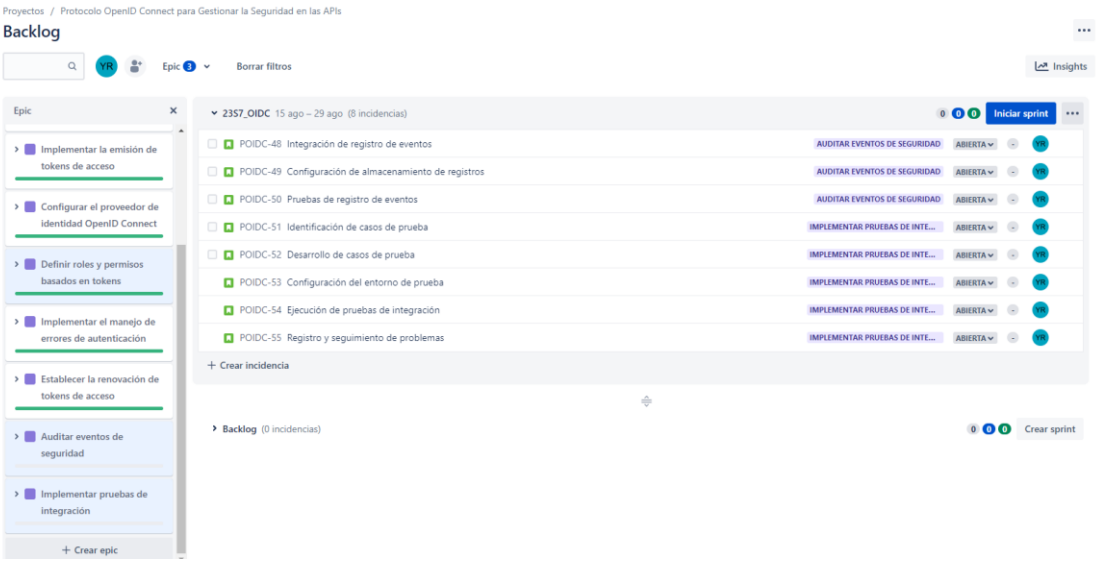


Nota. Diagrama que muestra los componentes de openId connect y swagger

Fuente: Elaboración Propia

Figura 7

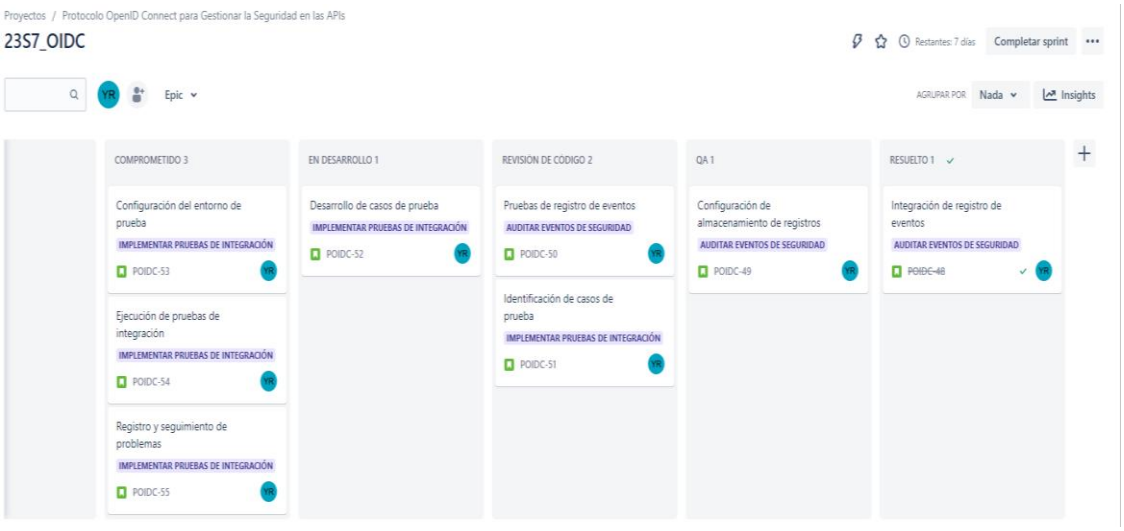
Product backlog



Fuente: Elaboración Propia

Figura 8

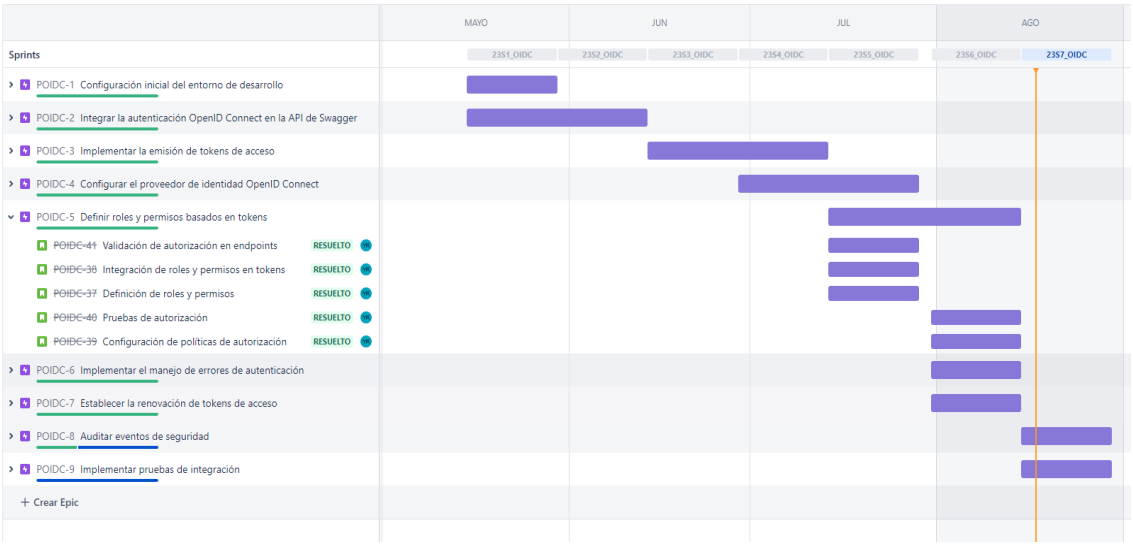
Tablero sprint



Fuente: Elaboración Propia

Figura 9

Cronograma (plan de trabajo)



Fuente: Elaboración Propia

Tabla 74

Código fuente. Configuración de conexión a base de datos

```
# Configuración para la conexion a la base de datos
quarkus.datasource.db-kind=mariadb
quarkus.datasource.username=root
quarkus.datasource.password=password
quarkus.datasource.jdbc.url=jdbc:mariadb://localhost:3306/pacientes
quarkus.hibernate-orm.database.generation=update
quarkus.hibernate-orm.unsupported-
properties."hibernate.physical_naming_strategy"=org.hibernate.boot.mod
el.naming.CamelCaseToUnderscoresNamingStrategy
```

Fuente: Elaboración Propia

Tabla 75

Código fuente.Configuración de OpenId Connect

```
# Configuración básica de OpenID Connect
quarkus.oidc.auth-server-url=http://127.0.0.1:2121/realms/poidc-dev
quarkus.oidc.client-id=backend-quarkus
quarkus.oidc.credentials.secret=ziInKO4glhtjqyj0A3fL4i3CcQm4x64w
quarkus.oidc.tls.verification=none
quarkus.keycloak.policy-enforcer.enable=true
```

Fuente: Elaboración Propia

Tabla 76

Código fuente. Configuración de swagger

```
#Tipos de Seguridad: basic, jwt, oauth2, oidc, oauth2-implicit
quarkus.smallrye-openapi.security-scheme=oidc
quarkus.smallrye-openapi.security-scheme-name=backend-quarkus
quarkus.smallrye-openapi.oidc-open-id-connect-
url=http://127.0.0.1:2121/realms/poidc-dev/.well-known/openid-
configuration
quarkus.swagger-ui.oauth2-redirect-
url=http://localhost:8080/q/swagger-ui

quarkus.keycloak.policy-enforcer.paths.public2.path=/q/swagger-ui/*
quarkus.keycloak.policy-enforcer.paths.public2.enforcement-
mode=disabled

quarkus.keycloak.policy-enforcer.paths.public3.path=/q/openapi
quarkus.keycloak.policy-enforcer.paths.public3.enforcement-
mode=disabled
```

Fuente: Elaboración Propia

Tabla 77

Código fuente. Clase principal ApiAplication

```
package pe.unsch.openapi.keycloak;

import jakarta.ws.rs.core.Application;
import org.eclipse.microprofile.openapi.annotations.OpenAPIDefinition;
import org.eclipse.microprofile.openapi.annotations.info.Contact;
import org.eclipse.microprofile.openapi.annotations.info.Info;
import org.eclipse.microprofile.openapi.annotations.info.License;
import org.eclipse.microprofile.openapi.annotations.tags.Tag;

@OpenAPIDefinition(
    tags = {
        @Tag(name = "Paciente", description = "Datos de la persona que recibe la vacuna."),
        @Tag(name = "Vacuna", description = "Vacunas del paciente")
    },
    info = @Info(
        title = "OpenID Connect para la Seguridad en las APIs",
        version = "1.0.0",
        contact = @Contact(
            name = "Dirección Regional de Salud de Ayacucho",
            url = "https://www.gob.pe/regionayacucho-diresa",
            email = "noreply@diresa.gob.pe"),
        license = @License(
```

```
                name = "Apache 2.0",
                url = "http://www.apache.org/licenses/LICENSE-2.0.html"))
)
public class ApiApplication extends Application {
}
```

Fuente: Elaboración Propia

Tabla 78

Código fuente. Modelos usando PanacheEntity

```
package pe.unsch.openapi.keycloak.models;

import io.quarkus.hibernate.orm.panache.PanacheEntity;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Transient;
import lombok.Getter;
import lombok.Setter;

import java.util.List;

@Getter
@Setter
@Entity
public class Paciente extends PanacheEntity {
    private String nombre;
    private String apellidos;
    @Column(name = "documento_identidad")
    private String documentoIdentidad;
    @Column(name = "historia_clinica")
    private String historiaClinica;
    @Transient
    private List<Vacuna> vacunas;
}

package pe.unsch.openapi.keycloak.models;

import io.quarkus.hibernate.orm.panache.PanacheEntity;
import jakarta.persistence.Entity;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@Entity
public class Vacuna extends PanacheEntity {
    private String nombre;
    private String fabricante;
    private String dosis;
    private String lugar;
}
```

Fuente: Elaboración Propia

Tabla 79

Código fuente, Clase Resource Vacuna

```
package pe.unsch.openapi.keycloak.controllers;

import io.quarkus.security.Authenticated;
import jakarta.annotation.security.PermitAll;
import jakarta.transaction.Transactional;
import jakarta.ws.rs.GET;
import jakarta.ws.rs.POST;
import jakarta.ws.rs.Path;
import jakarta.ws.rs.PathParam;
import org.eclipse.microprofile.openapi.annotations.tags.Tag;
import pe.unsch.openapi.keycloak.models.Vacuna;

import java.util.List;

@Tag(name = "Vacuna")
@Path("/vacuna")
@Authenticated
public class VacunaResource {

    @GET
    @Path("/publica/{pacienteId}/list")
    @PermitAll
    public List<Vacuna> getVacunasPublicas(@PathParam("pacienteId")
Long pacienteId) {
        return Vacuna.find("pacienteId=?1", pacienteId).list();
    }

    @GET
    @Path("/{pacienteId}/list")
//    @RolesAllowed("owner")
    public List<Vacuna> getVacunas(@PathParam("pacienteId") Long
pacienteId) {
        return Vacuna.find("pacienteId=?1", pacienteId).list();
    }

    @GET
    @Path("/{vacunaId}")
//    @RolesAllowed("manager")
    public Vacuna getVacuna(@PathParam("vacunaId") Long vacunaId) {
        return Vacuna.findById(vacunaId);
    }

    @POST
    @Transactional
//    @RolesAllowed("manager")
    public Vacuna createVacuna(Vacuna vacuna) {
        vacuna.persist();
        return vacuna;
    }
}
```

Fuente: Elaboración Propia

Tabla 80

Código fuente, Clase Resource Paciente

```
package pe.unsch.openapi.keycloak.controllers;

import io.quarkus.security.Authenticated;
import jakarta.annotation.security.PermitAll;
import jakarta.annotation.security.RolesAllowed;
import jakarta.transaction.Transactional;
import jakarta.ws.rs.GET;
import jakarta.ws.rs.POST;
import jakarta.ws.rs.Path;
import jakarta.ws.rs.PathParam;
import org.eclipse.microprofile.openapi.annotations.tags.Tag;
import pe.unsch.openapi.keycloak.models.Paciente;
import pe.unsch.openapi.keycloak.models.Vacuna;

import java.util.List;

@Tag(name = "Paciente")
@Path("/paciente")
@Authenticated
public class PacienteResource {
    @GET
    @Path("/list")
//    @RolesAllowed("admin")
    public List<Vacuna> getPacientes() {
        return Paciente.listAll();
    }

    @GET
    @Path("/vacunas/{pacienteId}")
//    @RolesAllowed("admin")
    public Paciente getVacunas(@PathParam("pacienteId") Long
pacienteId) {
        Paciente paciente = Paciente.findById(pacienteId);
       // paciente.setVacunas(Vacuna.find("pacienteId=?1",
pacienteId).list());
        return paciente;
    }

    @POST
    @Transactional
//    @RolesAllowed("admin")
    public Paciente createPaciente(Paciente paciente) {
        paciente.persist();
        paciente.getVacunas().forEach(vacuna -> {
            vacuna.persist();
        });
        return paciente;
    }
}
```

Fuente: Elaboración Propia

Tabla 81

Código fuente, Docker para ejecutar de forma nativa

```
FROM registry.access.redhat.com/ubi8/ubi-minimal:8.6
WORKDIR /work/
RUN chown 1001 /work \
    && chmod "g+rwX" /work \
    && chown 1001:root /work
COPY --chown=1001:root target/*-runner /work/application

EXPOSE 8080
USER 1001

CMD ["./application", "-Dquarkus.http.host=0.0.0.0"]
```

Fuente: Elaboración Propia

Tabla 82

Código fuente. Docker para ejecutar dentro de jvm

```
FROM registry.access.redhat.com/ubi8/openjdk-17:1.14

ENV LANGUAGE='en_US:en'


# We make four distinct layers so if there are application changes the
library layers can be re-used
COPY --chown=185 target/quarkus-app/lib/ /deployments/lib/
COPY --chown=185 target/quarkus-app/*.jar /deployments/
COPY --chown=185 target/quarkus-app/app/ /deployments/app/
COPY --chown=185 target/quarkus-app/quarkus/ /deployments/quarkus/

EXPOSE 8080
USER 185
ENV JAVA_OPTS="-Dquarkus.http.host=0.0.0.0 -
Djava.util.logging.manager=org.jboss.logmanager.LogManager"
ENV JAVA_APP_JAR="/deployments/quarkus-run.jar"
```

Fuente: Elaboración Propia

4.2.3. Fase de revisión y retrospectiva

En las etapas descritos en la tabla 13 del tercer capítulo, en la fase IV de revisión y retrospectiva de la metodología Scrum, facilita llevar a cabo la confirmación del sprint, alcanzar los criterios de aceptación y cumplir con la revisión retrospectiva del sprint para lograr resultados satisfactorios.

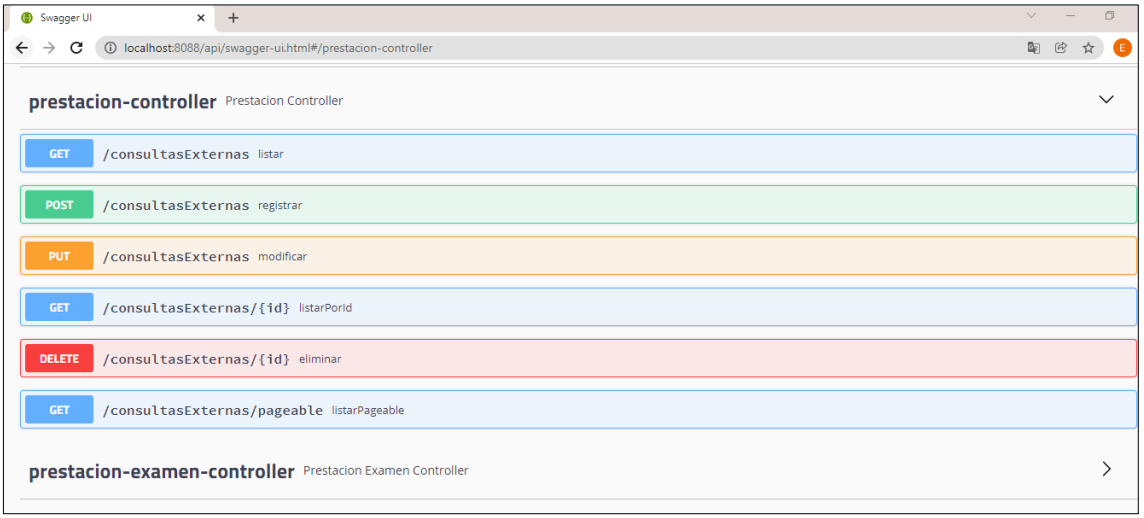
Figura 10

Reporte de la documentación de las apis

Fuente: Elaboración Propia

Figura 11

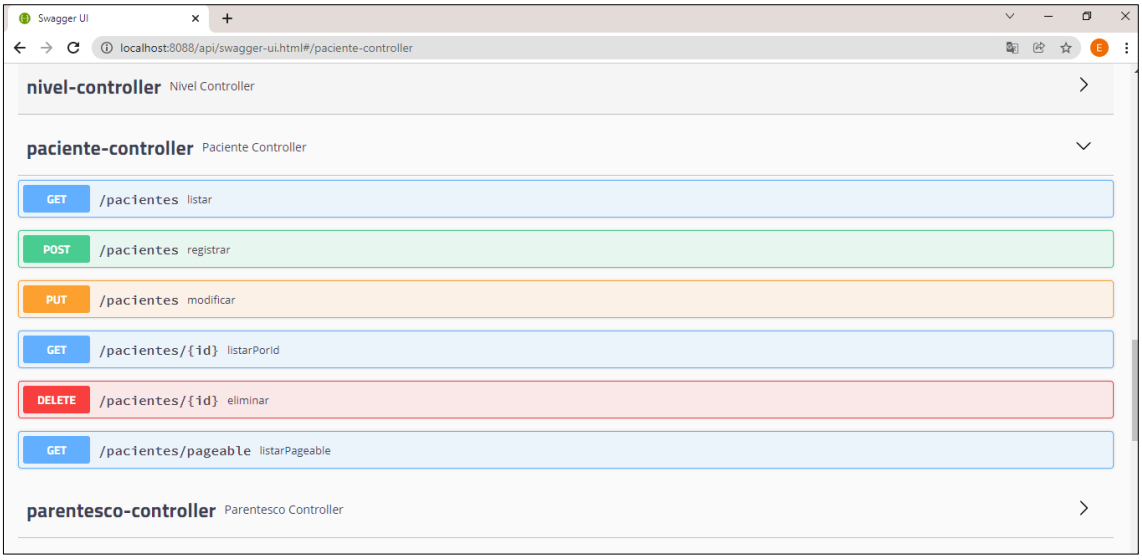
Documentación del servicio web consulta externa



Fuente: Elaboración Propia

Figura 12

Documentación de servicio web pacientes

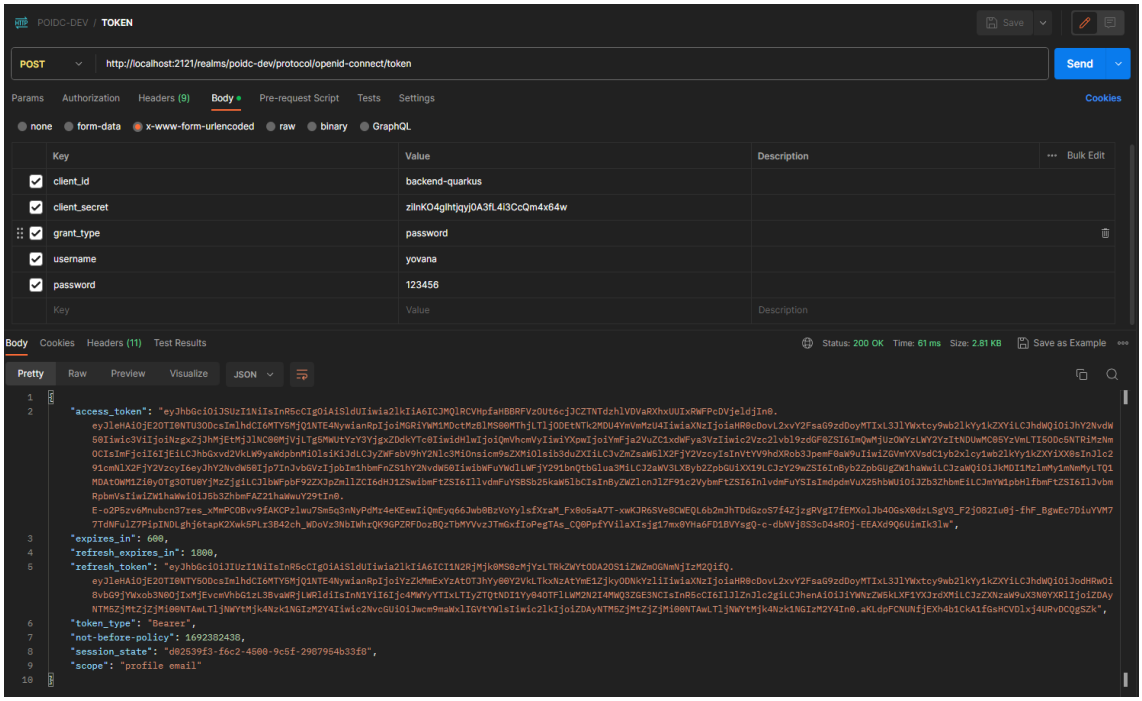


Nota. La figura muestra los 6 endpoints con 4 tipos de petición; Get, Post, Put y Delete donde se detalla los parámetros para cada tipo.

Fuente: Elaboración Propia

Figura 13

Prueba de solicitud de token de acceso



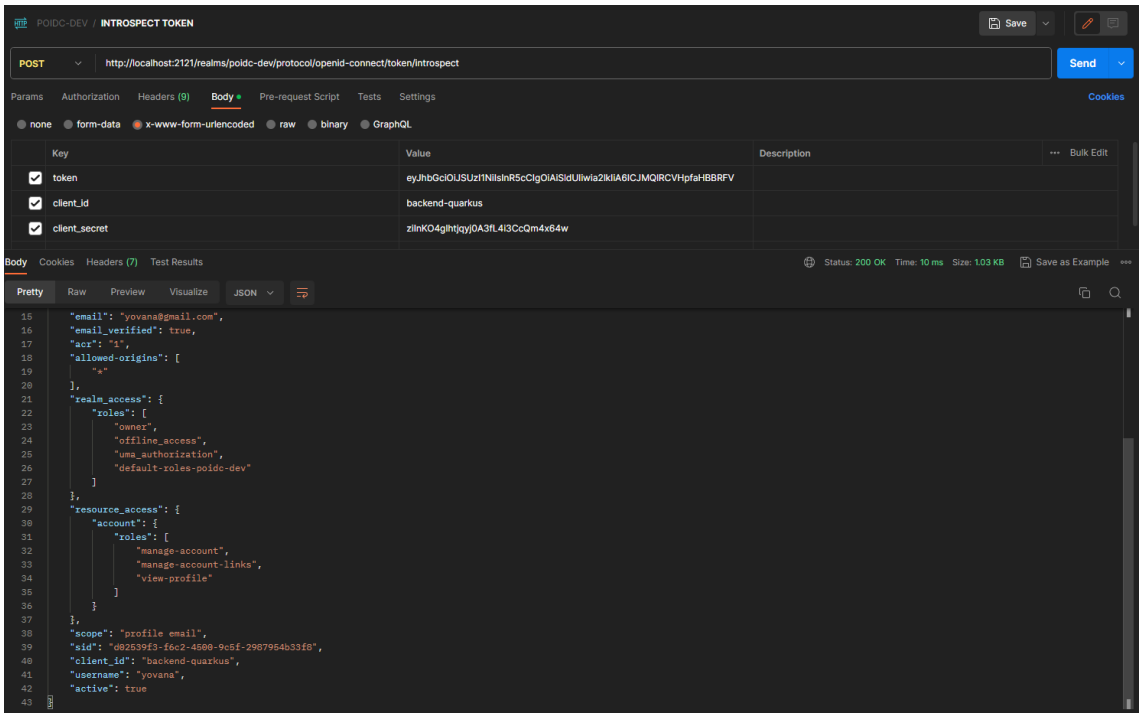
Nota. La prueba se realizó con la herramienta Postman y con el endpoint

http://localhost:2121/realms/poidc-dev/protocol/openid-connect/token, obteniendo respuesta satisfactoria.

Fuente: Elaboración Propia

Figura 14

Prueba de solicitud de datos del usuario

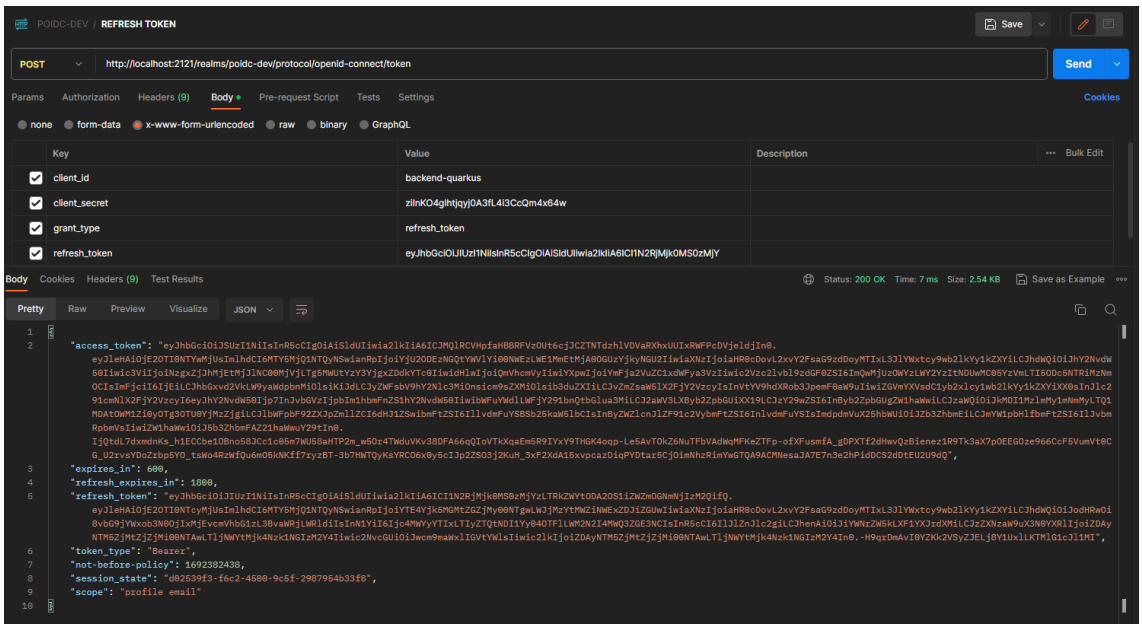


Nota. La prueba se realizó con la herramienta Postman y con el endpoint http://localhost:2121/realms/poidc-dev/protocol/openid-connect/token/introspect, obteniendo respuesta satisfactoria.

Fuente: Elaboración Propia

Figura 15

Prueba de generación del nuevo token a partir del refresh token

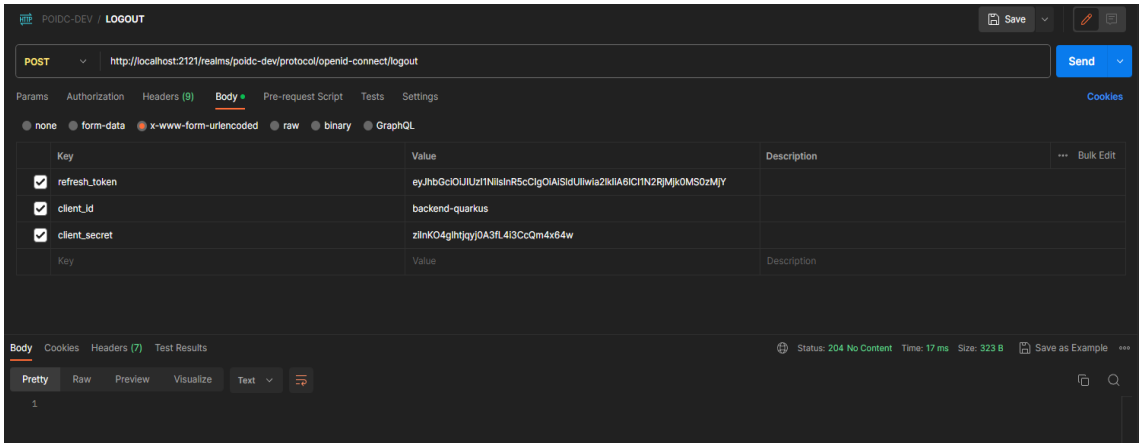


Nota. La prueba se realizó con la herramienta Postman y con el endpoint http://localhost:2121/realms/poidc-dev/protocol/openid-connect/token, obteniendo respuesta satisfactoria.

Fuente: Elaboración Propia

Figura 16

Prueba de cierre de sesión



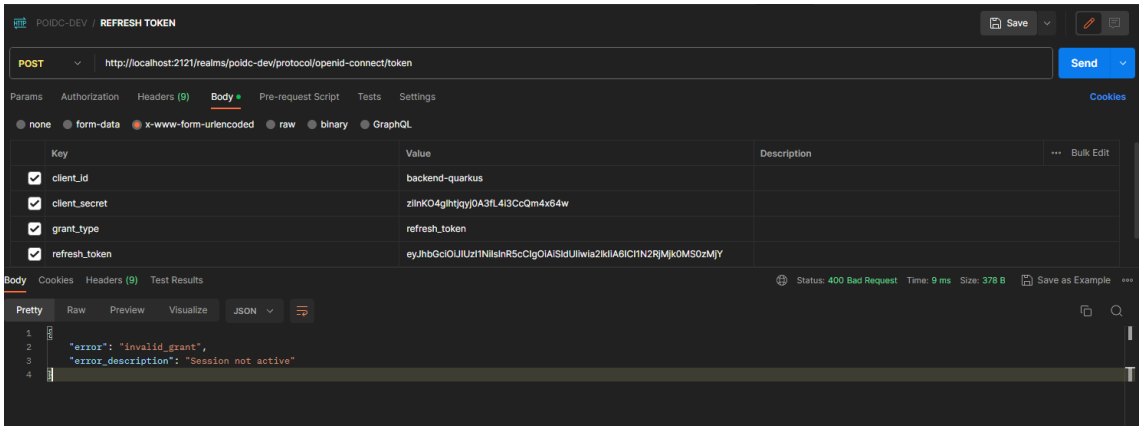
Nota. La prueba se realizó con la herramienta Postman y con el endpoint http://localhost:2121/realms/poidc-dev/protocol/openid-connect/logout, obteniendo

respuesta satisfactoria.

Fuente: Elaboración Propia

Figura 17

Prueba de intento de volver a obtener el Access token usando el refresh cuando la sesión ha sido cerrada.

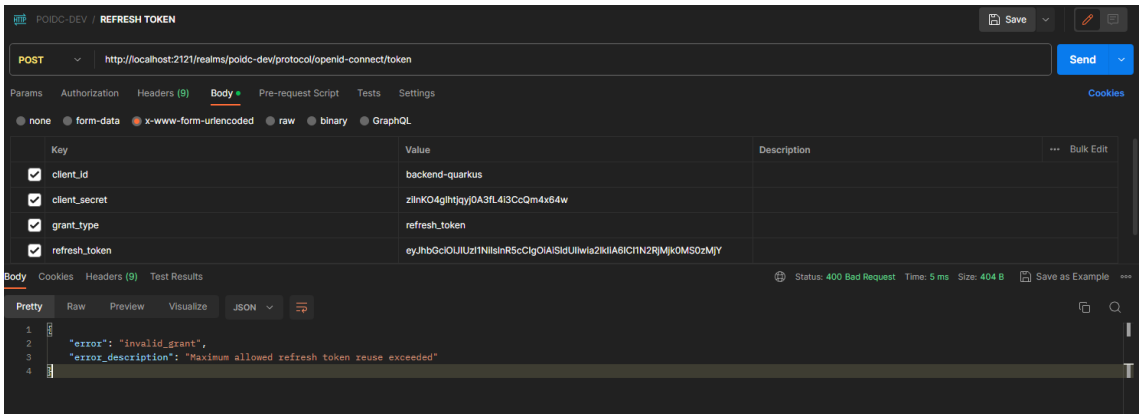


Nota. La prueba se realizó con la herramienta Postman y con el endpoint http://localhost:2121/realms/poidc-dev/protocol/openid-connect/token, obteniendo respuesta satisfactoria.

Fuente: Elaboración Propia

Figura 18

Prueba de 2 reintentos del refresh token (obtener token infinitamente)



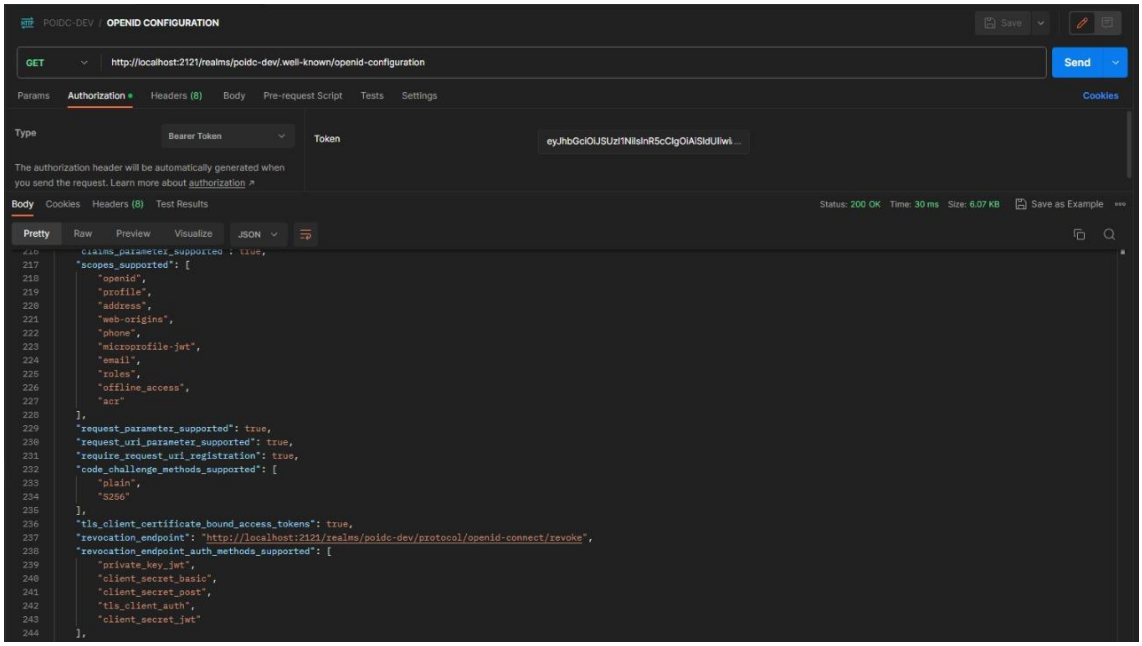
Nota. La prueba se realizó con la herramienta Postman y con el endpoint http://localhost:2121/realms/poidc-dev/protocol/openid-connect/token, obteniendo

respuesta satisfactoria.

Fuente: Elaboración Propia

Figura 19

Configuración del Alcance de OpenId Connect



Fuente: Elaboración Propia

Tabla 83

Configuración de Openid Connect (keycloak.conf)

```
# Basic settings for running in production. Change accordingly before deploying the
server.

# Database

# The database vendor.
db=mariadb

# The username of the database user.
db-username=root

# The password of the database user.
db-password=password

# The full database JDBC URL. If not provided, a default URL is set based on the
selected database vendor.
db-url=jdbc:mariadb://localhost:3306/keycloak
```

```
# Observability

# If the server should expose healthcheck endpoints.
#health-enabled=true

# If the server should expose metrics endpoints.
#metrics-enabled=true

# HTTP

# The file path to a server certificate or certificate chain in PEM format.
#https-certificate-file=${kc.home.dir}conf/server.crt.pem

# The file path to a private key in PEM format.
#https-certificate-key-file=${kc.home.dir}conf/server.key.pem

# The proxy address forwarding mode if the server is behind a reverse proxy.
#proxy=reencrypt

# Do not attach route to cookies and rely on the session affinity capabilities from
reverse proxy
#spi-sticky-session-encoder-infinispan-should-attach-route=false

# Hostname for the Keycloak server.
#hostname=myhostname

http-port=2121
```

Fuente: Elaboración Propia

Tabla 84

Docker compose de la base de datos (mariaDB)

```yaml
version: '3.1'

services:
  mariadb:
    image: mariadb:10.6
    container_name: mariadb
    environment:
      MYSQL_ROOT_PASSWORD: password
      MYSQL_DATABASE: mydb
      MYSQL_USER: user
      MYSQL_PASSWORD: password
    ports:
      - "3306:3306"
    volumes:
```

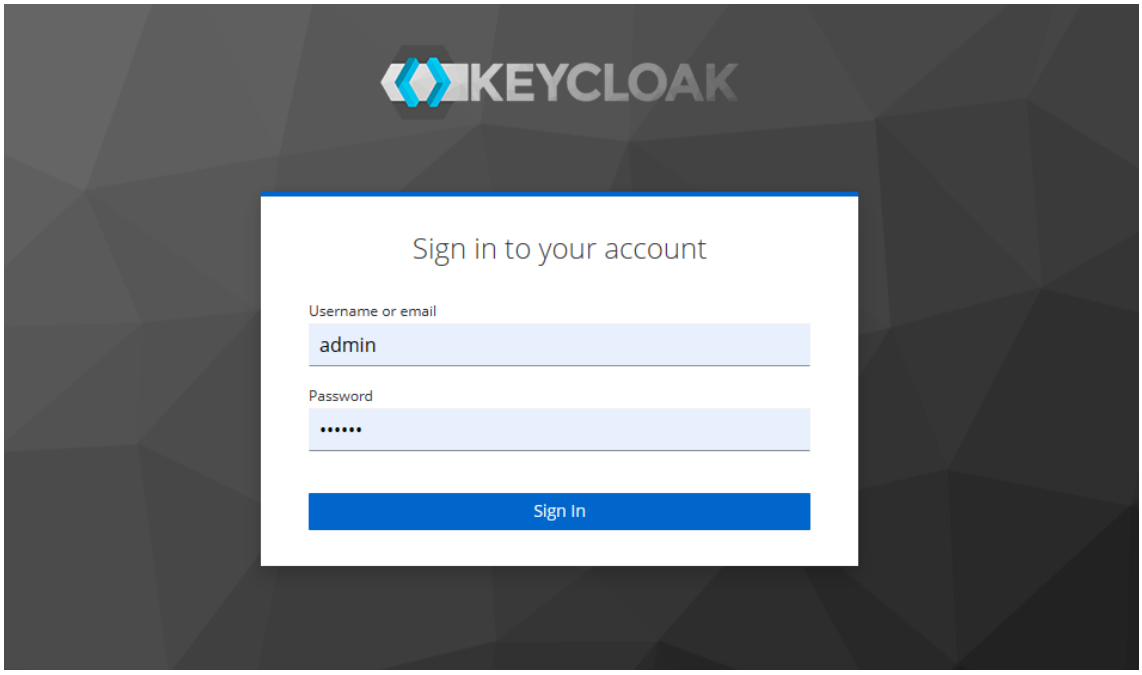
```
      - mariadb_data:/var/lib/mysql

volumes:
  mariadb_data:
```

Fuente: Elaboración Propia

Figura 20

Ingreso a OpenId connect (keycloak)

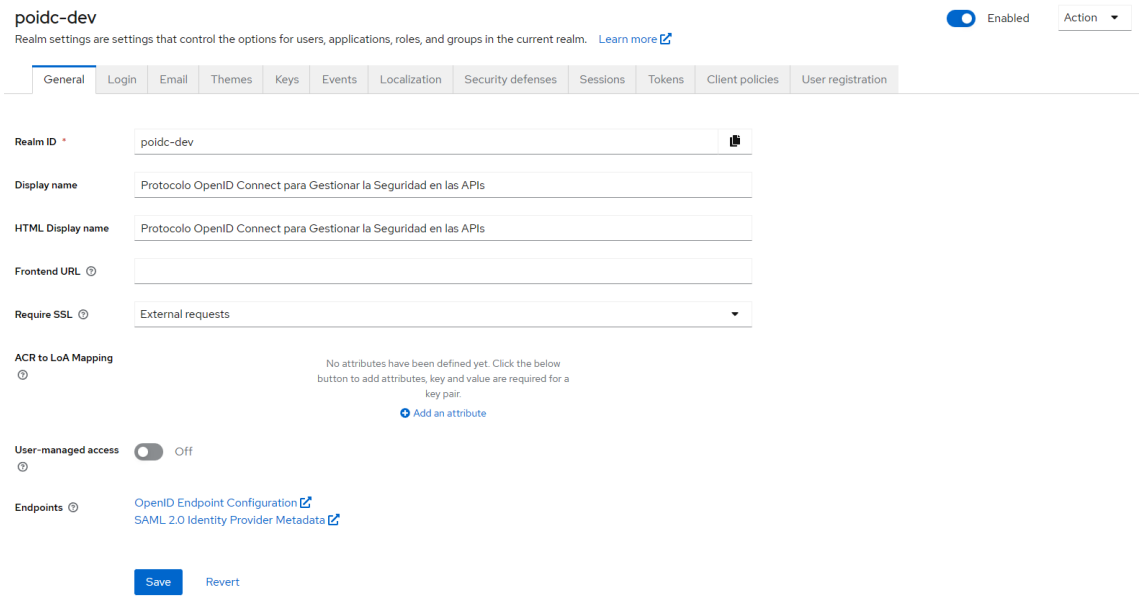


Nota. Para la configuración de todos los parámetros del api se usará la herramienta de autorización keycloak.

Fuente: Elaboración Propia

Figura 21

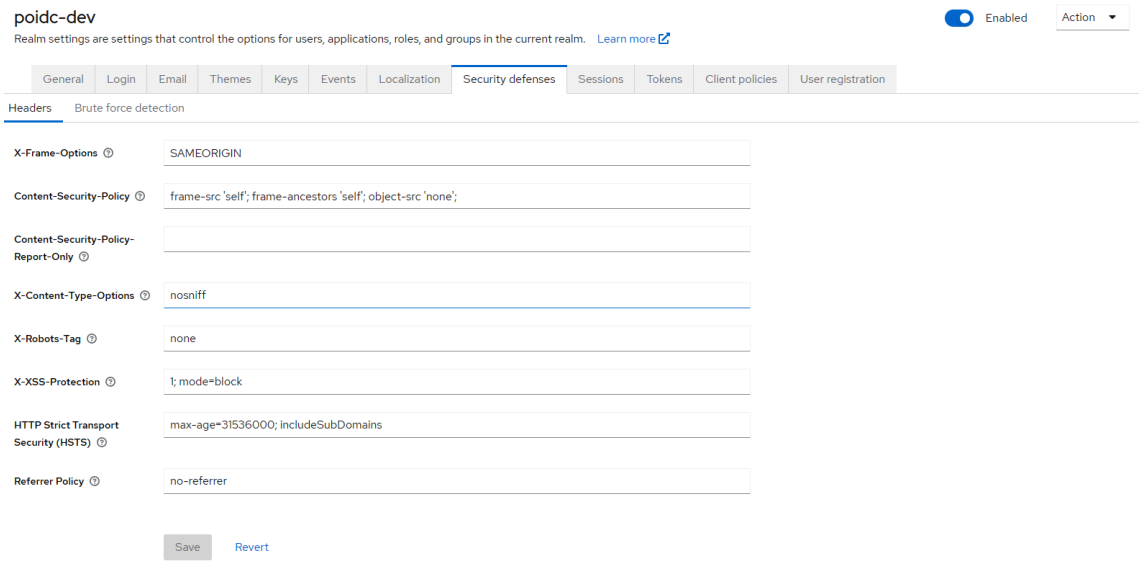
Creación del reino o realm



Fuente: Elaboración Propia

Figura 22

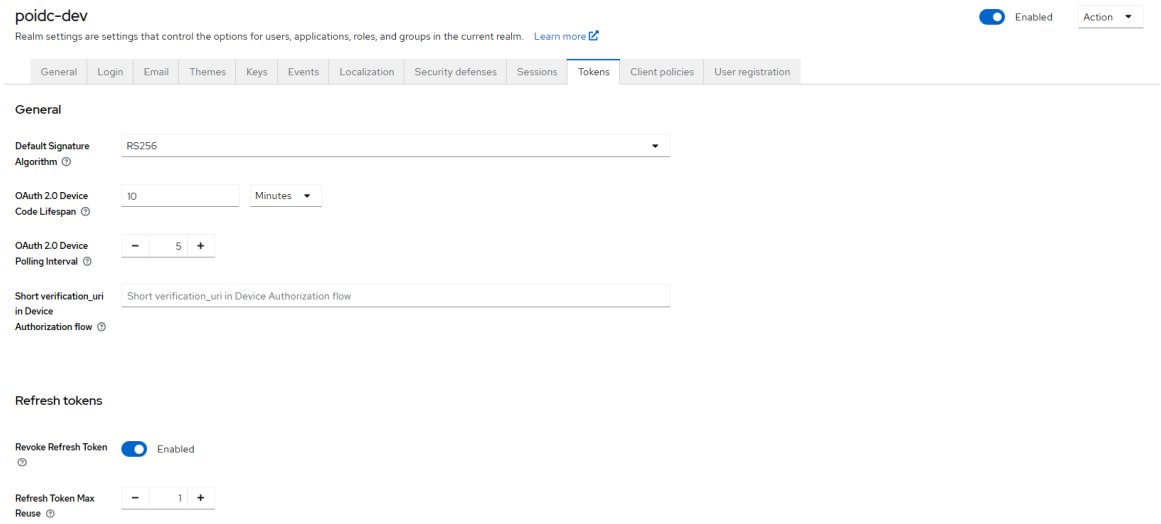
Configuración de CSP (Content security policy)



Fuente: Elaboración Propia

Figura 23

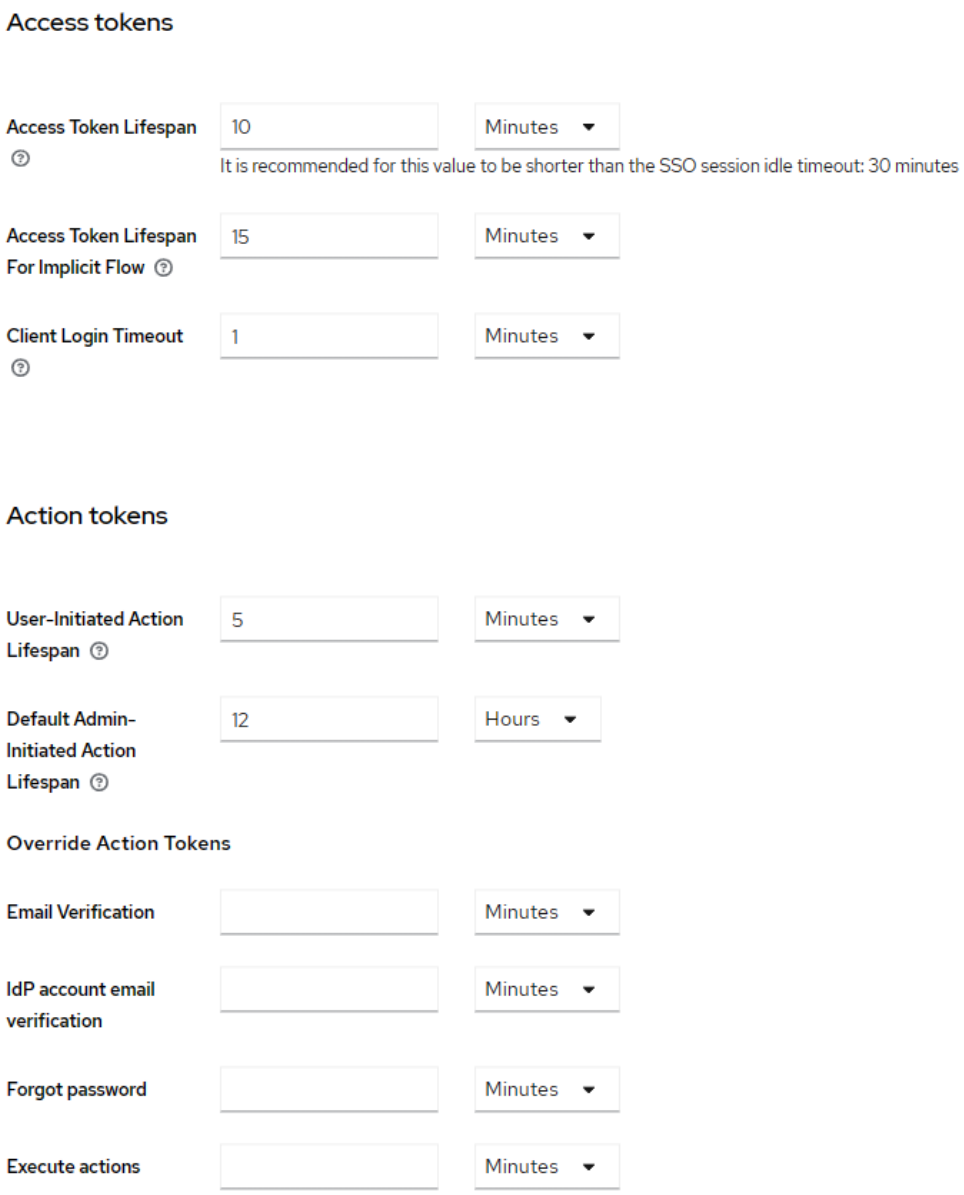
Configuración del algoritmo de cifrado de token, tiempo de vida del token y cantidad de usos del refresh token (1/2)



Fuente: Elaboración Propia

Figura 24

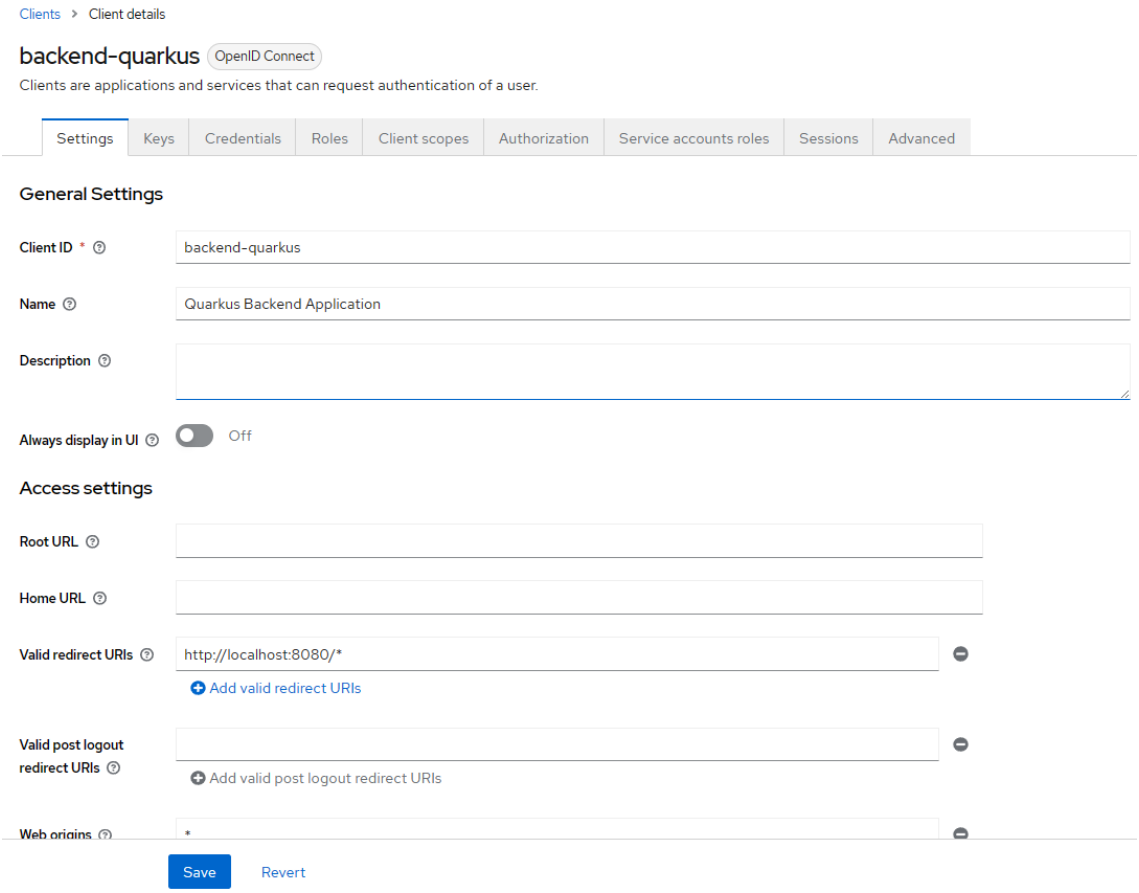
Configuración del algoritmo de cifrado de token, tiempo de vida del token y cantidad de usos del refresh token (1/2)



Fuente: Elaboración Propia

Figura 25

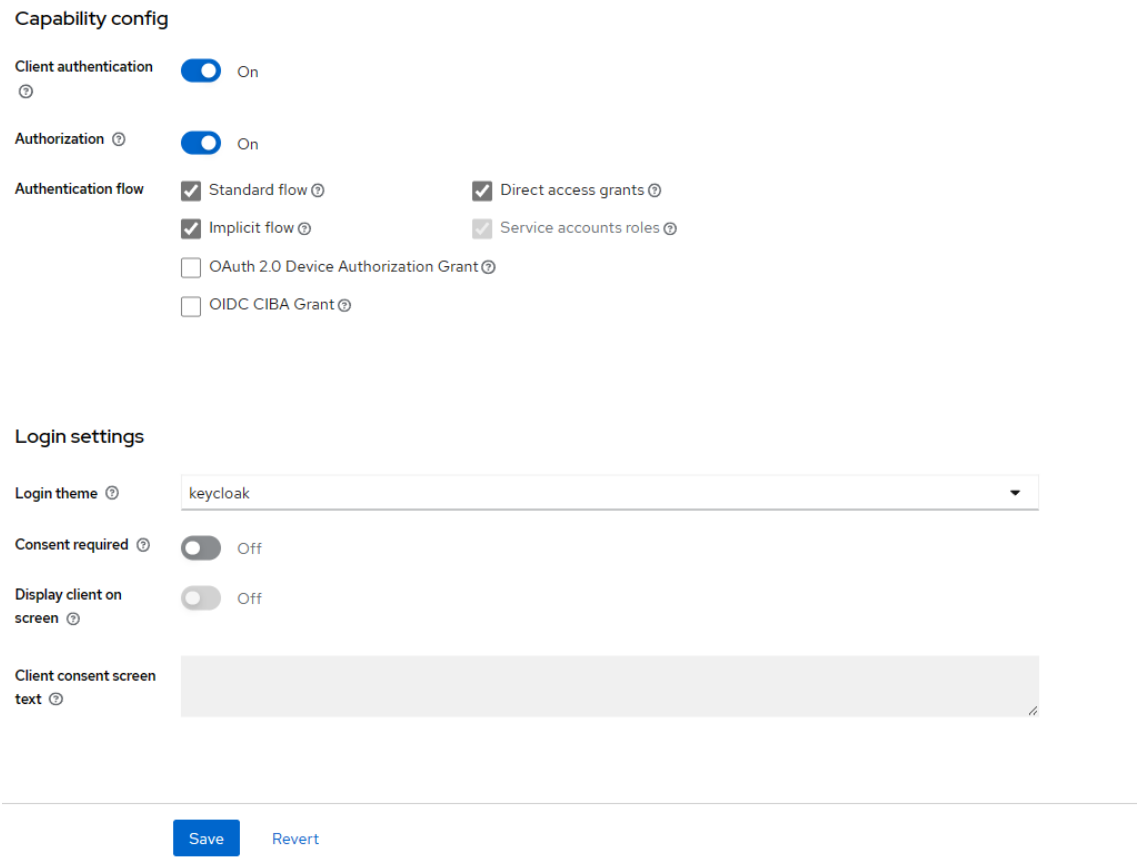
Configuración del cliente (quarkus-backend)



Fuente: Elaboración Propia

Figura 26

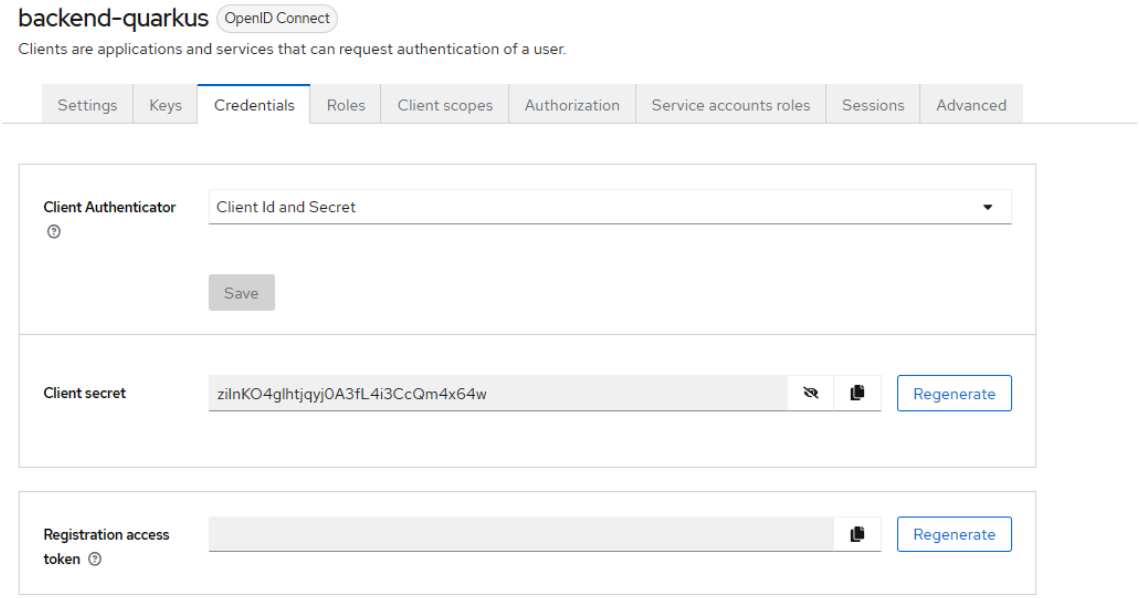
Configuración de credenciales para el cliente y autorización



Fuente: Elaboración Propia

Figura 27

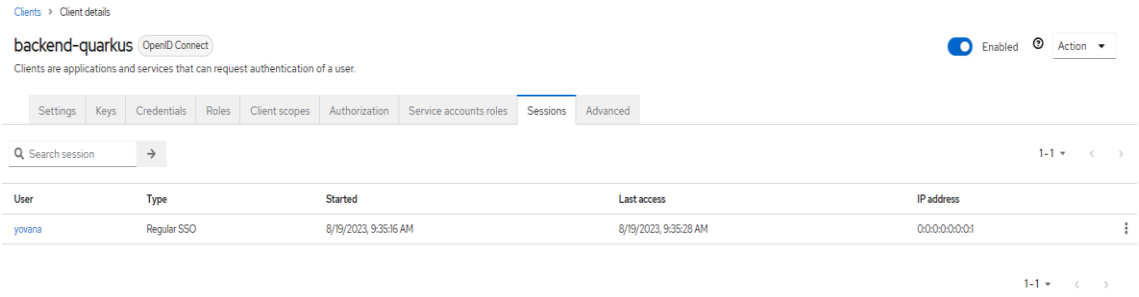
Credenciales del cliente



Fuente: Elaboración Propia

Figura 28

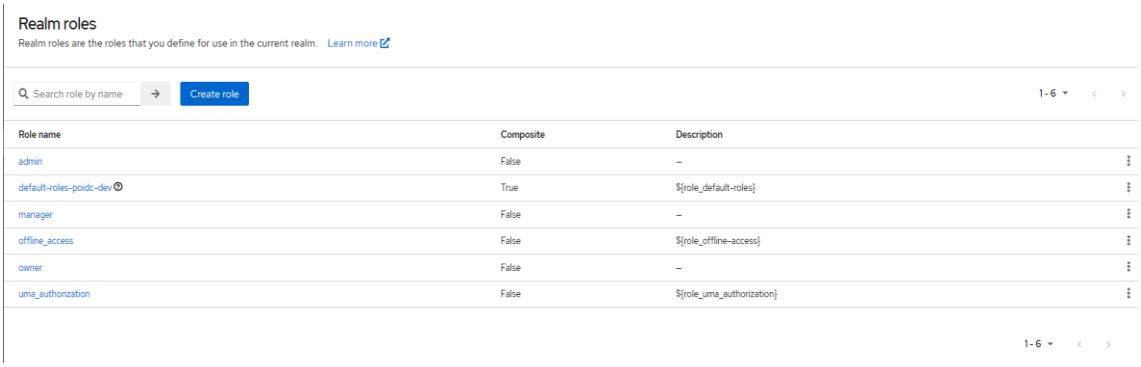
Sección de las sesiones iniciadas con el cliente



Fuente: Elaboración Propia

Figura 29

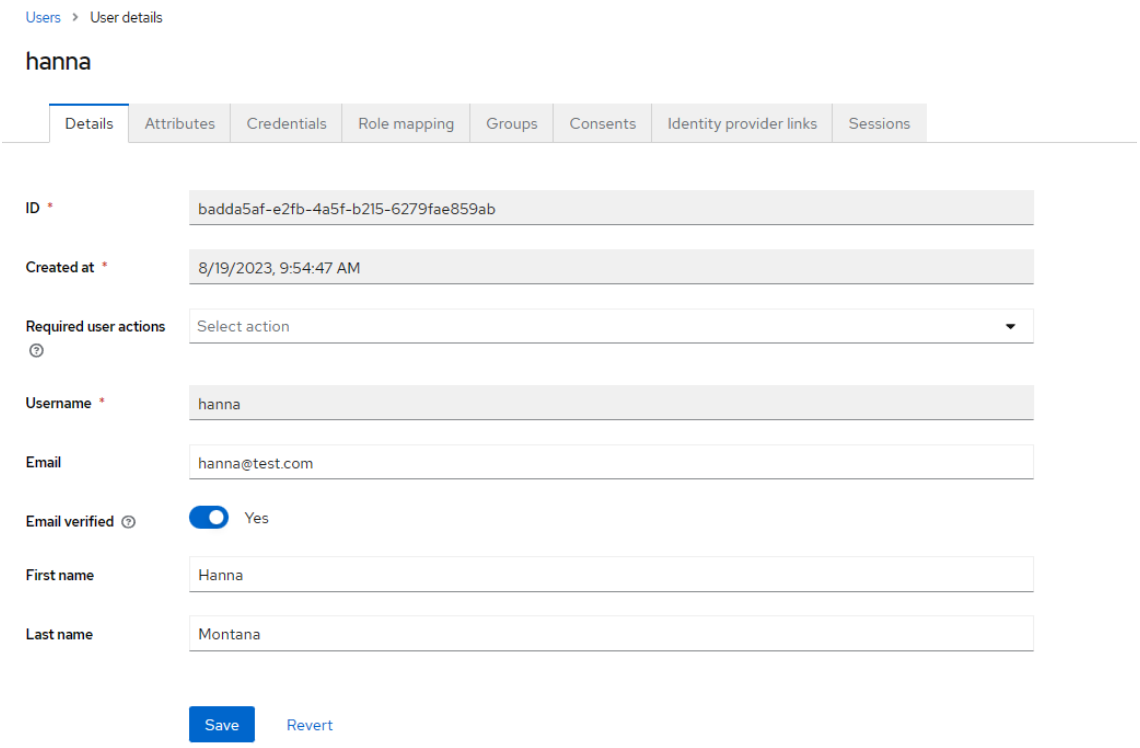
Creación de roles (admin, manager y owner)



Fuente: Elaboración Propia

Figura 30

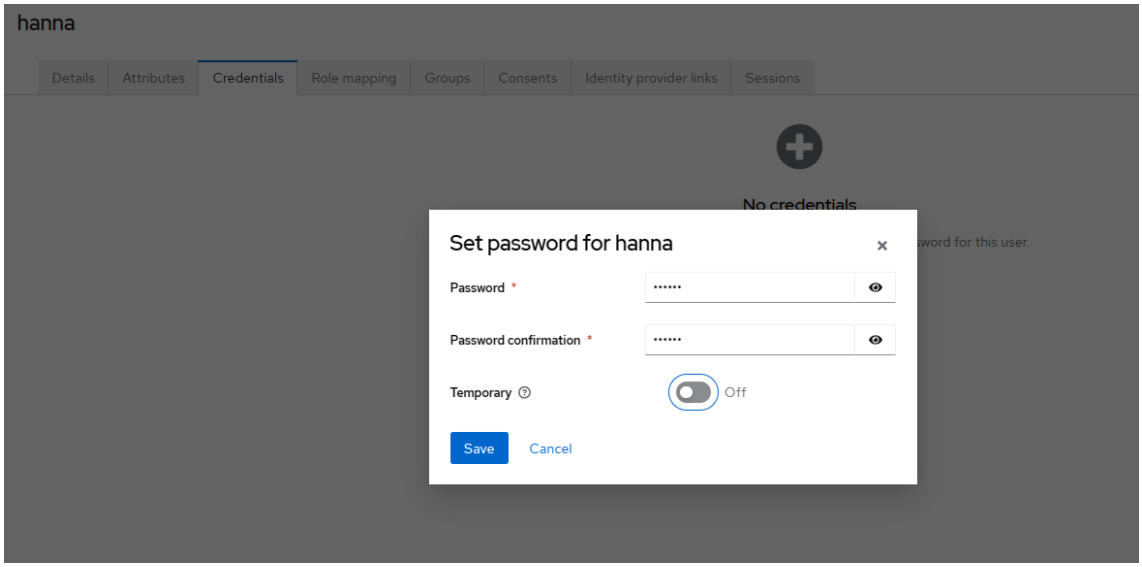
Creación de usuarios



Fuente: Elaboración Propia

Figura 31

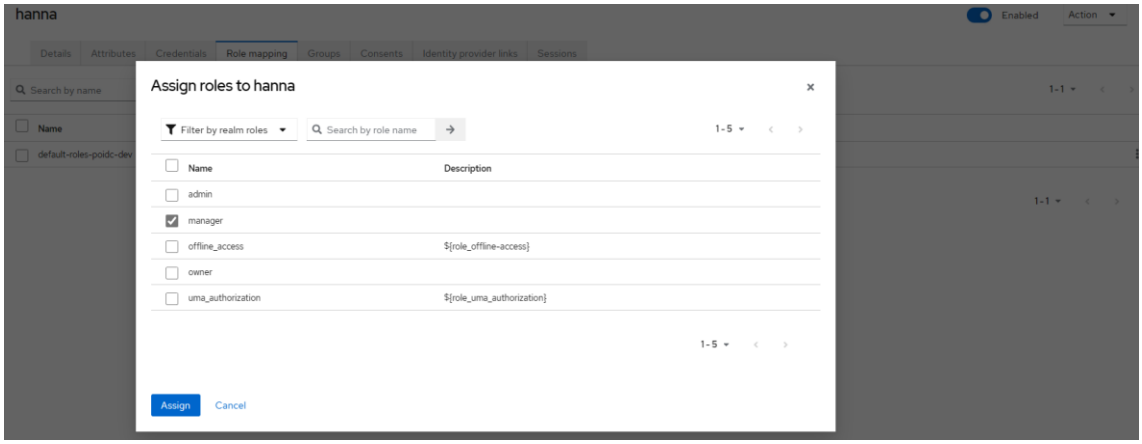
Generación de contraseña del usuario



Fuente: Elaboración Propia

Figura 32

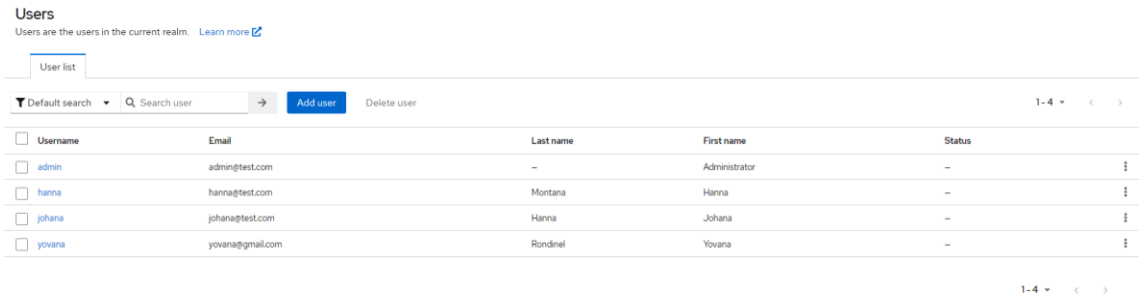
Asignación del rol manager al usuario



Fuente: Elaboración Propia

Figura 33

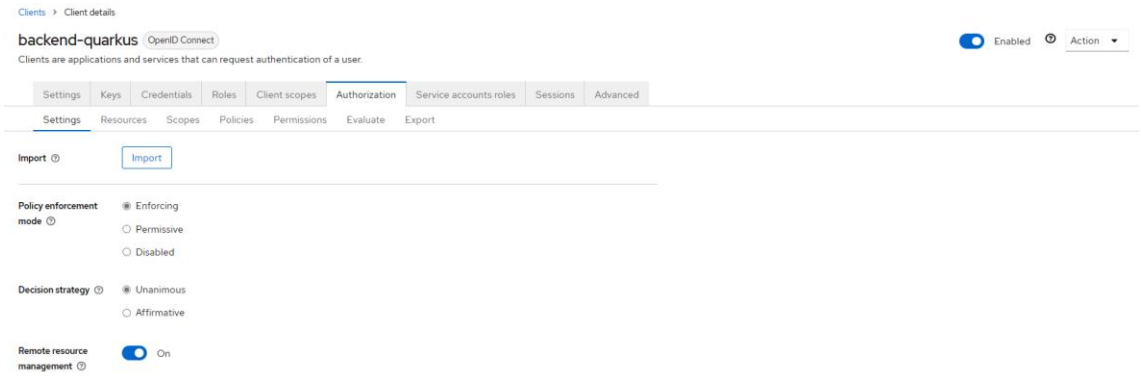
Usuarios configurados



Fuente: Elaboración Propia

Figura 34

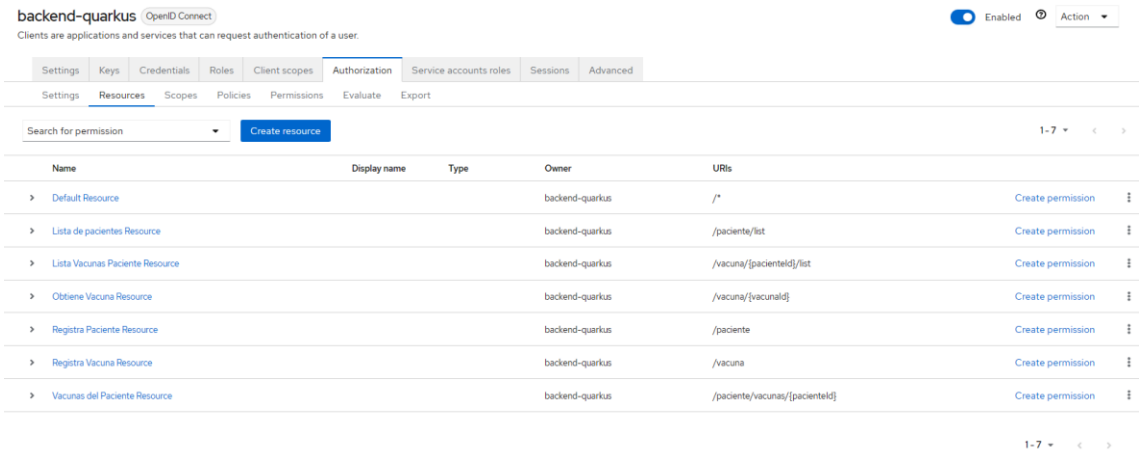
Configuración de autorizaciones permitidas en el backend



Fuente: Elaboración Propia

Figura 35

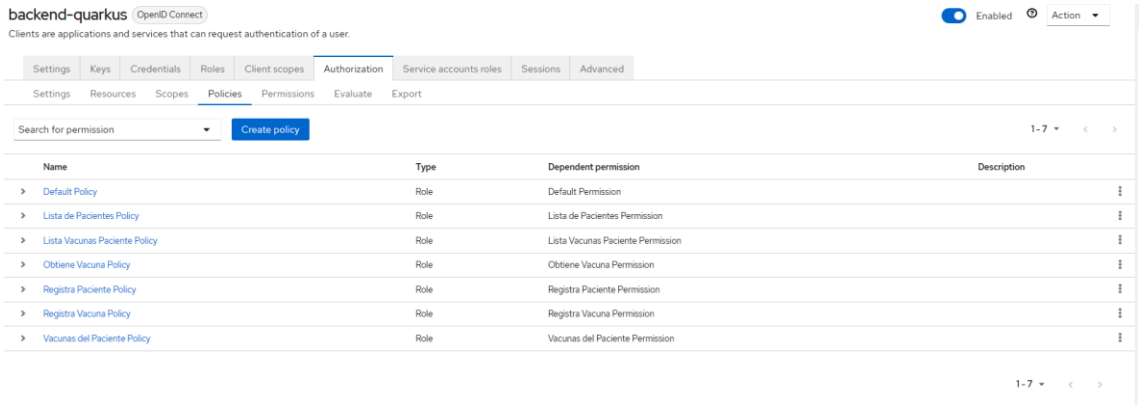
Configuración de recursos o endpoint(apis) del backend



Fuente: Elaboración Propia

Figura 36

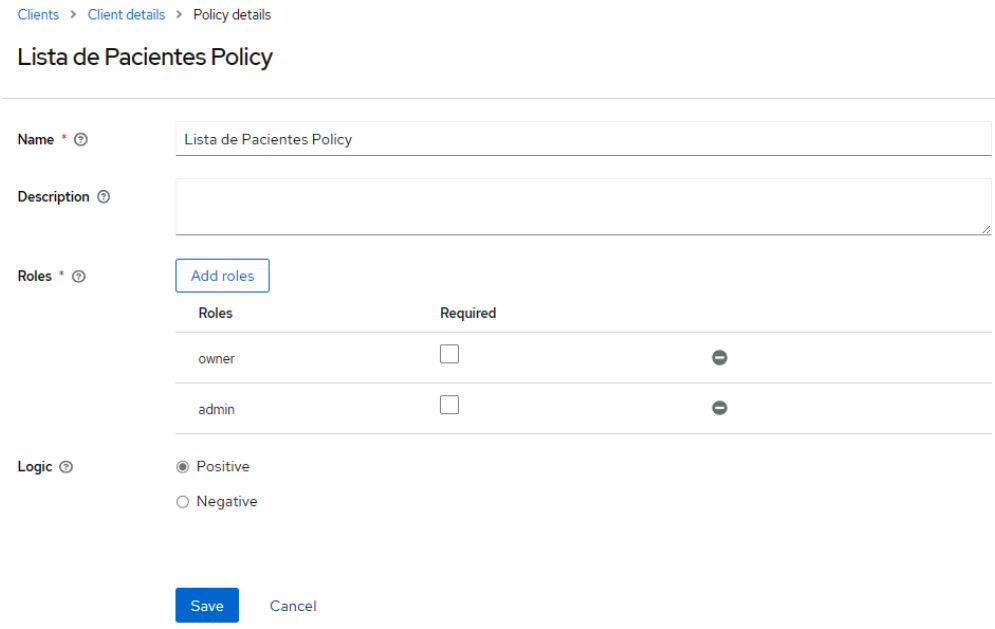
Configuración de las políticas de acceso a los recursos (basados en roles)



Fuente: Elaboración Propia

Figura 37

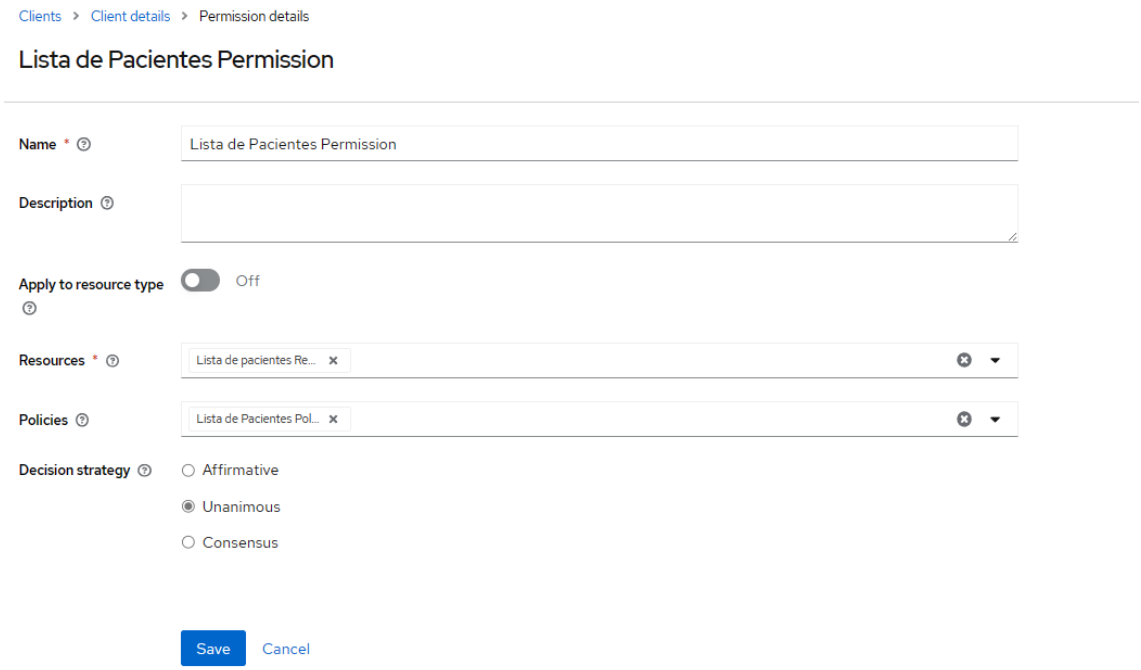
Lista de Pacientes Policy permite dos tipos de roles para su acceso



Fuente: Elaboración Propia

Figura 38

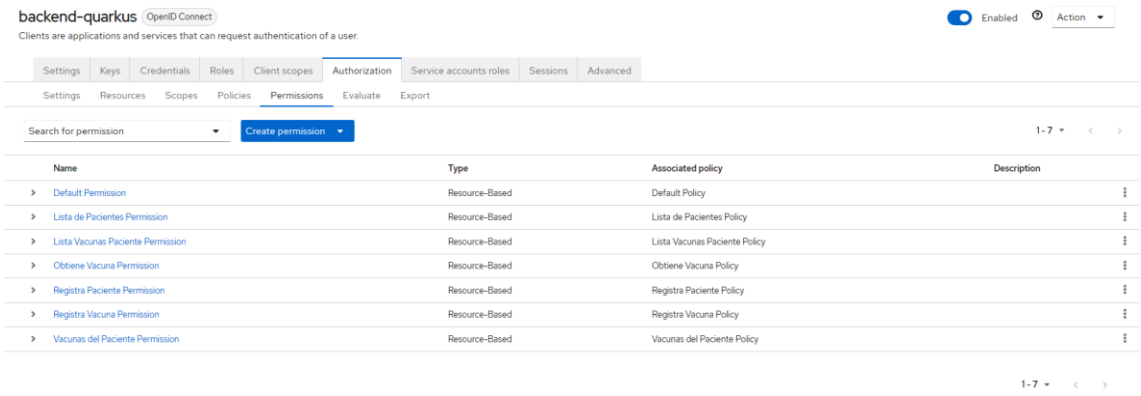
Configuración y asignación de permisos a las políticas y recursos



Fuente: Elaboración Propia

Figura 39

Permisos configurados para acceso a recursos y políticas



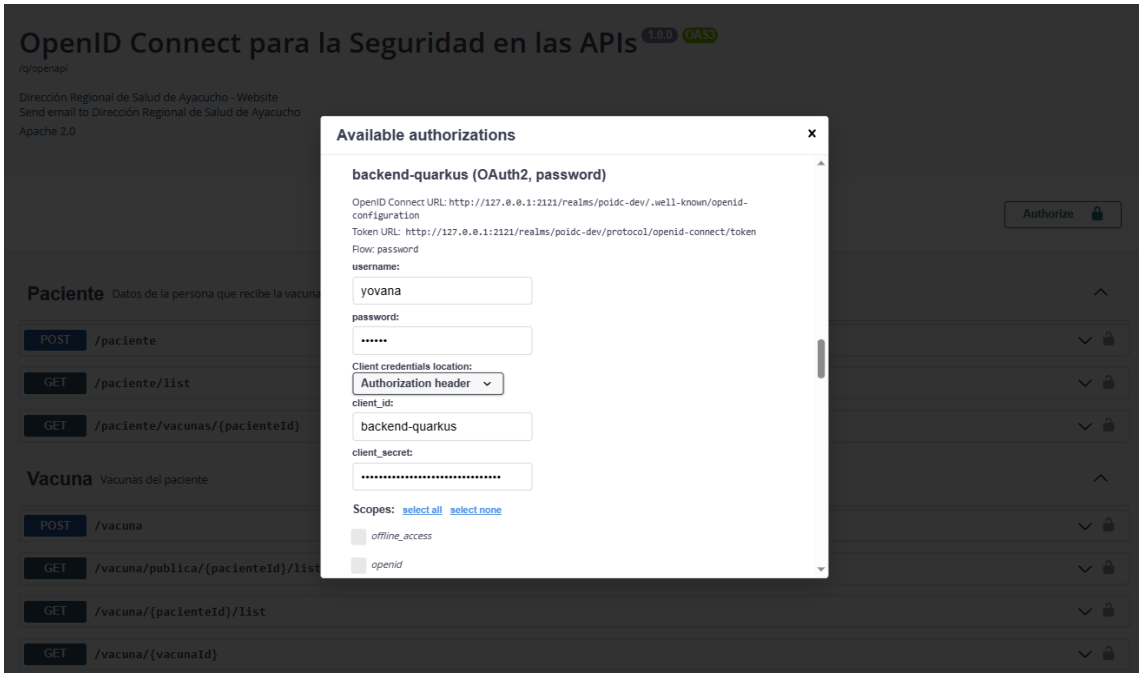
Fuente: Elaboración Propia

4.2.4. Fase de lanzamiento

En las etapas descritos en la tabla 14 del tercer capítulo, en la fase V de lanzamiento de la metodología Scrum, posibilita la revisión de resultados, determinar lanzamiento del producto y la retrospectiva del proyecto.

Figura 40

Prueba del Login con swagger



Fuente: Elaboración Propia

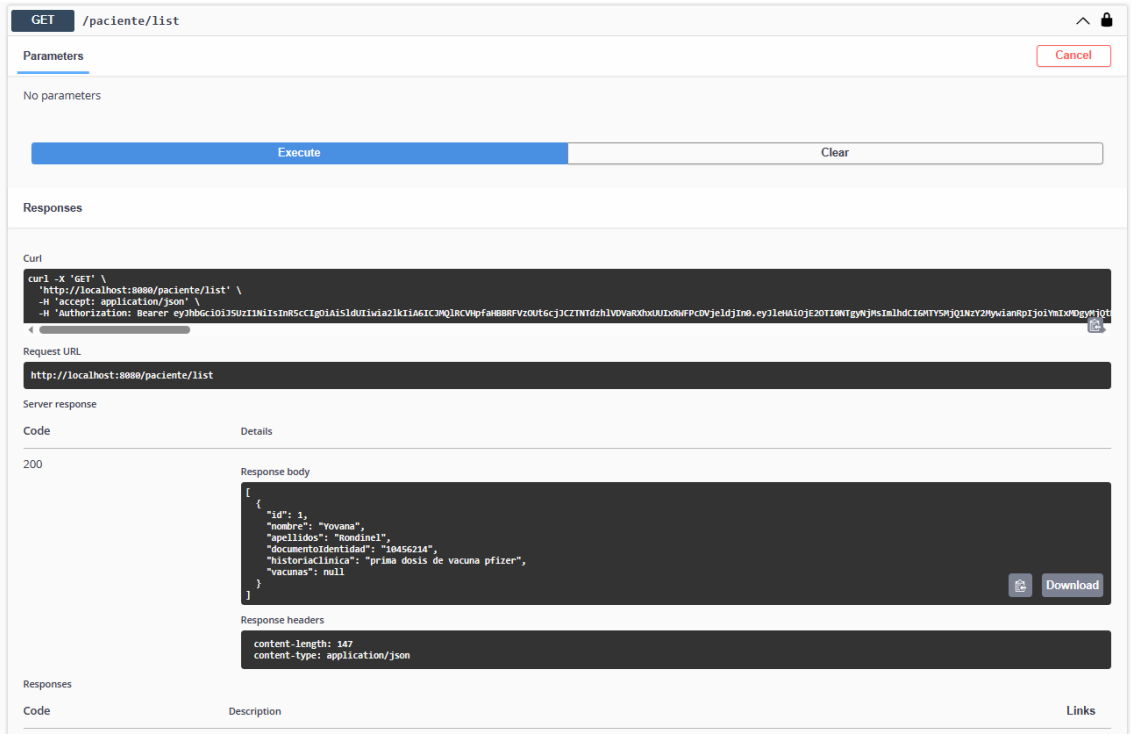
Figura 41

Ingreso correcto con el usuario yovana, rol owner

Fuente: Elaboración Propia

Figura 42

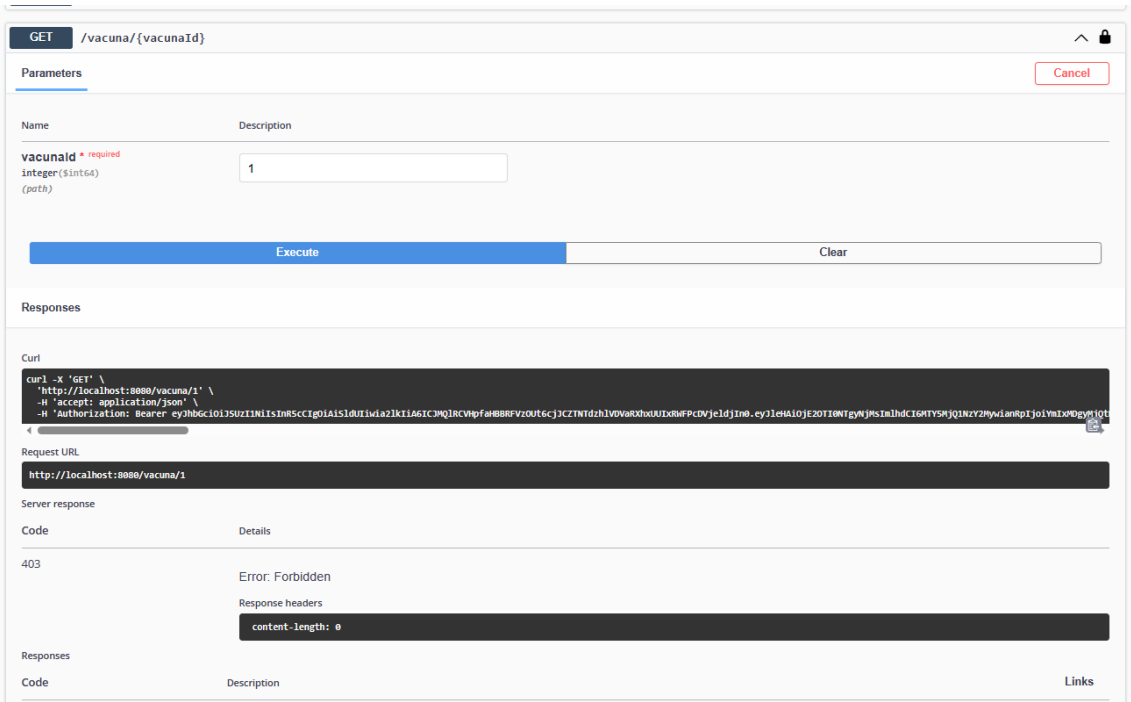
Consulta la lista de pacientes con Usuario Yovana, rol owner



Fuente: Elaboración Propia

Figura 43

Consulta a vacuna con el id de vacuna con Usuario Yovana, rol owner

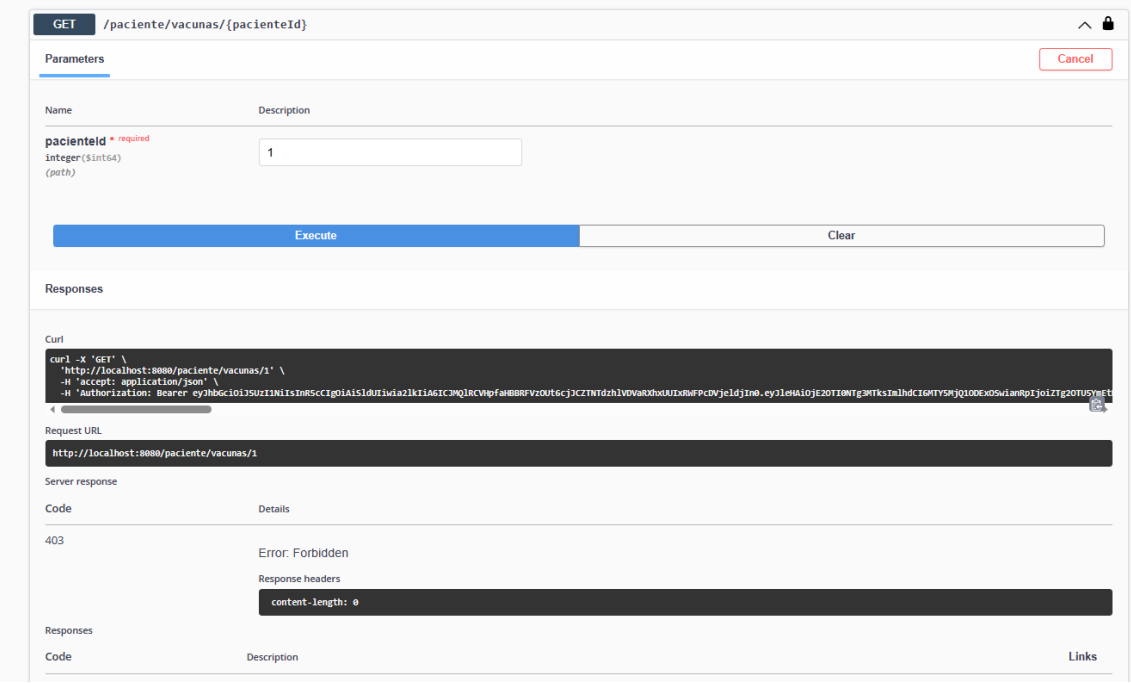


Nota: no obtiene resultados y el response marca como prohibido, ya que el usuario no tiene rol manager y también el recurso está configurado para que pueda ser consumida si los usuarios tienen el rol manager.

Fuente: Elaboración Propia

Figura 44

Consulta a cualquier recurso con usuario Johana sin rol

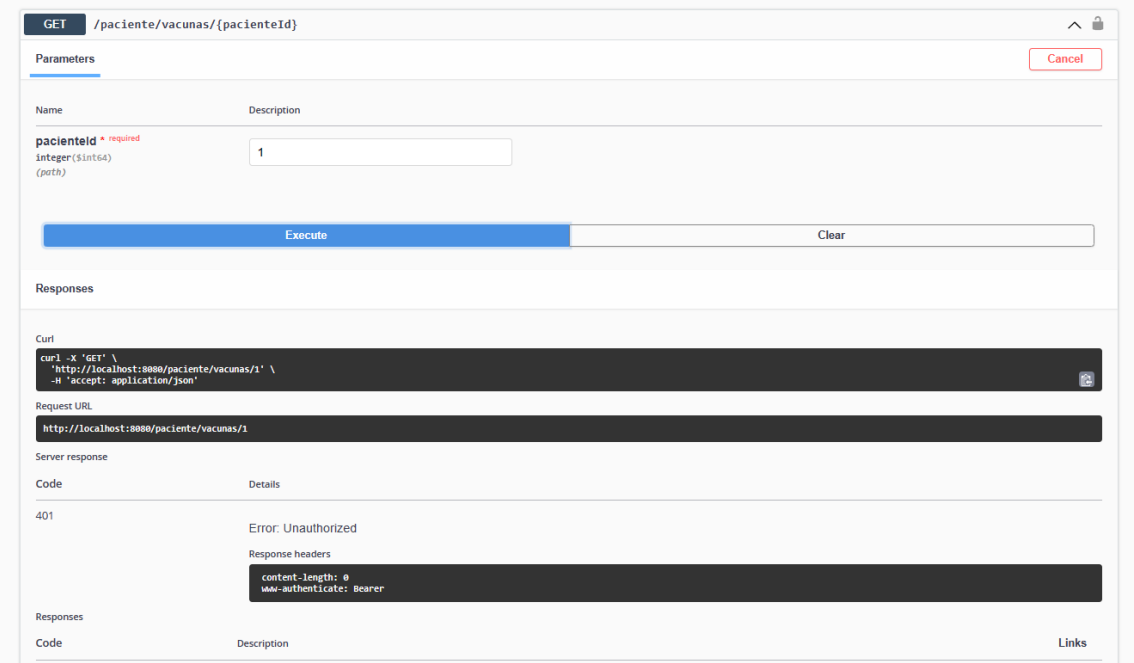


Nota: el resultado muestra error: prohibido ya que los recursos solo pueden ser accedidos según la configuración de las políticas de acceso.

Fuente: Elaboración Propia

Figura 45

Prueba del consumo de recurso sin logearse

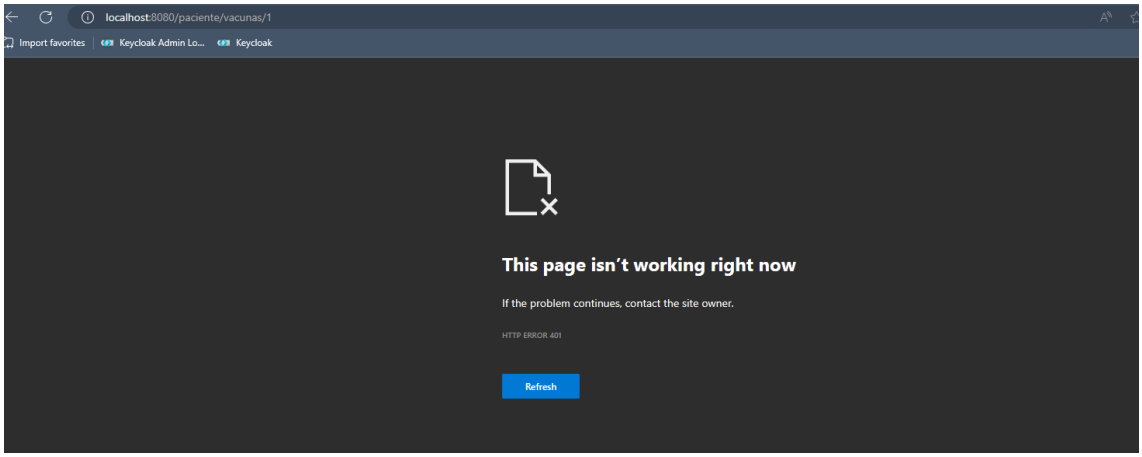


Nota: El recurso retorna el error: no autorizado, ya que no se cumple con las políticas configuradas y el usuario no está logeado.

Fuente: Elaboración Propia

Figura 46

Probando desde la url del navegador

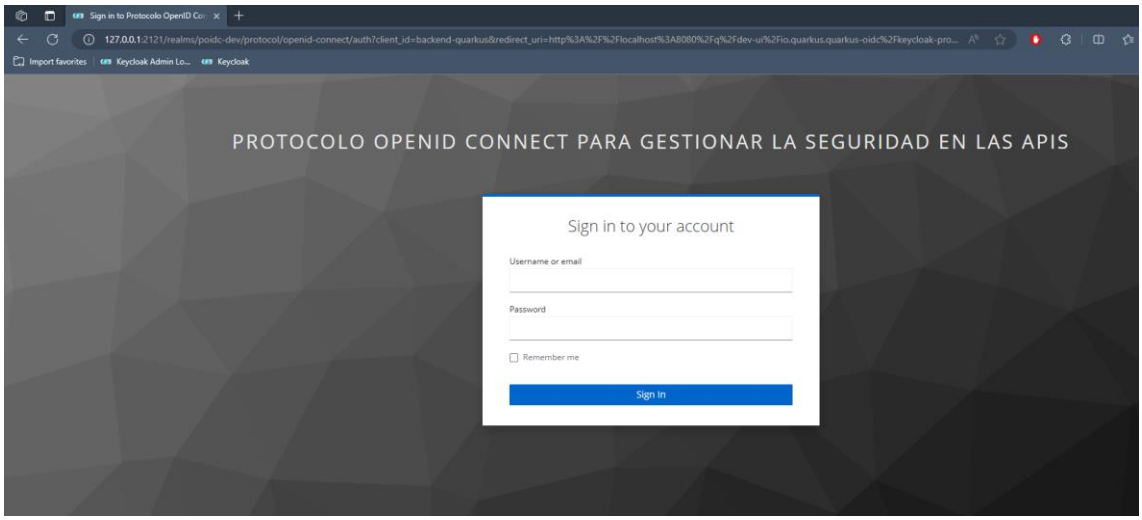


Nota: el navegador muestra el error 401 (Unauthorized), esto es debido a que no se cumple con las políticas configuradas.

Fuente: Elaboración Propia

Figura 47

Probando redirect_uri desde una aplicación cliente



Nota: pantalla de inicio de sesión de tipo SSO, al ingresar el usuario y contraseña automáticamente te envía a la pantalla de donde fue llamado.

url:http://127.0.0.1:2121/realms/poidc-dev/protocol/openid-connect/auth?client_id=backend-quarkus&redirect_uri=http%3A%2F%2Flocalhost%3A8080%2Fq%2Fdev-ui%2Fio.quarkus.quarkus-oidc%2Fkeycloak-provider&scope=openid&response_type=code&response_mode=query&prompt=login&nonce=Rpr5xeW&state=cXIAZOO

Fuente: Elaboración Propia

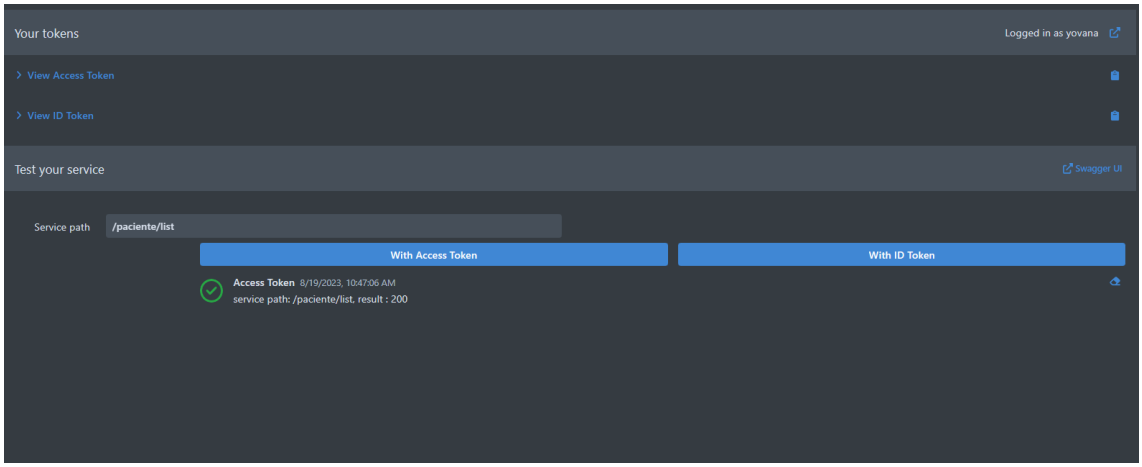
Figura 48

Id del token después del logeo

Fuente: Elaboración Propia

Figura 49

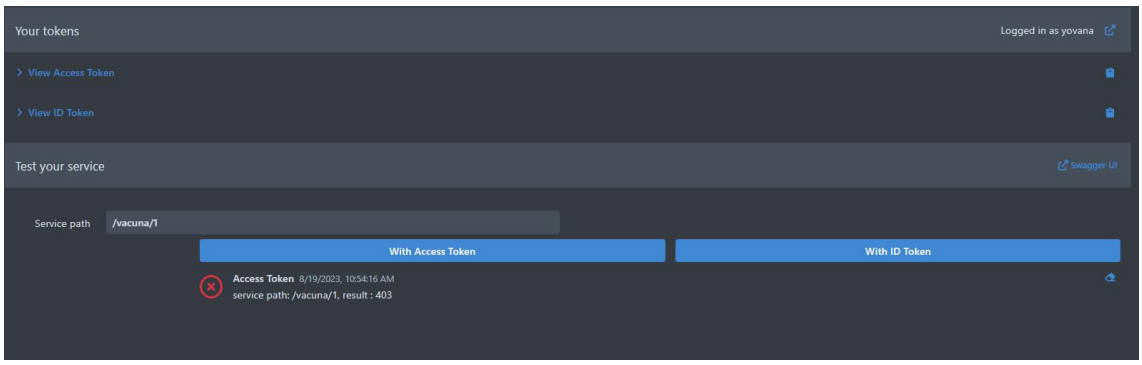
Probando el recurso /paciente/list con el usuario yovana



Fuente: Elaboración Propia

Figura 50

Probando el recurso /vacuna/{pacienteId}/list con el usuario yovana



Nota: sale error porque no tiene autorización para este recurso.

Fuente: Elaboración Propia

4.3. DISCUSIÓN

Según Sujanani & Vinod (2018), OpenId connect permitió un acceso más eficiente a las diferentes aplicaciones, ya que los usuarios necesitaron autenticarse solo una vez para acceder a múltiples servicios de estas aplicaciones. En el estudio se ha implementado OpenId connect en Swagger para las APIs. Entonces, se logra mayor seguridad en la autenticación y autorización a los recursos de los microservicios.

Según Li & Mitchell (2020), que para garantizar una verdadera protección de la privacidad es necesario realizar modificaciones sustanciales en el funcionamiento de los protocolos OpenId Connect. En el estudio se ha implementado el alcance(scope) en los microservicios orquestados por swagger, en donde se limita el acceso a ciertos recursos de la aplicación.

De acuerdo a Singh & Chaudhary (2022), incorporando los tokens de acceso con tiempo de vida limitado como característica para la autenticación de aplicaciones cliente de terceros mejoró significativamente la seguridad de las aplicaciones. En el estudio se ha implementado y configurado el Access Token y Refresh token para mayor seguridad en las Apis expuestas por Swagger.

CAPÍTULO V

CONCLUSIONES

5.2. CONCLUSIONES

a. En el presente estudio se definió el alcance de la autorización del Access Token para gestionar la seguridad en las Apis expuestas por Swagger, logrando controlar la vida útil y el alcance de los Access token generados por openid connect, que se muestran en la tabla 75 y las figuras 11, 12, 14, 20, 21, 33, 34, 35, 36, 47, 48.

b. En el presente estudio se ha implementado la solicitud de la actualización del Token mediante el uso de Refresh Token para gestionar la seguridad en las Apis expuestas por Swagger, logrando controlar el refresh token de un único uso y con caducidad en 10 minutos, como se muestra en las figuras 13, 16, 22.

c. En presente estudio se configuró el Redirect_uri para gestionar la seguridad en las Apis expuestas por Swagger, logrando implementar esta propiedad para que el openid connect solo redireccione a los clientes permitidos y autorizados, que se muestran en la tabla 76 y la figuras 17, 23 45.

CAPÍTULO VI

RECOMENDACIONES

6.1. RECOMENDACIONES

a. Implementar microservicios con las últimas versiones de OpenID Connect que incluyen características nuevas y mejoras de seguridad.

b. Utilizar el doble factor de autenticación en las aplicaciones nativas de la nube usando OpenId Connect, de esta manera se podrá garantizar la estabilidad y seguridad de las aplicaciones.

c. Utilizar Quarkus y Keycloak para implementar microservicios de manera rápida y sencilla para una autorización completa utilizando OIDC (OpenID Connect) y tener la máxima flexibilidad para configurar las políticas de acceso.

Referencias Bibliográficas

Alejandra y Jonathan. (2021). *CD 11490*.

Allen, B., & Baier, D. (2022). *IdentityServer4 Documentation Release 1.0.0*.

auth. (2023). Access Tokens. *Auth0.Com*.

Authorization. (2023, January). *Authorization*. Developers.

Avior.API. (2022). api. *HTTP Methods*.

AWS. (2023). ¿Qué es una API? *Https://Aws.Amazon.Com/*.

Bermeo, F., Silvano Hernández-Mosqueda, J., & Tobón-Tobón, S. (2016). *ANÁLISIS DOCUMENTAL DE LA V HEURÍSTICA MEDIANTE LA CARTOGRAFÍA CONCEPTUAL DOCUMENTARY ANALYSIS OF V HEURISTIC BY CONCEPTUAL CARTOGRAPHY*.

Casal, E. F., Mendoza, M., Victor, F., & Font, G. (2019). *Seguridad en APIs*.

Copete, J. (2019, February 16). *Introducción a OpenID Connect*. Arquitectoit.

Curity. (2023). *Actualización de OAuth*. Https://Curity.Io/Resources/Learn/Oauth-Refresh/.

Dan Arias y Sam Bellen. (2021). *What Are Refresh Tokens and How to Use Them Securely*.

Datatracker. (2020, January 31). *OAuth 2.0 Threat Model and Security Considerations*. Datatracke.IETF.

De la Fuente García, C. (2020, June 25). *Guía práctica para la publicación de Datos Abiertos usando APIs*. Datos.Gob.Es.

Duran Castañeda. (2020). *UNIVERSIDAD NACIONAL TECNOLÓGICA DE LIMA SUR FACULTAD DE INGENIERÍA Y GESTIÓN ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS "PROPUESTA PARA LA APLICACIÓN DEL FRAMEWORK OAUTH2 PARA LA SEGURIDAD DE UNA ARQUITECTURA BASADA EN TRABAJO DE SUFICIENCIA PROFESIONAL Para optar el Título Profesional de INGENIERO DE SISTEMAS PRESENTADO POR EL BACHILLER*.

Feria. (2018). *"SEGURIDAD PARA EL CONTROL DE ACCESO A RECURSOS DE LA APLICACIÓN WEB DE FACTURACIÓN ELECTRÓNICA OPENFACT, AHREN CONTRATISTAS GENERALES - AYACUCHO, 2017."* Universidad Nacional de San Cristobal de Huamanga.

Francisco Sánchez. (2018). *2014030364Tesinatokenpdf3805*.

García Delgado, Á., José Vázquez Delgado Ponente, J., & Varona Martínez, P. (2018). *UNIVERSIDAD AUTONOMA DE MADRID ESCUELA POLITECNICA SUPERIOR TRABAJO FIN DE GRADO Implementación de un proveedor de autorizaciones OAuth 2.0 con Scala*.

Guzmán, R., Director, S., & Guillermo, J. (2018). *Trabajo Fin de Máster presentado por*.

Hugo Pagola, I. (2018). *Implementación de API Gateway*.

IBM. (2023, July 2). *OpenID Connect*. WebSphere Application Server Liberty.

IONOS. (2020). *GET vs. POST: los dos métodos de petición HTTP*. Https://Www.Ionos.Es/Digitalguide/Paginas-Web/Desarrollo-Web/Get-vs-Post/.

Li, W., & Mitchell, C. J. (2020). User Access Privacy in OAuth 2.0 and OpenID Connect. *Proceedings - 5th IEEE European Symposium on Security and Privacy Workshops*,

Euro S and PW 2020, 664–672. https://doi.org/10.1109/EuroSPW51379.2020.00095

López, R., & Lucía, M. (2018). *Trabajo Fin de Máster Metodología para describir servicios RESTful consumidos automáticamente por clientes máquina*.

Marisela Dszul. (2021). Diseño No-Experimental. *Https://Www.Uaeh.Edu.Mx/Docencia/VI_Presentaciones/Licenciatura_en_mercadotecnia/Fundamentos_de_metodologia_investigacion/PRES38.Pdf*.

Microsoft. (2023, March 7). *Restricciones y limitaciones de URI de redireccionamiento (URL de respuesta)*. Https://Learn.Microsoft.Com/En-Us/Azure/Active-Directory/Develop/Reply-Url.

Noelia Martín. (2018, October 29). *OAuth 2.0, ¿qué es y cómo funciona?* Paradigmadigital.

Okta. (2023, February 14). *Token de acceso*. Https://Www.Okta.Com/Identity-101/Access-Token/.

OpenID. (2020). *Fundación OpenID* . Https://Openid.Net/Developers/How-Connect-Works/.

Oscar Blancarte. (2018, December 3). *Métodos HTTP (REST)*. Https://Www.Oscarblancarteblog.Com/2018/12/03/Metodos-Http-Rest/.

Pedro Luis López. (2014). POBLACIÓN MUESTRA Y MUESTREO. *Http://Www.Scielo.Org.Bo/Scielo.Php?Script=sci_arttext&pid=S1815-02762004000100012*.

Rafael Neto. (2023, February 28). *Flujos de autorización con OAuth 2.0 y OpenID Connect*. Https://Rafaelneto.Dev/Blog/Flujos-Autorizacion-Oauth-2-0-Openid-Connect/.

Rasiwasia, A. (2018). *A Framework To Implement OpenID Connect Protocol For Federated Identity Management In Enterprises*.

Rushdy, E., Khedr, W., & Salah, N. (2021). Framework to secure the OAuth 2.0 and JSON web token for rest API. *Journal of Theoretical and Applied Information Technology*, *99*(9), 2144–2161.

Singh, J., & Chaudhary, N. K. (2022). OAuth 2.0: Architectural design augmentation for mitigation of common security vulnerabilities. *Journal of Information Security and Applications*, *65*. https://doi.org/10.1016/j.jisa.2021.103091

Spasovski, Martin. (2019). *OAuth 2.0 identity and access management patterns : a practical hands-on guide to implementing secure API authorization flow scenariaos with OAuth 2.0*.

Sujanani, T., & Vinod, S. (2018). Implementation of OpenId connect and O Auth 2.0 to create SSO for educational institutes. *International Journal of Engineering and Technology(UAE)*, *7*(2), 153–157. https://doi.org/10.14419/ijet.v7i2.6.10142

Supo, F., Hugo, C., & Cavero, N. (2014). *UNIVERSIDAD NACIONAL DEL ALTIPLANO-PUNO UNIVERSIDAD ANDINA NÉSTOR CÁCERES VELÁSQUEZ-JULIACA FUNDAMENTOS TEÓRICOS Y PROCEDIMENTALES DE LA INVESTIGACIÓN CIENTÍFICA EN CIENCIAS SOCIALES COMO DISEÑAR Y FORMULAR TESIS DE MAESTRÍA Y DOCTORADO*.

SYDLE. (2022). Qué es API. *Transformación Digital*.

Teodoro, N., & Nieto, E. (2018). *TIPOS DE INVESTIGACIÓN*. http://repositorio.usdg.edu.pe/handle/USDG/34

Torres, G. (2019, April 14). *OAuth 2.0, OpenID Connect y JSON Web Tokens (JWT)*. Returngis.Net.

Urrutia Egaña, M., Barrios Araya, S., Marina Gutiérrez Núñez, L., & Magdalena Mayorga Camus, L. (2014). Métodos óptimos para determinar validez de contenido Optimal method for content validity. In *Educación Médica Superior* (Vol. 28, Issue 3). http://scielo.sld.cu

Williams Morales. (2023). Métodos HTTP – POST, GET, PUT, DELETE. *Http://Estilow3b.Com/Metodos-Http-Post-Get-Put-Delete/*.

Lista de Abreviaturas

HTTP: Hypertext transfer protocol, en español protocolo de transferencia de hipertexto.
JSON: JavaScript object notation, en español notación de objeto de JavaScript.
REST: Representational state transfer, en español transferencia de estado representacional.
SOAP: Simple object access protocol, en español protocolo simple de acceso a objetos.
URI: Uniform resource identifier, en español identificador de recursos uniforme.
XML: Extensible markup language, en español lenguaje de marcado extensible.
OIDC: OpenID Connect (protocolo de autenticación y autorización).
POIDC: Protocolo OpenID Connect
API: Application Programming Interface, en español Interfaz de Programación de Aplicaciones.
DOS: Denegación de servicio
OP: Proveedor de OpenID Connect
UX: User Experience, en español experiencia del usuario
IDE: entorno de desarrollo integrado

Glosario

Payload: Son los datos que se envían junto con la solicitud en el cuerpo del mensaje.

Scope: Es un parámetro que se incluye en una solicitud de autenticación para especificar los permisos y el alcance de acceso que una aplicación.

Idempotente: Es una función en la que realizar la misma acción repetidamente.

Update: Es el proceso que implica la modificación o actualización de datos.

Anexos

Anexo 1. Matriz de consistencia

Título: Protocolo Openid Connect para gestionar la seguridad en las Apis expuestas por Swagger, 2024.

PROBLEMAS	OBJETIVOS	VARIABLES	MÉTODO DE INVESTIGACIÓN
PROBLEMA GENERAL ¿Cómo el protocolo Openid Connect gestiona la seguridad en las Apis expuestas por Swagger, 2024? **PROBLEMAS ESPECÍFICOS** a. ¿De qué manera usar el Access Token para gestionar la seguridad en las Apis expuestas por Swagger? b. ¿Cómo generar el Refresh Token para gestionar la seguridad en las Apis expuestas por Swagger? c. ¿Cómo el Redirect_uri gestiona la seguridad en las Apis expuestas por Swagger	**OBJETIVO GENERAL** Implementar el protocolo Openid Connect para gestionar la seguridad en las Apis expuestas por Swagger, 2024. **OBJETIVOS ESPECÍFICOS** a. Definir el alcance de la autorización del Access Token para gestionar la seguridad en las Apis expuestas por Swagger. b. Configuración del Refresh Token para gestionar la seguridad en las Apis expuestas por Swagger. c. Determinar el alcance de Redirect_uri para gestionar la seguridad en las Apis expuestas por Swagger.	**Primera Variable Interés** X: Protocolo Openid Connect **Variables Descriptivas** X1: Access Token X2: Refresh Token X3: Redirect_uri **Segunda Variable Interés** Y: Apis **Variables Descriptivas** Y1: Get Y2: Post Y3: Put Y4: Delete	**TIPO DE INVESTIGACIÓN** Aplicada **NIVEL DE INVESTIGACIÓN** Descriptivo **DISEÑO** No experimental **POBLACIÓN** La población está conformada por 10 Microservicios de la DIRESA - Ayacucho 2024. **MUESTRA** La muestra es mediante un muestro no probabilístico por conveniencia,

			se tomó un Microservicio de la DIRESA-Ayacucho 2024. **TÉCNICA** Análisis documental **INSTRUMENTO** Registro de datos

Anexo 2. Instrumento registro Documental para la recolección de datos.

REGISTRO DOCUMENTAL DE INGRESO A LAS EN LOS MICROSERVICIOS DE LA DIRECCIÓN REGIONAL DE SALUD DE AYACUCHO.

Universidad Nacional de San Cristóbal de Huamanga

REGISTRO DOCUMENTAL				
FECHA:				
FUENTE:				
OBJETIVO DE LA FICHA:				
DESCRIPCIÓN:	**CONCLUSIONES**			
¿El access token es utilizado como clave para acceder a los recursos protegidos?	Acceso seguro ☐	Autenticación eficiente ☐	Autorización eficiente ☐	Tiempo de duración es limitada ☐
¿Se requiere enviar un Access token válido para obtener acceso a recursos específicos?	APIs restringidas ☐	Servicios Web ☐	Microservicios ☐	Recursos confidenciales ☐
¿El refresh token de qué manera es utilizado para renovar el access token una vez haya expirado?	Renovación segura ☐	Actualización automática ☐	Extensión acceso ☐	Continuidad autorización ☐
¿El proceso de actualización del token garantiza la identidad y seguridad del usuario de manera efectiva?	Refuerza confianza ☐	Mantiene protección ☐	Mejora control ☐	Asegura acceso ☐
¿Permite el redirect_uri a las aplicaciones	Control autenticación	Permisos adaptados	Acceso dirigido	Alcances personalizados

especificar los permisos y alcances necesarios durante la autenticación?	☐	☐	☐	☐
¿El redirect_uri asegura que solo se activen los permisos solicitados durante la autenticación?	Control redireccionamiento ☐	Permisos validados ☐	Flujo autorización ☐	Accesos específicos ☐
¿Las solicitudes GET son utilizadas para recuperar información o recursos específicos?	Recuperación datos ☐	Obtención recursos ☐	Información específica ☐	Datos solicitados ☐
¿Existe alguna convención o estándar específico que se siga al crear recursos de tipo GET para asegurar su coherencia y claridad?	Uso convenciones (practicas o patrones) ☐	Estándares y normas seguidos ☐	Prácticas recomendadas ☐	Conformidad normas y directrices ☐
¿Las solicitudes POST se envían en el cuerpo de la petición para crear recursos?	Envío cuerpo ☐	Datos incluidos ☐	Cuerpo petición ☐	Contenido POST ☐
¿Las respuestas de las solicitudes POST contienen datos precisos y relevantes?	Datos específicos ☐	Contenido relevante ☐	Respuestas detalladas ☐	Información puntual ☐
¿Las solicitudes PUT son empleadas para actualizar los datos de recursos existentes de manera segura y eficiente?	Actualización recursos ☐	Modificación datos ☐	Cambios existentes ☐	Datos actualizados ☐
¿Las solicitudes PUT siguen un proceso	Proceso seguro	Seguridad	Garantía	Cumple

seguro para la actualización de datos?	☐	aplicada ☐	protección ☐	estándares ☐
¿Las solicitudes DELETE son utilizadas para eliminar recursos de manera segura?	Eliminación segura ☐	Borrado confiable ☐	Proceso protegido ☐	Datos removidos ☐
¿Se implementa algún tipo de confirmación o medida de seguridad antes de proceder con la eliminación para evitar acciones no deseadas?	Confirmación previa ☐	Medidas seguridad ☐	Protección eliminación ☐	Proceso seguro ☐

Nota: Basándonos en la evaluación de la validez de contenido, se ha revisado y reformulado las interrogantes 2, 4 y 13 con el propósito de mejorar la claridad de cada pregunta.

Anexo 3. Certificado de validez de contenido del instrumento.

Señor juez se solicita su colaboración para para evaluar la validez de contenido de instrumento registro. Se agradece por su colaboración.

FICHA DE ANALISIS DOCUMENTAL

Nº	DIMENSIONES / ítems	Relevancia[1]		Represen_tatividad[2]		Claridad[3]		Sugerencias / Observaciones
		Sí	No	Sí	No	Sí	No	
Dimensión: Access Token								
1	¿El access token es utilizado como clave para acceder a los recursos protegidos?	X		X		X		
2	¿Se requiere enviar un Access token válido para obtener acceso a recursos específicos?	X		X		X		
Dimensión: Refresh Token								
3	¿El refresh token de qué manera es utilizado para renovar el access token una vez haya expirado?	X		X		X		
4	¿El proceso de actualización del token garantiza la identidad y seguridad del usuario de manera efectiva?	X		X		X		
Dimensión: Redirect_uri								
5	¿Permite el redirect_uri a las	X		X		X		

	aplicaciones especificar los permisos y alcances necesarios durante la autenticación?							
6	¿El redirect_uri asegura que solo se activen los permisos solicitados durante la autenticación?	X		X		X		
Dimensión: Get								
7	¿Las solicitudes GET son utilizadas para recuperar información o recursos específicos?	X		X		X		
8	¿Existe alguna convención o estándar específico que se siga al crear recursos de tipo GET para asegurar su coherencia y claridad?	X		X		X		
Dimensión: Post								
9	¿Las solicitudes POST se envían en el cuerpo de la petición para crear recursos?	X		X		X		
10	¿Las respuestas de las solicitudes POST contienen datos precisos y relevantes?	X		X		X		
Dimensión: Put								
11	¿Las solicitudes PUT son empleadas para	X		X		X		

	actualizar los datos de recursos existentes de manera segura y eficiente?							
12	¿Las solicitudes PUT siguen un proceso seguro para la actualización de datos?	X		X		X		
Dimensión: Delete								
13	¿Las solicitudes DELETE son utilizadas para eliminar recursos de manera segura?	X		X		X		
14	¿Se implementa algún tipo de confirmación o medida de seguridad antes de proceder con la eliminación para evitar acciones no deseadas?	X		X		X		

Observaciones (precisar si hay observaciones): **NINGUNA**

Opinión de aplicabilidad: Aplicable [**X**] Aplicable después de corregir [ ] No aplicable [ ]
Apellidos y nombres del juez validador Dr. / Mg / Ing: **TORRES SIMBRON LUIS ALBERTO**
DNI o CE: **72848768**
Especialidad del validador: **DESARROLLADOR FULL STACK**

……
Firma

[1]**Relevancia:** El ítem corresponde al concepto teórico formulado.

[2]**Representatividad:** El ítem es apropiado para representar al componente o dimensión específica del constructo.

[3]**Claridad:** Se entiende sin dificultad alguna el enunciado del ítem, es conciso, exacto y directo.

Nota: Suficiencia, se dice suficiencia cuando los ítems planteados son suficientes para medir la dimensión.

16 de mayo de 2024

FICHA DE ANALISIS DOCUMENTAL

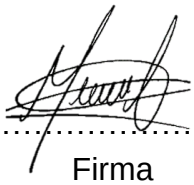
Nº	DIMENSIONES / ítems	Relevancia[1]		Represen_tatividad[2]		Claridad[3]		Sugerencias / Observaciones
		Sí	No	Sí	No	Sí	No	
Dimensión: Access Token								
1	¿El access token es utilizado como clave para acceder a los recursos protegidos?	x		x		x		
2	¿Se requiere enviar un Access token válido para obtener acceso a recursos específicos?	x		x			x	
Dimensión: Refresh Token								
3	¿El refresh token de qué manera es utilizado para renovar el access token una vez haya expirado?	x		x		x		
4	¿El proceso de actualización del token garantiza la identidad y seguridad del usuario de manera efectiva?	x		x		x		
Dimensión: Redirect_uri								
5	¿Permite el redirect_uri a las aplicaciones especificar los permisos y alcances necesarios durante la autenticación?	x		x		x		

6	¿El redirect_uri asegura que solo se activen los permisos solicitados durante la autenticación?	x		x		x		
Dimensión: Get								
7	¿Las solicitudes GET son utilizadas para recuperar información o recursos específicos?	x		x		x		
8	¿Existe alguna convención o estándar específico que se siga al crear recursos de tipo GET para asegurar su coherencia y claridad?	x		x		x		
Dimensión: Post								
9	¿Las solicitudes POST se envían en el cuerpo de la petición para crear recursos?	x		x		x		
10	¿Las respuestas de las solicitudes POST contienen datos precisos y relevantes?	x		x		x		
Dimensión: Put								
11	¿Las solicitudes PUT son empleadas para actualizar los datos de recursos existentes de manera segura y eficiente?	x		x		x		

12	¿Las solicitudes PUT siguen un proceso seguro para la actualización de datos?	x		x		x		
Dimensión: Delete								
13	¿Las solicitudes DELETE son utilizadas para eliminar recursos de manera segura?	x		x		x		
14	¿Se implementa algún tipo de confirmación o medida de seguridad antes de proceder con la eliminación para evitar acciones no deseadas?	x		x		x		

Observaciones (precisar si hay observaciones): **NINGUNA**

Opinión de aplicabilidad: Aplicable [**X**] Aplicable después de corregir [ ] No aplicable [ ]
Apellidos y nombres del juez validador Dr. / Mg / Ing: **VILCHEZ IMAN MARTIN SAUL**
DNI o CE: **47941030**
Especialidad del validador: **BACK-END .NET**



……
Firma

[1]**Relevancia:** El ítem corresponde al concepto teórico formulado.

[2]**Representatividad:** El ítem es apropiado para representar al componente o dimensión específica del constructo.

[3]**Claridad:** Se entiende sin dificultad alguna el enunciado del ítem, es conciso, exacto y directo.

Nota: Suficiencia, se dice suficiencia cuando los ítems planteados son suficientes para medir la dimensión.

16 de mayo de 2024

FICHA DE ANALISIS DOCUMENTAL

Nº	DIMENSIONES / ítems	Relevancia[1]		Represen_tatividad[2]		Claridad[3]		Sugerencias / Observaciones
		Sí	No	Sí	No	Sí	No	
Dimensión: Access Token								
1	¿El access token es utilizado como clave para acceder a los recursos protegidos?	x		x		x		
2	¿Se requiere enviar un Access token válido para obtener acceso a recursos específicos?	x		x		x		
Dimensión: Refresh Token								
3	¿El refresh token de qué manera es utilizado para renovar el access token una vez haya expirado?	x		x		x		
4	¿El proceso de actualización del token garantiza la identidad y seguridad del usuario de manera efectiva?	x		x		x		
Dimensión: Redirect_uri								
5	¿Permite el redirect_uri a las aplicaciones especificar los permisos y alcances necesarios durante la autenticación?	x		x		x		

6	¿El redirect_uri asegura que solo se activen los permisos solicitados durante la autenticación?	x		x		x		
Dimensión: Get								
7	¿Las solicitudes GET son utilizadas para recuperar información o recursos específicos?	x		x		x		
8	¿Existe alguna convención o estándar específico que se siga al crear recursos de tipo GET para asegurar su coherencia y claridad?	x		x		x		
Dimensión: Post								
9	¿Las solicitudes POST se envían en el cuerpo de la petición para crear recursos?	x		x		x		
10	¿Las respuestas de las solicitudes POST contienen datos precisos y relevantes?	x		x		x		
Dimensión: Put								
11	¿Las solicitudes PUT son empleadas para actualizar los datos de recursos existentes de manera segura y eficiente?	x		x		x		

12	¿Las solicitudes PUT siguen un proceso seguro para la actualización de datos?	x		x		x		
Dimensión: Delete								
13	¿Las solicitudes DELETE son utilizadas para eliminar recursos de manera segura?	x		x			x	
14	¿Se implementa algún tipo de confirmación o medida de seguridad antes de proceder con la eliminación para evitar acciones no deseadas?	x		x		x		

Observaciones (precisar si hay observaciones): **NINGUNA**

Opinión de aplicabilidad: Aplicable [**X**] Aplicable después de corregir [ ] No aplicable [ ]
Apellidos y nombres del juez validador Dr. / Mg / Ing: **PALOMINO PARIONA ALEX**
DNI o CE: **45602561**
Especialidad del validador: **Arquitecto de Software**

……

Firma

[1]**Relevancia:** El ítem corresponde al concepto teórico formulado.

[2]**Representatividad:** El ítem es apropiado para representar al componente o dimensión específica del constructo.

[3]**Claridad:** Se entiende sin dificultad alguna el enunciado del ítem, es conciso, exacto y directo.

Nota: Suficiencia, se dice suficiencia cuando los ítems planteados son suficientes para medir la dimensión.

16 de mayo de 2024

FICHA DE ANALISIS DOCUMENTAL

Nº	DIMENSIONES / ítems	Relevancia[1]		Represen_tatividad[2]		Claridad[3]		Sugerencias / Observaciones
		Sí	No	Sí	No	Sí	No	
Dimensión: Access Token								
1	¿El access token es utilizado como clave para acceder a los recursos protegidos?	x		x		x		
2	¿Se requiere enviar un Access token válido para obtener acceso a recursos específicos?	x		x		x		
Dimensión: Refresh Token								
3	¿El refresh token de qué manera es utilizado para renovar el access token una vez haya expirado?	x		x		x		
4	¿El proceso de actualización del token garantiza la identidad y seguridad del usuario de manera efectiva?	x		x			x	
Dimensión: Redirect_uri								
5	¿Permite el redirect_uri a las aplicaciones especificar los permisos y alcances necesarios durante la autenticación?	x		x		x		

6	¿El redirect_uri asegura que solo se activen los permisos solicitados durante la autenticación?	x		x		x		
Dimensión: Get								
7	¿Las solicitudes GET son utilizadas para recuperar información o recursos específicos?	x		x		x		
8	¿Existe alguna convención o estándar específico que se siga al crear recursos de tipo GET para asegurar su coherencia y claridad?	x		x		x		
Dimensión: Post								
9	¿Las solicitudes POST se envían en el cuerpo de la petición para crear recursos?	x		x		x		
10	¿Las respuestas de las solicitudes POST contienen datos precisos y relevantes?	x		x		x		
Dimensión: Put								
11	¿Las solicitudes PUT son empleadas para actualizar los datos de recursos existentes de manera segura y eficiente?	x		x		x		

12	¿Las solicitudes PUT siguen un proceso seguro para la actualización de datos?	x		x		x		
Dimensión: Delete								
13	¿Las solicitudes DELETE son utilizadas para eliminar recursos de manera segura?	x		x		x		
14	¿Se implementa algún tipo de confirmación o medida de seguridad antes de proceder con la eliminación para evitar acciones no deseadas?	x		x		x		

Observaciones (precisar si hay observaciones): **NINGUNA**

Opinión de aplicabilidad: Aplicable [**X**] Aplicable después de corregir [ ] No aplicable [ ]
Apellidos y nombres del juez validador Dr. / Mg / Ing: **SILVA MEZA JOSÉ PEDRO**
DNI o CE: **42694350**
Especialidad del validador: **Arq. Software .Net, Adm. Base de datos**

[1]**Relevancia:** El ítem corresponde al concepto teórico formulado.

[2]**Representatividad:** El ítem es apropiado para representar al componente o dimensión específica del constructo.

[3]**Claridad:** Se entiende sin dificultad alguna el enunciado del ítem, es conciso, exacto y directo.

Nota: Suficiencia, se dice suficiencia cuando los ítems planteados son suficientes para medir la dimensión.

16 de mayo de 2024

FICHA DE ANALISIS DOCUMENTAL

Nº	DIMENSIONES / ítems	Relevancia[1]		Represen_tatividad[2]		Claridad[3]		Sugerencias / Observaciones
		Sí	No	Sí	No	Sí	No	
Dimensión: Access Token								
1	¿El access token es utilizado como clave para acceder a los recursos protegidos?	x		x		x		
2	¿Se requiere enviar un Access token válido para obtener acceso a recursos específicos?	x		x		x		
Dimensión: Refresh Token								
3	¿El refresh token de qué manera es utilizado para renovar el access token una vez haya expirado?	x		x		x		
4	¿El proceso de actualización del token garantiza la identidad y seguridad del usuario de manera efectiva?	x		x		x		
Dimensión: Redirect_uri								
5	¿Permite el redirect_uri a las aplicaciones especificar los permisos y alcances necesarios durante la autenticación?	x		x		x		

6	¿El redirect_uri asegura que solo se activen los permisos solicitados durante la autenticación?	x		x		x		
Dimensión: Get								
7	¿Las solicitudes GET son utilizadas para recuperar información o recursos específicos?	x		x		x		
8	¿Existe alguna convención o estándar específico que se siga al crear recursos de tipo GET para asegurar su coherencia y claridad?	x		x		x		
Dimensión: Post								
9	¿Las solicitudes POST se envían en el cuerpo de la petición para crear recursos?	x		x		x		
10	¿Las respuestas de las solicitudes POST contienen datos precisos y relevantes?	x		x		x		
Dimensión: Put								
11	¿Las solicitudes PUT son empleadas para actualizar los datos de recursos existentes de manera segura y eficiente?	x		x		x		

12	¿Las solicitudes PUT siguen un proceso seguro para la actualización de datos?	x		x		x		
Dimensión: Delete								
13	¿Las solicitudes DELETE son utilizadas para eliminar recursos de manera segura?	x		x		x		
14	¿Se implementa algún tipo de confirmación o medida de seguridad antes de proceder con la eliminación para evitar acciones no deseadas?	x		x		x		

Observaciones (precisar si hay observaciones): **NINGUNA**

Opinión de aplicabilidad: Aplicable [**X**] Aplicable después de corregir [ ] No aplicable [ ]

Apellidos y nombres del juez validador Dr. / Mg / Ing: **CORONEL ARAUJO YOEL KERELI**

DNI o CE: **72024497**

Especialidad del validador: **DevOps**

……

Firma

[1]**Relevancia:** El ítem corresponde al concepto teórico formulado.

[2]**Representatividad:** El ítem es apropiado para representar al componente o dimensión específica del constructo.

[3]**Claridad:** Se entiende sin dificultad alguna el enunciado del ítem, es conciso, exacto y directo.

Nota: Suficiencia, se dice suficiencia cuando los ítems planteados son suficientes para medir la dimensión.

16 de mayo de 2024

Anexo 4. Cálculo de la V de Aiken con intervalos de confianza.

Inserte valores		
mín	0	
máx	1	
k	1	
n	5	
sig	1.96	95%

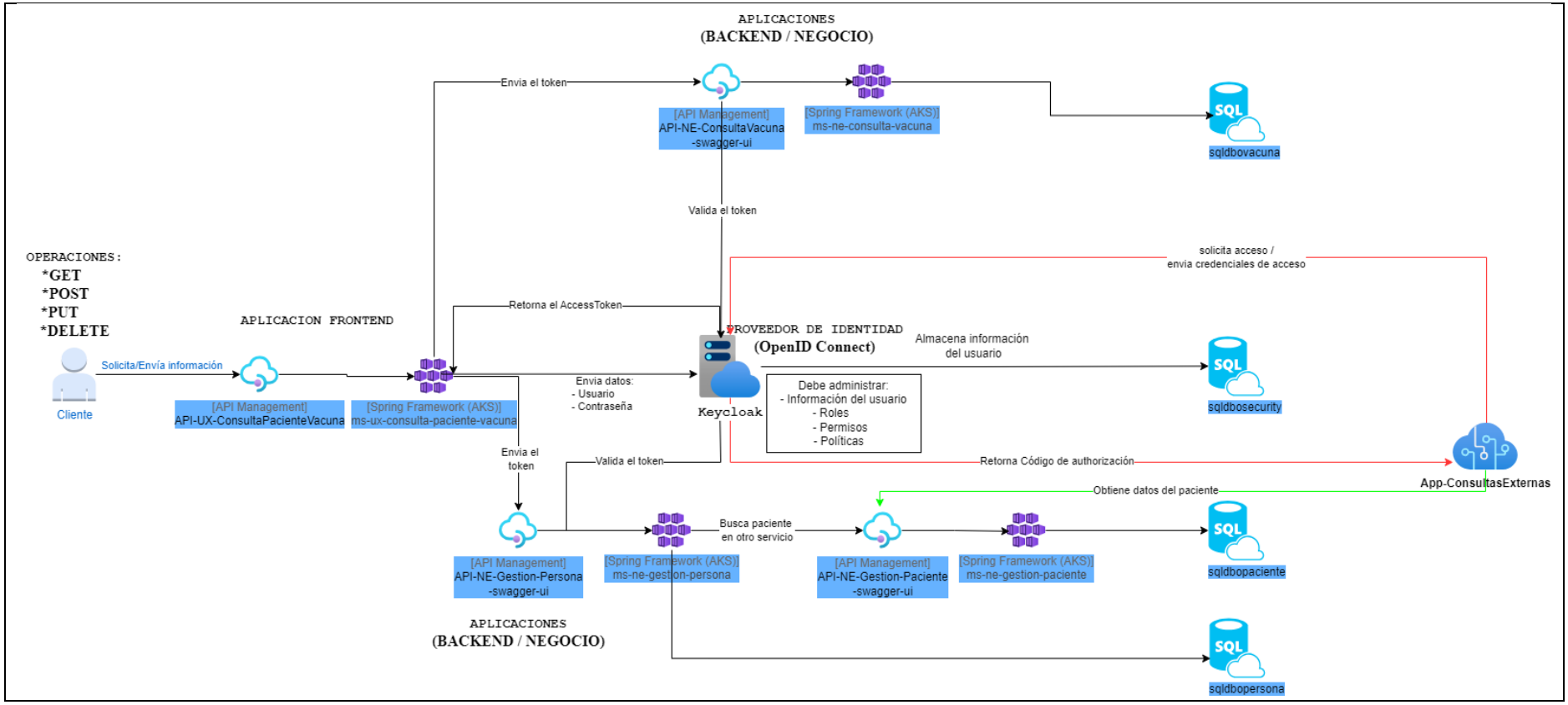
											Intervalo de Confianza	
		Juez 1	Juez 2	Juez 3	Juez 4	Juez 5	Media	DE	V de Aiken	Interpretación V	Inferior	Superior
¿El access token es utilizado como clave para acceder a los recursos protegidos?	**Relevancia**	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Representatividad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Claridad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
¿Se requiere enviar un Access token válido para obtener acceso a recursos específicos?	**Relevancia**	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Representatividad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Claridad	1	0	1	1	1	0.80	0.40	0.80	VALIDO	0.38	0.96
¿El refresh token de qué manera es utilizado para renovar el access token una vez haya expirado?	**Relevancia**	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Representatividad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Claridad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
¿El proceso de actualización del token garantiza la identidad y seguridad del usuario de manera efectiva?	**Relevancia**	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Representatividad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Claridad	1	1	1	0	1	0.80	0.40	0.80	VALIDO	0.38	0.96
¿Permite el redirect_uri a las aplicaciones especificar los	**Relevancia**	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Representatividad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00

permisos y alcances necesarios durante la autenticación?	**Claridad**	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
¿El redirect_uri asegura que solo se activen los permisos solicitados durante la autenticación?	**Relevancia**	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Representatividad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Claridad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
¿Las solicitudes GET son utilizadas para recuperar información o recursos específicos?	**Relevancia**	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Representatividad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Claridad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
¿Existe alguna convención o estándar específico que se siga al crear recursos de tipo GET para asegurar su coherencia y claridad?	**Relevancia**	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Representatividad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Claridad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
¿Las solicitudes POST se envían en el cuerpo de la petición para crear recursos?	**Relevancia**	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Representatividad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Claridad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
¿Las respuestas de las solicitudes POST contienen datos precisos y relevantes?	**Relevancia**	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Representatividad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Claridad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
¿Las solicitudes PUT son empleadas para actualizar los datos de recursos existentes de manera segura y eficiente?	**Relevancia**	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Representatividad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Claridad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
¿Las solicitudes PUT siguen un proceso seguro para la actualización de datos?	**Relevancia**	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Representatividad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Claridad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00

¿Las solicitudes DELETE son utilizadas para eliminar recursos de manera segura?	**Relevancia**	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Representatividad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Claridad	1	1	0	1	1	0.80	0.40	0.80	VALIDO	0.38	0.96
¿Se implementa algún tipo de confirmación o medida de seguridad antes de proceder con la eliminación para evitar acciones no deseadas?	**Relevancia**	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Representatividad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00
	Claridad	1	1	1	1	1	1.00	0.00	1.00	VALIDO	0.57	1.00

Nota: El coeficiente V de Aiken es de 0.95, indicando que el instrumento posee una excelente validez de contenido. Cuanto más se aproxime a uno, mayor será la validez de contenido del instrumento.

Anexo 5. Flujo de implementación



UNSCH FACULTAD DE INGENIERÍA DE MINAS, GEOLOGIA Y CIVIL

ACTA DE SUSTENTACION DE TESIS N° 056-2024-FIMGC

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERA DE SISTEMAS

En la Universidad Nacional de San Cristóbal de Huamanga de la ciudad de Ayacucho, en cumplimiento a la **Resolución Decanal N° 586-2024-FIMGC-D,** a los cinco días del mes de setiembre de 2024, siendo las 10:00 a.m, reunidos en el Auditorio de la Escuela Profesional de Ingeniería de Minas, bajo la presidencia del MSc. Ing. José Ernesto ESTRADA CÁRDENAS y los miembros; Mg. Ingª. Edith Felicitas GUEVARA MOROTE, Mg. Ing. Juan Carlos CARREÑO GAMARRA y Mg. Ingª. Elinar CARRILLO RIVEROS actuando como secretario docente el MSc. Ing. Kelvis BERROCAL ARGUMEDO, para proceder a la sustentación de tesis para optar el Título Profesional de Ingeniera de Sistemas, del bachiller:

Yovana RONDINEL PEREZ

Quien presentó la tesis denominada:

Protocolo OpenID Connect para gestionar la seguridad en las APIs expuestas por Swagger, 2024.

Los señores miembros del jurado luego de expuesto la tesis y absueltas las preguntas, delibera y lo declaran:

APROBADO CON NOTA DIECISÉIS (16)

Siendo las 12:10 p.m. del día 05 de setiembre de 2024, culmina el acto de sustentación de tesis, y en conformidad a lo actuado los miembros del jurado firmamos al pie del presente.

MSc. Ing. José Ernesto ESTRADA CÁRDENAS
Presidente

Mg. Ingª. Edith Felicitas GUEVARA MOROTE
Miembro

Mg. Ing. Juan Carlos CARREÑO GAMARRA
Miembro

Mg. Ingª. Elinar CARRILLO RIVEROS
Miembro - Asesora

MSc. Kelvis BERROCAL ARGUMEDO
Secretario docente de la FIMGC

cc:
Archivo

FACULTAD DE INGENIERÍA
DE MINAS, GEOLOGÍA Y CIVIL
Av. Independencia S/N
Ciudad Universitaria
Central Tel 066 312510
Anexo 151

CONSTANCIA DE ORIGINALIDAD DE TRABAJO DE INVESTIGACIÓN

CONSTANCIA Nº 011-2024-KPS-FIMGC/UNSCH

El que suscribe; responsable verificador de originalidad de trabajos de tesis de pregrado con el software Turnitin, en segunda instancia para las **Escuelas Profesionales** de la **Facultad de Ingeniería de Minas, Geología y Civil**; en cumplimiento a la **Resolución de Consejo Universitario Nº 039-2021-UNSCH-CU**, Reglamento de Originalidad de Trabajos de Investigación de la Universidad Nacional San Cristóbal de Huamanga y **Resolución Decanal Nº 473-2023-FIMGC-D**, deja constancia de originalidad de trabajo de investigación, que el/la Sr./Srta.

Nombres y Apellidos	:	Yovana RONDINEL PEREZ
Escuela Profesional	:	INGENIERÍA DE SISTEMAS
Título de la Tesis	:	Protocolo OpenID Connect para gestionar la seguridad en las APIs expuestas por Swagger, 2024.
Evaluación de la Originalidad	:	**13%** Índice de Similitud
Identificador de la entrega	:	2474900998

Por tanto, según los Artículos 12, 13 y 17 del Reglamento de Originalidad de Trabajos de Investigación, es **PROCEDENTE** otorgar la **Constancia de Originalidad** para los fines que crea conveniente.

En señal de conformidad y verificación se firma la presente constancia

Ayacucho, 09 de octubre de 2024

FIRMA DIGITAL

Firmado digitalmente por: PERALTA SOTOMAYOR Karel FAU 20143660754 soft
Motivo: Soy el autor del documento
Fecha: 09/10/2024 22:10:00-0500

Con depósito para Sustentación y Tramites
C.c. Archivo

FACULTAD DE INGENIERIA DE MINAS, GEOLOGIA Y CIVIL
Av. Independencia S/N Ciudad Universitaria
Central Tel. 066 312510
Anexo 151

Protocolo OpenID Connect para gestionar la seguridad en las APIs expuestas por Swagger, 2024.

por Yovana RONDINEL PEREZ

Fecha de entrega: 04-oct-2024 09:28a.m. (UTC-0500)
Identificador de la entrega: 2474900998
Nombre del archivo: Yovana_Rondinel_Pérez.pdf (5.02M)
Total de palabras: 23412
Total de caracteres: 128493

Protocolo OpenID Connect para gestionar la seguridad en las APIs expuestas por Swagger, 2024.

INFORME DE ORIGINALIDAD

13%	11%	3%	10%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	Submitted to Universidad Nacional de San Cristóbal de Huamanga Trabajo del estudiante	7%
2	hdl.handle.net Fuente de Internet	2%
3	git.laquadrature.net Fuente de Internet	1%
4	Submitted to Universidad Pontificia de Salamanca Trabajo del estudiante	<1%
5	fdocuments.es Fuente de Internet	<1%
6	www.slideshare.net Fuente de Internet	<1%
7	estilow3b.com Fuente de Internet	<1%
8	repositorio.upse.edu.ec Fuente de Internet	<1%

9	repositorio.unfv.edu.pe Fuente de Internet	<1 %
10	dev.to Fuente de Internet	<1 %
11	Submitted to Universidad TecMilenio Trabajo del estudiante	<1 %
12	docs.redhat.com Fuente de Internet	<1 %
13	link.springer.com Fuente de Internet	<1 %
14	repositorio.unsch.edu.pe Fuente de Internet	<1 %
15	repositorio.untels.edu.pe Fuente de Internet	<1 %
16	Submitted to University of Northampton Trabajo del estudiante	<1 %
17	doczz.es Fuente de Internet	<1 %
18	cybertesis.uni.edu.pe Fuente de Internet	<1 %
19	repositorio.ucv.edu.pe Fuente de Internet	<1 %

Excluir citas	Activo	Excluir coincidencias	< 30 words
Excluir bibliografía	Activo		