Universidad de Buenos Aires

Facultades de Ciencias Económicas, Cs. Exactas y Naturales e Ingeniería

Carrera de Especialización en Seguridad Informática

Trabajo Final

Implementación de API Gateway

Autor: Ing. Hugo Pagola.

Tutor: Mg Ing Alberto Dams

Por medio de la presente, el autor manifiesta conocer y aceptar el Reglamento de Trabajos Finales vigente y se hace responsable que la totalidad de los contenidos del presente documento son originales y de su Creación exclusiva, o bien pertenecen a terceros u otras fuentes, que han sido adecuadamente referenciados y cuya inclusión no infringe la legislación Nacional e Internacional de Propiedad Intelectual

---

Hugo Alberto Pagola

DNI: 20892783

Resumen

El presente informe es una bitácora de la implementación de un Sistema de API Gateway en una importante organización. Básicamente se encaró la problemática de autorización de acceso a las APIS. Para ello se implementó un servicio de oauth y OpenIDConnect para autenticar los usuarios y autorizar sus accesos.

La implementación se realizó en un entorno hibrido con la nube cloud de Azure. Es decir, dando servicios principalmente desde equipos virtualizados en la nube de Azure y en forma secundaria desde el centro de cómputos de la organización.

En el presente trabajo primero se da una introducción o marco teórico, en la segunda etapa se describen los requerimientos no funcionales y la implementación del sistema. Por último, se detalla una prueba de concepto de cliente que usa el servicio implementado.

Palabras Claves

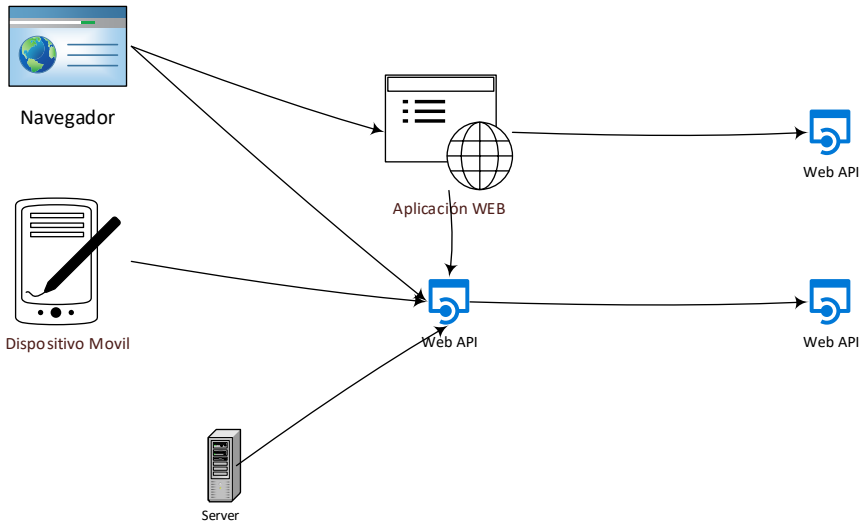
Autenticación, Autorización, Oauth, openidConnect, API.

Contenido

1 Introducción

En el presente trabajo se analizará la casuística de aplicaciones que deben acceder a los datos por medio de interfaces API tipo WEB.

La mayoría de las aplicaciones actualmente siguen un esquema como el siguiente[1]:



Generalmente el usuario interactúa con un navegador o con un dispositivo, los navegadores se comunican con aplicaciones web. Y el acceso a las APIS se produce de la siguiente manera:

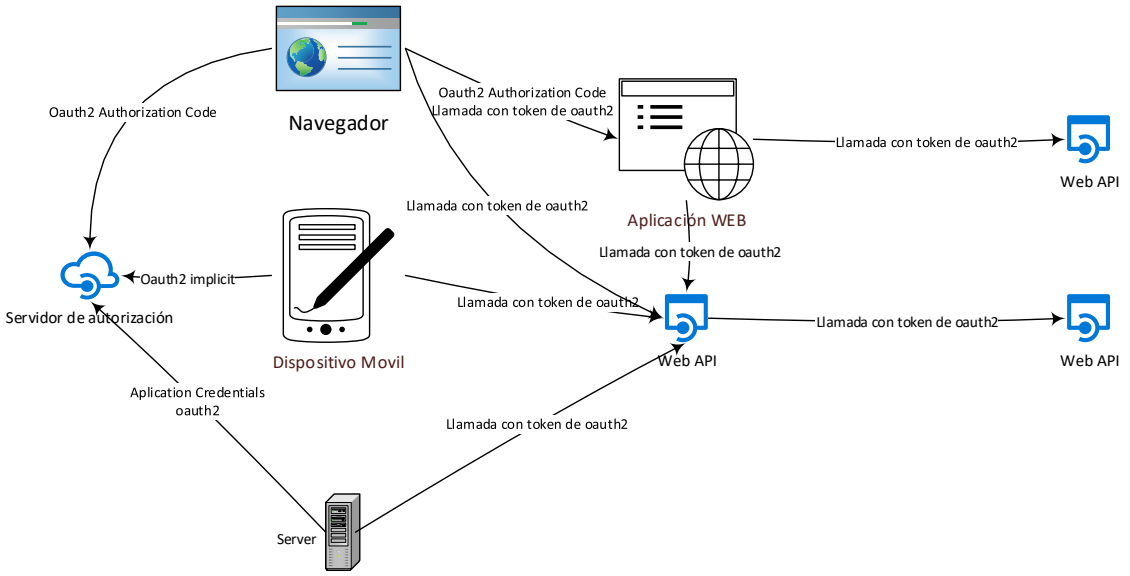
- Las aplicaciones web se comunican con las API web (a veces por temas propios de la aplicación y a veces en nombre de un usuario)
- Las aplicaciones basadas en navegador se comunican con API web
- Las aplicaciones nativas se comunican con las API web
- Las aplicaciones basadas en servidor se comunican con API web
- Las API web se comunican con otras API web (a veces por su cuenta, a veces en nombre de un usuario)

Por lo general, las capas de una aplicación deben proteger los recursos autenticando y/o autorizando a los usuarios habitualmente en una base de datos de cuentas de usuario.

Si pensamos la seguridad como un servicio corporativo, es importante que las aplicaciones no reinventen la rueda y utilicen marcos de trabajo estandarizados como los que analizaremos en el presente trabajo. Para ello se propone utilizar el protocolo oauth para trabajar con una autoridad que emita tokens que se

utilizan posteriormente para acceder a los servicios. Este token de seguridad puede ser al portador o requerir algún tipo de doble factor.

Suponiendo una autoridad que entrega tokens de seguridad el esquema anterior se reduce a una arquitectura de seguridad como la siguiente:



De esta forma se divide la problemática en dos pasos: En el primer paso, se autentica al usuario y se obtiene el token de seguridad; En el segundo se obtiene el acceso a los distintos servicios utilizando el token de seguridad. Dicho token deberá ser validado por el servicio.

Autenticación

La autenticación es necesaria cuando la aplicación necesita conocer la identidad del usuario actual. Habitualmente, las aplicaciones deben administrar datos en nombre del usuario y deben asegurarse de que este solo pueda acceder a los datos que tiene permitido.

Los protocolos de autenticación más comunes son SAML2, WS-Federation y actualmente se está imponiendo el marco OpenID Connect/oauth2. Dicho marco fue creado teniendo en consideración escenarios de aplicaciones móviles desde el principio y está pensado específicamente para ser compatible con infraestructuras que manejan APIs.

Acceso a datos y funcionalidad por medio de APIs

Las aplicaciones tienen dos formas fundamentales de trabajar con APIs: La primera es utilizar la identidad de la aplicación, la segunda es utilizar en forma delegada la identidad del usuario.

El protocolo OAuth2 permite solicitar tokens de acceso a un servicio de tokens de seguridad y usarlos para comunicarse con las API. Esto reduce la complejidad tanto en las aplicaciones cliente como en las API, ya que la autenticación y la autorización se pueden centralizar.

OpenID Connect es una extensión de OAuth2 que incorpora el manejo de la Identidad en forma federada, es decir el acceso a los datos básicos de los usuarios.

La combinación de OpenID Connect y OAuth2 es un buen enfoque para asegurar aplicaciones modernas y resolver los problemas de seguridad típicos de las aplicaciones móviles, nativas y web actuales.

2 Marco Teórico

2.1 Protocolo oauth

El marco oauth [2] [3] define un esquema de trabajo para autenticar y autorizar clientes HTTP y facilita las interacciones de estos con los recursos por medio de la definición de un Token de acceso.

En el modelo tradicional de autenticación, el cliente solicita un recurso de acceso restringido o recurso protegido al servidor mediante la autenticación con el servidor utilizando su clave. En caso de necesitar proporcionar acceso a aplicaciones de terceros a los recursos el propietario tendría que compartir sus credenciales. Este modelo es muy limitado y fue ampliamente superado por marcos como el de oauth. Entre los problemas se destacan

- Se pierde el manejo de la contraseña y se la expone a terceros.
- Las aplicaciones de terceros obtienen un acceso excesivamente amplio al recurso, dejando a los propietarios de recursos sin capacidad de restringir la duración o el acceso a un subconjunto limitado de recursos.
- Los propietarios de recursos no pueden revocar el acceso a un tercero individual sin revocar el acceso a todos los que tienen acceso.

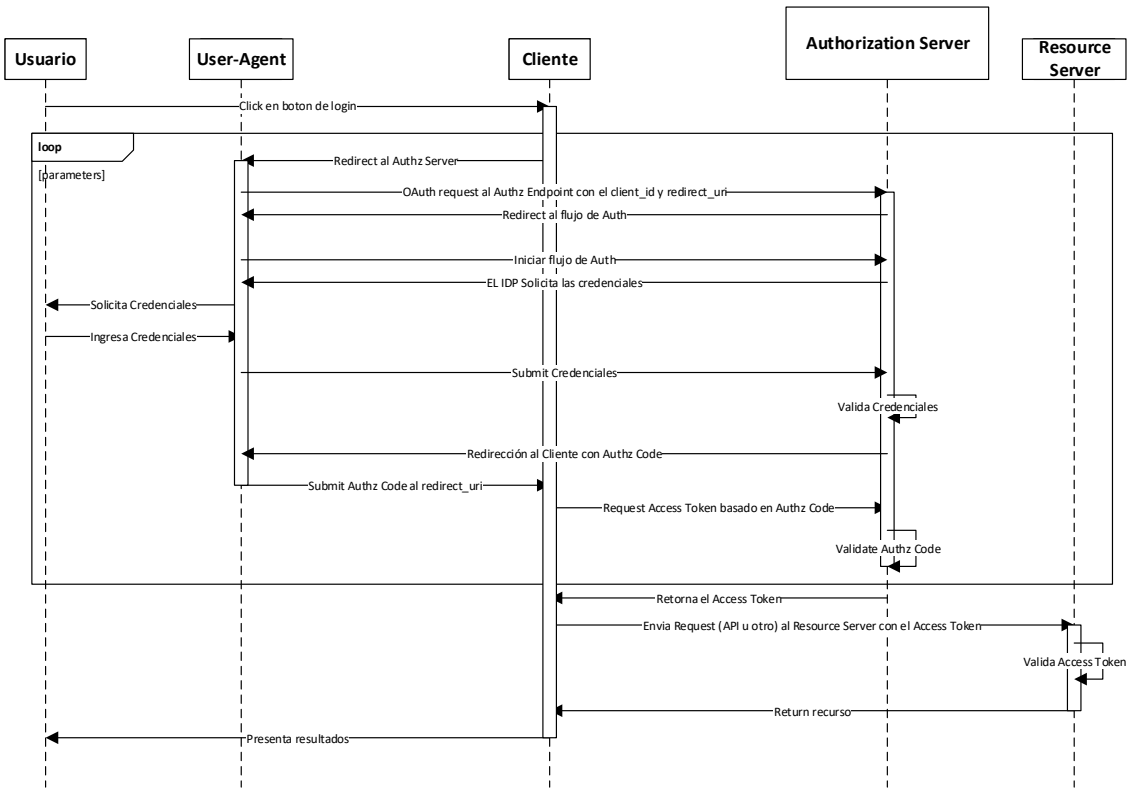
OAuth aborda estos problemas mediante la introducción de una capa de autorización y separando la función del cliente de la del propietario.

El cliente solicita el acceso a los recursos controlados por el propietario. En lugar de utilizar las credenciales del propietario del recurso, el cliente obtiene un token de acceso que denota un alcance específico, duración y otros atributos de acceso. El cliente usa el token de acceso para acceder a los recursos protegidos alojados por el servidor de recursos. Por último, el servidor de recursos valida el token y brinda el acceso al recurso protegido.

Por ejemplo, un usuario final (propietario del recurso) puede otorgar a un servicio online de impresión el acceso a sus fotos almacenadas en un servidor de fotos. sin compartir su nombre de usuario ni contraseña con el servicio de impresión.

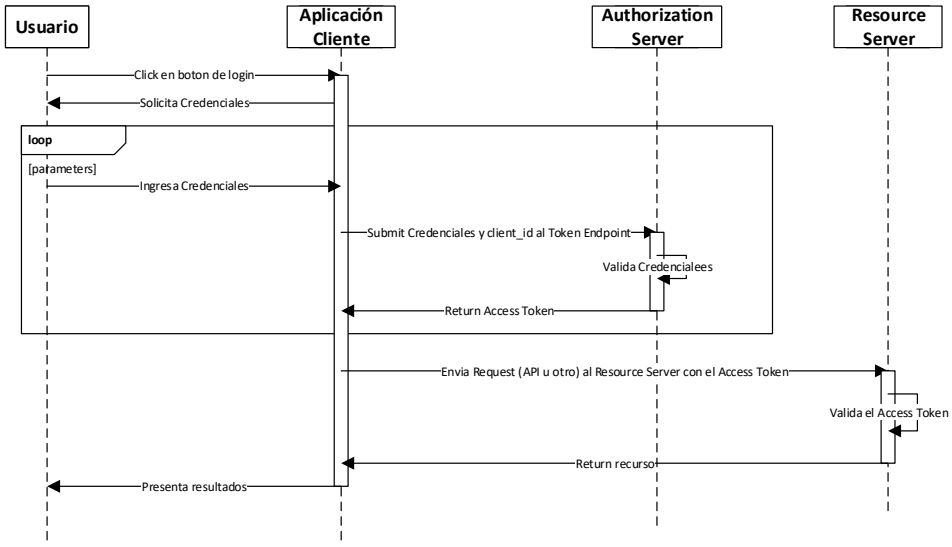
2.1.1 Flujo Authorization Credentials

Este es el flujo principal de oauth, requiere de un backend de la aplicación denominado cliente. En el diagrama se puede ver que primero hay una etapa donde el usuario interactúa con el navegador y este con el cliente de tal forma de obtener un código de autorización. Posteriormente el cliente obtendrá el Access token directamente del servidor de autorización.



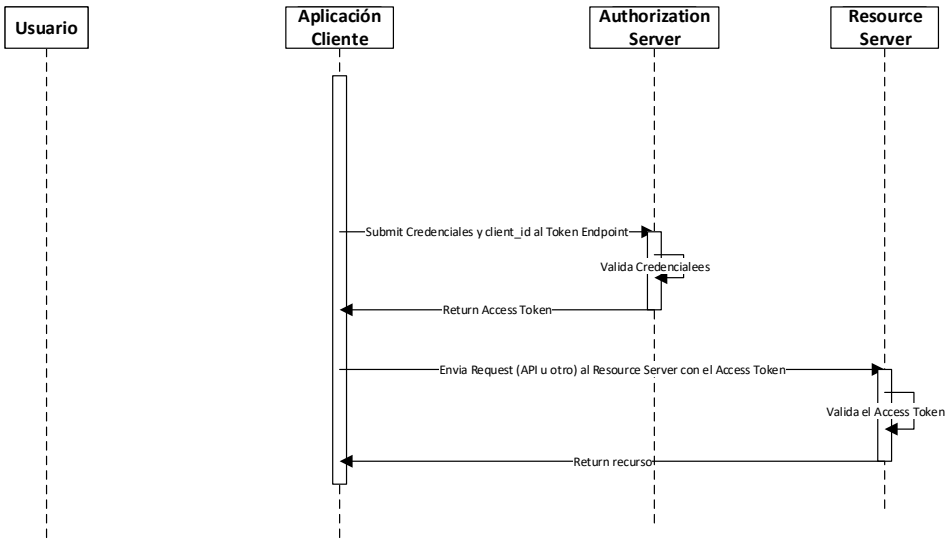
2.1.2 Flujo Resource Owner Password

Notar que el usuario entrega la clave a la aplicación cliente, quien inicia el flujo y conoce la clave. El Access Token es recibido por el cliente. Este flujo solo se puede utilizar en aplicaciones desarrolladas en la propia organización o de fuentes confiables ya que la aplicación cliente tiene acceso a la clave del usuario.



2.1.3 Client Credentials

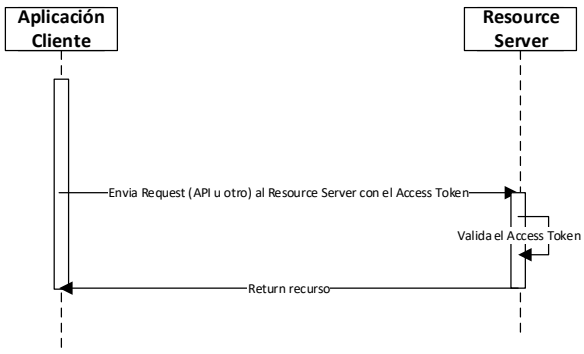
Notar que no hay interacción con los usuarios. Este flujo se puede usar en el consumo de apis que son batch y que no están ligadas a un usuario en particular.



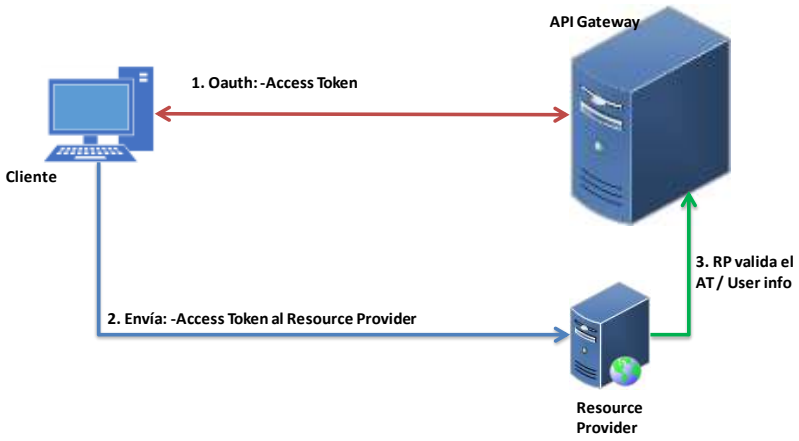
2.1.4 Validación de Access Tokens

El proveedor de recursos recibe de la aplicación cliente una llamada para el acceso de un recurso junto con el token de acceso.

Antes de dar acceso al recurso o dato el token de acceso debe ser validado



Dicha validación se hace invocando el endpoint userinfo del proveedor de acceso. Dicho endpoint proporciona los datos del usuario.



2.1.1 Alternativa proveedores desacoplados por JWT.

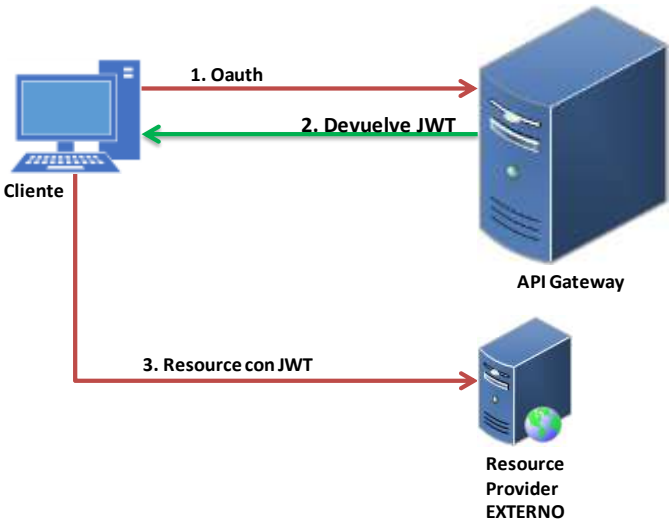
En caso de que el proveedor de acceso no pueda invocar el endpoint de userinfo se puede usar de alternativa que el Access token no sea un string oscuro sino un JWT[4] [5] (Json Web Token).

Otro caso que se puede resolver de esta forma es cuando los requerimientos de performance hacen que la invocación a userinfo no sea práctica.

Un JWT es un registro autocontenido firmado digitalmente que tiene tres secciones separadas por punto y codificadas en base 64. La primera sección es el encabezado que incluye una descripción de los algoritmos de firma, la segunda sección es un JSON con los datos del usuario y la tercera sección es un JSON con la firma de los dos primeros.

Es decir el JWT es un protocolo de autenticación basado en el RFC 7519[6], que define la forma de emitir y validar tokens firmados. La información incluida es en formato JSON y el método de firmado puede ser un algoritmo HMAC o un par de claves RSA.

OAuth como framework de autorización puede utilizar diferentes tipos de tokens entre ellos el propio token JWT.



Esta característica simplifica la implementación de un resource provider ya que se debe limitar a validar la firma del token JWT y tomar los datos básicos del usuario desde el JSON.

2.1.1 Scope

Es un mecanismo en OAuth 2.0 para limitar el acceso de una aplicación a la cuenta de un usuario. Son una lista de identificadores que se utilizan en el proceso de autorización de las APIs en los tokens OAuth, permiten determinar los datos qué se autoriza acceder por parte del cliente.

Esta lista de identificadores está separada por espacios y se utilizan en el proceso de autorización y consentimiento de las APIs en los tokens OAuth sirven para determinar el acceso que solicita la aplicación y a qué se autoriza acceder por parte del usuario. Por ejemplo: datos del usuario, fotos, mensajes de usuarios. etc,

2.1.1 Duración del Access Token y uso del Refresh Token

Como se mencionó los tokens de acceso son claves proporcionadas por el servidor que permite el acceso a los recursos (APIs). Es importante destacar que el Access Token posee un tiempo limitado a partir del cual el token deja de ser válido.

El refresh token es una cadena también proporcionada por el servidor de autorización a la aplicación que permite durante un intervalo de tiempo que es parametrizable solicitar nuevos access token después de que estos hayan caducado. Se evita de esta forma que el sistema solicite nuevamente las credenciales al usuario.

2.2 Protocolo OpenID Connect

El protocolo OpenID Connect 1.0 [7] brinda una capa de identidad sobre el protocolo OAuth 2.0. Permite a los Clientes verificar la identidad del Usuario en función de la autenticación realizada por un Servidor de autorización, así como obtener información de perfil básica sobre de una manera interoperable y de tipo REST. Permite a clientes de todo tipo, incluidos clientes basados en web, móviles y JavaScript, solicitar y recibir información sobre sesiones autenticadas y usuarios finales.

2.2.1 Scope

Como ya se mencionó el scope es un mecanismo en OAuth 2.0 para limitar el acceso de una aplicación a la cuenta de un usuario.
Los scopes que incorpora openid son:

Scope	Descripción del scope
Openid	(standard) Obligatorio para usar open id
Profile	(standard) Se pide acceso a los datos del usuario
Email	(standard) Se pide acceso al email del usuario

Así mismo se establece que las customizaciones pueden incorporar scopes. En un apartado posterior se verán los scopes definidos durante el proyecto.

2.2.1 Userinfo

El servicio UserInfo es un recurso protegido de OAuth 2.0 que devuelve una afirmación (Asertion) con información sobre el usuario final autenticado.
Para obtener los datos del Usuario final, el Cliente realiza una solicitud al Punto final de Información de usuario utilizando un Token de acceso obtenido a través de la Autenticación de conexión OpenID. Normalmente están representadas por un objeto JSON que contiene una colección de pares de nombre y valor.

De acuerdo a la especificación CORE de openid[8] el servicio UserInfo Endpoint debe admitir el uso de los métodos HTTP GET y HTTP POST definidos en RFC

2616, soportar TLS y aceptar tokens de acceso como uso de token portador OAuth 2.0 [RFC6750] y debería admitir el uso de Cross Origin Resource Sharing (CORS) [CORS] y otros métodos, según corresponda, para permitir que los clientes Java Script accedan al punto final.

2.3 Protección de APIs

Actualmente, existe un auge en el uso de las APIs[9] y un gran interés por parte de las empresas en incorporar plataformas API Management para darle seguridad[10] a sus APIs.

La estrategia para securizar APIs es un equilibrio entre la seguridad y la usabilidad. Una estrategia compleja disuade a usuarios malintencionados, pero también hace difícil su uso por parte de usuarios legítimos.

Es importante evaluar y priorizar:

- Autenticar y autorizar a los clientes que consumen las APIs
- Establecer el nivel de confianza de cada uno de ellos.
- Solicitar la autorización explícita de los consumidores.
- Analizar y resolver la delegación de la autorización para acceder a los datos.

OAuth/ OpenIDConnect representan un marco de trabajo que se presenta como respuesta a esas necesidades. Actualmente se ha convertido en el estándar de facto en el uso de APIs.

Permite a las aplicaciones un acceso limitado por los scopes a los recursos de los usuarios, Como mencionamos anteriormente sin tener que proporcionar las credenciales, desacoplando la autenticación y la autorización a los datos.

3 Requisitos no funcionales

3.1 Escalabilidad y Rendimiento

La siguiente tabla resume los requisitos de escalabilidad definidos:

	Requisitos de confiabilidad
Escalabilidad	La solución de administración de API puede escalar de dos maneras: Horizontalmente, agregando más nodos al clúster. Verticalmente, aumentando las especificaciones de hardware de los nodos.
Rendimiento	La solución funcionará en el entorno de producción en modo activo-activo de alta disponibilidad, por lo tanto, debe tener el tamaño adecuado para que ningún servidor utilice más del 60% de su capacidad de procesamiento normal.

3.2 Recuperación ante desastres

La recuperación ante desastres[11] se puede describir como el establecimiento y, cuando sea necesario, la ejecución de los procesos y procedimientos asociados con la garantía de que los sistemas o servicios de TI o de negocios requeridos se pueden recuperar luego de un desastre (natural o de otro tipo) dentro de los plazos acordados.

RTO es la cantidad aceptable de tiempo que se tarda en restaurar un servicio después de un desastre; el servicio de gestión de fallas, cambio y transición se debe restaurar dentro de las 4 horas.

RPO es la cantidad aceptable de pérdida de datos que un sistema podría soportar como resultado de un desastre. Se definio que se puede tolerar la pérdida de datos de hasta 2 horas.

3.3 Mantenimiento

La siguiente tabla resume los requisitos para el mantenimiento del sistema:

	Requisitos de mantenimiento
Indisponibilidad durante el mantenimiento programado	La solución se puede actualizar sin impactos de disponibilidad siempre que las actualizaciones se apliquen gradualmente a los nodos del clúster. Esta estrategia permite probar parches en un nodo específico antes de que se propaguen a los otros nodos. Actualmente, la actualización y la aplicación de soluciones se realizan de forma manual: no hay herramientas para la actualización automática.

3.4 Respaldo y Recuperación de la infraestructura

La siguiente tabla resume la estrategia de backup:

Servidor	Estrategia de backup
API Gateway	Hay dos tipos diferentes de copias de seguridad para realizar[12]: Virtual machine snapshot – extraído en eventos excepcionales. Gateway Database Backup – extraído diariamente a través de Backup API.
API Developer Portal	Hay dos tipos diferentes de copias de seguridad para realizar: Virtual machine snapshot – extraído en eventos excepcionales. Portal Database Backup – extraído de los volúmenes de la base de datos del contenedor Docker.

La siguiente tabla resume la estrategia de recuperación:

Servidor	Estrategia de recovery
API Gateway	Si se puede recuperar el servidor API Gateway, es posible restablecer las bases de datos de Gateway utilizando la última copia de seguridad diaria. Si se pierde el servidor API Gateway, es posible restaurar el último snapshot de la máquina virtual, seguida de una restauración de la base de datos de Gateway.

Servidor	Estrategia de recovery
API Developer Portal	Si se puede recuperar el servidor del Portal del desarrollador, es posible restaurar las Bases de datos del Portal con la última copia de seguridad diaria. Si se pierde el servidor del Portal de Desarrolladores, es posible restaurar el último snapshot de la máquina virtual, seguido de una restauración de la Base de Datos del Portal.

3.5 Monitoreo

La siguiente tabla resume la estrategia de monitoreo:

Servidor	Estrategia de monitoreo
API Gateway	Los servidores API Gateway se pueden monitorear a través de SNMP. Además, es posible ejecutar pruebas de estado del servidor utilizando API Gateway Ping API.
API Developer Portal	Portal del desarrollador se puede controlar mediante la comprobación del estado de sus contenedores en el host Docker.

3.6 Atributos de seguridad

Los requisitos de seguridad están definidos por los documentos publicados por el Grupo de seguridad de la información. Estos han sido revisados y se han identificado los siguientes requisitos de seguridad relevantes para este trabajo.

3.6.1 Roles de usuario, Funciones y Alcance

La siguiente tabla resume los actores y las acciones que pueden realizar en el sistema. Además, se resume el alcance de la información a la que pueden acceder.

Persona	Tareas y / o historia de usuario funcional Cross-Reference	Alcance aproximado
Desarrollador	• Proporcionar servicios para el consumo. • Consumir servicios internos. • Transformar formatos de datos para servicios.	API Gateway
Arquitecto API	• Proporcionar servicios para el consumo. • Proteger los servicios de las amenazas. • Administrar servicios en el Portal de Desarrolladores. • Mantener el contenido del sitio.	API Gateway, API Developer Portal
Equipo de negocios	• Hacer cumplir las restricciones de uso a los servicios	API Developer Portal
Seguridad de información	• Proteger los servicios. • Acceso a logs y auditoría	API Gateway
API Consumidor	• Enroll developers • Acceder a los servicios expuestos	API Developer Portal
Web Designer	• Ajustar la identidad visual del sitio	API Developer Portal
Operaciones	• Mantenimiento del Cluster. • Mantenimiento de recursos de servicio (JDBC, JMS, Message Queues, etc)	API Gateway

3.6.2 Autenticación y Autorización

La siguiente tabla resume los requisitos de autenticación, autorización y titularidad. La tabla también incluye información sobre actores importantes específicos dentro de la solución:

Atributo	Persona	Descripción de requerimientos
Autenticación	Consumidores API - Sistemas internos	Hay dos recomendaciones para este escenario: Certificado de cliente SSL: se usa para integraciones entre sistemas, sin la intervención del usuario final. Se debe almacenar las claves públicas del cliente para autenticar las conexiones SSL. OAuth2 – *Client authentication*: se usa para integraciones entre sistemas que autentican a sus propios usuarios finales. En este escenario, la relación de confianza se establece con los sistemas cliente, que usarán las credenciales de OAuth Client para autenticarse.

Atributo	Persona	Descripción de requerimientos
Autenticación	Consumidores API - Sistemas externos	Hay dos recomendaciones para este escenario: Certificado de cliente SSL: se usa para integraciones entre sistemas, sin la intervención del usuario final. Se debe almacenar las claves públicas del cliente para autenticar las conexiones SSL. OAuth2 – *Client authentication*: se usa para integraciones entre sistemas que autentican a sus propios usuarios finales. En este escenario, la relación de confianza se establece con los sistemas cliente, que usarán las credenciales de OAuth Client para autenticarse.
Autenticación	Consumidores API: usuarios finales internos o externos	El mecanismo autenticado propuesto para este escenario es OAuth2 / OpenID Connect. Hay dos recomendaciones: *Authorization Code*: la aplicación envía sus credenciales de cliente para su validación y las credenciales del usuario final se validan en el Authorization Server. *Resource Owned Password*: la aplicación envía credenciales de cliente y de usuario final para validación. La validación se puede realizar contra el proveedor de identidad

Atributo	Persona	Descripción de requerimientos
Autorización	Clientes de API	La autorización se realiza a través de OAuth2 scopes concedidas a las API de cliente. Las API deben determinar qué scopes están permitidos para tener acceso a los recursos.

3.6.3 Gerencia de usuarios

La siguiente tabla resume los requisitos de geston de usuarios:

Seguridad Identificador de Identidad - Administración de usuarios	Descripción de requerimientos
Virtual Appliance administrative users	Los usuarios adminstractivos de la Virtual Appliance están en custodia del equipo de operaciones.
Policy Manager Users	El acesso a Policy Manager se concede por la pertenencia de usuarios a grupos en el directorio corporativo. Estos grupos de usuarios deben asociarse con las directivas internas de administrador para controlar los privilégios de los usuarios.

3.6.4 Seguridad de datos y comunicaciones

La siguiente tabla resume los requisitos de seguridad y comunicaciones:

Identificador de atributo de seguridad: datos y comunicaciones	Requisito Descripción
HTTPS	La integridad de la comunicación entre las aplicaciones del consumidor y la solución de administración API se logra mediante HTTPS.

Identificador de atributo de seguridad: datos y comunicaciones	Requisito Descripción
SSL Mutuo	Es posible hacer cumplir la comunicación Mutual SSL para lograr la máxima protección para servicios sensibles. Este enfoque requiere un proceso de registro para almacenar los certificados de cliente del consumidor.

3.6.5 Auditoría y log de acceso

La siguiente tabla resume los requisitos de auditoria y log de acceso:

Identificador de atributo de seguridad - Auditoría	Requisito Descripción
Log de acceso	La solución de administración de API recopila datos de auditoría y los almacena en bases de datos internas. Los datos de auditoría también se redireccionan al SIEM de la organización.

4 Implementación

Este capítulo describe la implementación de la herramienta de API Gateway para autenticar y autorizar usuarios en aplicaciones móviles y WEB.

Para la autenticación y autorización de un usuario a una aplicación de un cliente se utiliza el estándar Oauth2 con flujo Authorization Code. Para obtener los datos del usuario se usa OpenId por medio del método estándar Userinfo. Como se mostró en el marco teórico Este método permite obtener el perfil del usuario con la información que se necesita mostrar.

Para que un usuario pueda autenticar y autorizar con una aplicación, la misma debe estar previamente registrada por el Administrador de la Herramienta en la plataforma de APIGW utilizando la herramienta de administración de OAuth (Oauth Manager).

4.1 Oauth2: Autorización y autenticación de Usuarios

El proceso se realiza mediante Oauth2, el protocolo permite a la aplicación un acceso de datos de los usuarios, sin tener que proporcionar las credenciales de estos. Al invocar se recibe un Token de Acceso que representa que el usuario fue autenticado.

Para poder controlar la información que se comparte con los sitios web se utiliza el estándar OpenID que es una extensión de OAuth que agrega y define estrictamente un token de identificación para devolver la información del usuario.

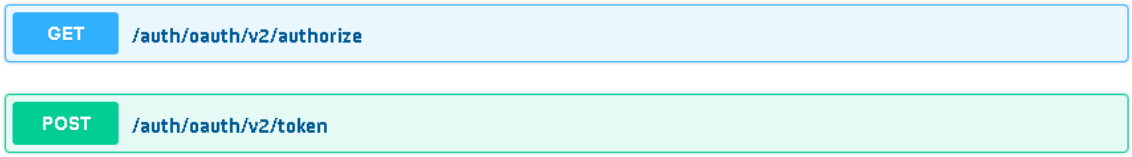
OAuth provee 4 flujos distintos para autenticar y autorizar:

Flujo	Descripción	Usado en
Authorization Code	Logon de aplicaciones con el identity provider manejando la clave.	Aplicaciones mobiles a ser integradas
Implicit	Logon de aplicaciones con el identity provider manejando la clave.	No se usó este método en la implementación.

Password Credencials	La aplicación administra la clave y la envía al flujo durante la invocación. El frontend con el usuario lo pone la aplicación.	Este método solo puede ser usado por desarrollo de la organización.
Client Credentials	No se autentican usuarios. Es para aplicaciones que usan servicios en background.	Es para aplicaciones que usan servicios en background. Este método solo puede ser usado por desarrollo de la organización.

4.2 Authorization Code

El flujo que se describe en esta guía para la autenticación de usuarios es Authorization Code. Este flujo permite la utilización/autorización de Oauth 2.0 y funciona a través de dos endpoints, los cuales son:



El primer endpoint realiza el proceso de autenticación y autorización y genera un numero llamado authcode. Es decir, el cliente invoca a /auth/oauth/v2/authorize, el cual se encarga de mostrar una página de logon al usuario para solicitar las credenciales. Cuando el usuario ingresa las credenciales correctas el endpoint devuelve el código de autorización al programa cliente.

Con este código el programa cliente deberá invocar al segundo endpoint "/auth/oauth/v2/token" que retornará el "Token de Acceso". Este token de acceso tendrá una duración y unos permisos (scope) dependiendo la configuración que tenga la aplicación,

Para poder autenticarse por este flujo durante la configuración del cliente se le entregara:

- **Client Id:** Es un identificador único de la aplicación que requerirá acceder a los recursos en representación del usuario.
- **Client Secret:** Es un código utilizado en ciertos flujos específicos de OAuth2, para la obtención de un access token y refresh token.

- **Scope:** Son los permisos que se están pidiendo sobre la API, se especifica más sobre los scope en el punto 3.2
- **Auth URL:** Se utiliza para la interacción con usuarios, cuando este tiene que identificarse.
- **Access Token URL:** Genera el access_token de OAuth que finalmente es entregado al Usuario, en este proceso no existe interacción del usuario.

La aplicación deberá proporcionar:

- **CallBack URL:** Cuando la autenticación del cliente finaliza, es necesario especificar una URL de vuelta a la aplicación que lleva los parámetros de respuesta al cliente solicitante.
- especificar si va a autenticar usuarios internos, externos o customer.
- Indicar los roles (grupos de AD) que requiere sean enviados en el assertion.

Para un ejemplo de uso de estos parámetros ver el capítulo con el “Prueba de concepto”.

4.3 Open Id Connect

Como se mencionó, OpenId Connect es un protocolo que extiende a OAuth 2 con una capa que permite estandarizar la entrega de información de los usuarios.

Open Id Connect tiene las siguientes funcionalidades que complementa a Oauth:

- Un ID token que nos permite saber los datos básicos del usuario.
- Un nuevo endpoint, UserInfo, que nos permite recuperar más información del mismo.
- Un conjunto de scopes estándar y posibilidad de extensiones.

4.4 Scope

Como ya se mencionó el scope es un mecanismo en OAuth 2.0 para limitar el acceso de una aplicación a la cuenta de un usuario.

Los scopes definidos durante el proyecto fueron:

Scope	Descripción del scope
Openid	(standard) Obligatorio para usar open id
Profile	(standard) Se pide acceso a los datos del usuario
Email	(standard) Se pide acceso al email del usuario
JWT	Si se configura el Access Token será un JWT firmado por la clave privada del servidor. Si no se envía, el AT será un string obscuro.

Interno	La aplicación autenticara usuarios internos
Externo	La aplicación autenticara usuarios externos
Customer	La aplicación autenticara usuarios customer (usuarios del portal web)

Nota: En el perfil de la aplicación el Administrador de la Herramienta pondrá los scopes soportados, luego el desarrollador durante la invocación debe enviar los scopes que requiera. Si el desarrollador envía un scope que no está configurado será ignorado por la herramienta.

4.1 Json Web Token

En caso de utilizar el scope jwt, el sistema generará un token de acceso del tipo JWT

Signature Algorithm	Algoritmo de firma token JWT	RSAv1.5 con SHA-256
Clave de firma	Clave de firma	Se utiliza la clave privada del sitio de APIS.

4.2 Duración del Access Token y del Refresh Token

Como se muestra en la siguiente tabla, se customizaron 4 variables que afectan el token que se retornará

Variable	Parámetro	Valor establecido
oauth2_access_token_lifetime_sec	Cantidad máxima de tiempo (en segundos) que el token puede ser válido después de emitirse.	En 0 para invalidarlo inmediato, por defecto 3600 = 1 hora.
oauth2_refresh_token_lifetime_sec	Vida útil de los refresh tokens emitidos.	604800 segundos = 1 semana. Para tener un token que no caduque, establezca el valor en el valor máximo de 631138520 segundos (20 años).
oauth2_auth_code_lifetime_sec	Vida útil de los code OAuth emitidos.	300 segundos = 5 minutos
id_token_lifetime_s	Tiempo vida de id_token openid.	86400 segundos = 1 día

4.3 OpendID Userinfo

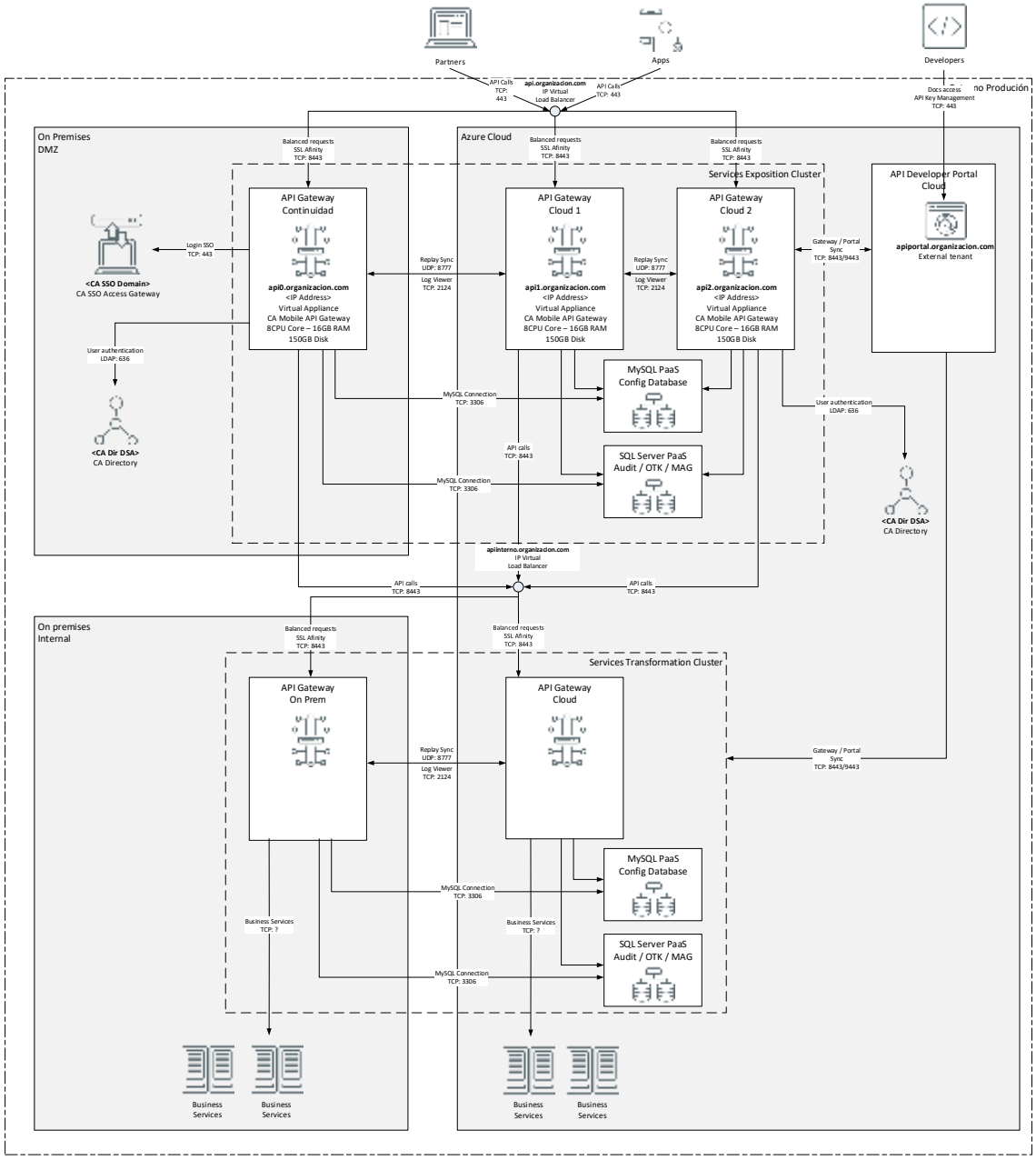
Al agregar el scope openid se identifica que se va poder utilizar este endpoint. Userinfo solo funciona con openid, valida el AccessToken y que el scope contenga “profile”, en ese caso retorna un JSON con las variables del usuario validado en la autorización.

Ejemplo:


```
{
        "sub": "7OfBDBx2FXRszLzrOUEuPYJqZSXkPwf-JREF5HnI7aI",
        "given_username": "SE92710",
        "uid": "SE92710",
        "first_name": "PRUEBAPAD2",
        "last_name": "PRUEBA",
        "mail": "pruebapad2.prueba@organizacion.com",
        "tipo_empleado": "Interno",
        "roles": "IDM-DESA-COMERCIAL, IDM-DESA-DESPACHANTE"
}
```


4.4 Diagrama de arquitectura

En el diagrama se muestra el punto de entrada del servicio y la distribución de módulos adoptada. Básicamente se incluyeron en alta disponibilidad dos módulos de API Gateway en la nube de Azure y un módulo en la DMZ del centro de cómputos de la organización.



El diagrama también muestra la conexión con los directorios de usuario y los servicios a ser protegidos.

4.5 Roles de administración y desarrollo

Interacción de Actores

Los siguientes diagramas ilustran la interacción de personas en los procesos de API Management.

Actores Internos:

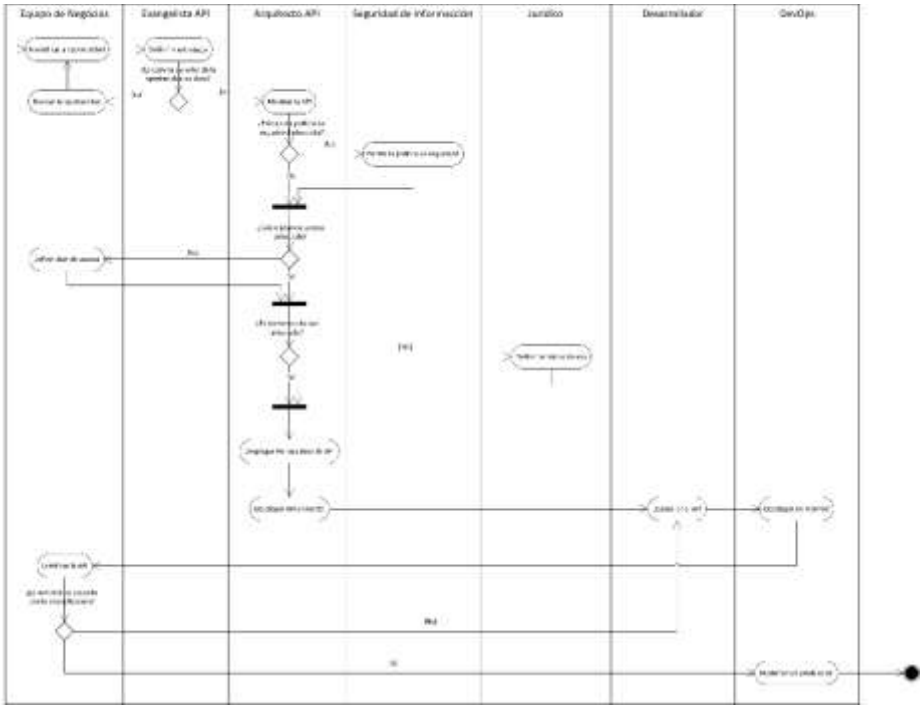


Diagrama de actividad - Proceso de creación de API

El rol desempeñado por el **Equipo de Negocios** busca representar los intereses comerciales de la organización, evaluando las oportunidades para usar API y traducirlas en beneficios para la compañía. El equipo de negocios también es responsable de probar la API desarrollada y certificar que está funcionando según lo previsto.

El rol desempeñado por **API Evangelist** es liderar la iniciativa API Management a través del conocimiento técnico y las perspectivas del mercado, creando un puente entre las partes interesadas técnicas y comerciales. Es su responsabilidad hablar con las partes interesadas y definir las estrategias API que mejor se adapten a los objetivos de la empresa y también ajustar cada nueva demanda API a esta estrategia.

El rol desempeñado por el **Arquitecto de API** es modelar las API de una manera que se adhiera a las estrategias API de la compañía, sin renunciar a las mejores prácticas de mercado. A esto se añade la misión de facilitar la adopción de la API, teniendo en cuenta a los socios y desarrolladores. La definición de políticas

de seguridad, términos de uso y planes de acceso a las API también es responsabilidad del Arquitecto.

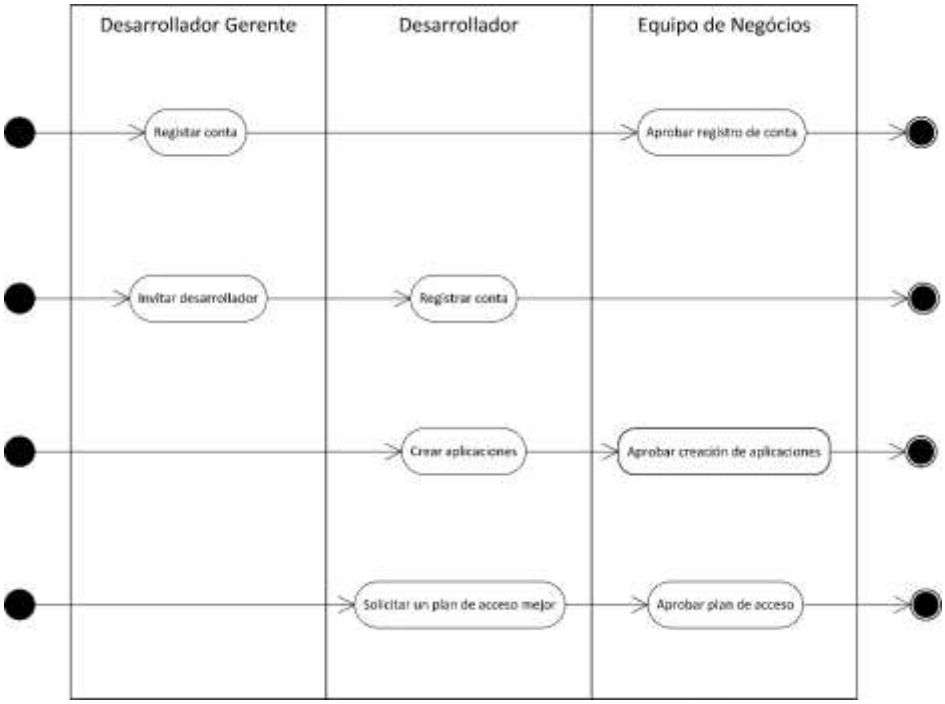
La función de la **Seguridad de la Información** es definir las políticas de seguridad que se aplicarán a los distintos tipos de API, como el control de acceso, los mecanismos de comunicación segura, la prevención de amenazas y el manejo de incidentes para las API.

El rol desempeñado por **Legal** busca evaluar y definir los términos de uso apropiados para las API de acuerdo con el contenido expuesto y la propiedad de la compañía en él. Deben tenerse en cuenta los elementos como la privacidad, el uso de la marca, la propiedad intelectual y la retención de datos por parte de los consumidores API.

El rol de **Desarrollador** es construir la API a partir del modelo y las políticas definidas por el Arquitecto de API, haciendo toda la integración y transformación de los datos del sistema fuente para cumplir con el nuevo contrato.

La función de **DevOps** es definir y mantener el conducto de implementación de las API entre los entornos mediante la aplicación de mecanismos de integración continua, pruebas automatizadas y la obtención de métricas de calidad a partir de las API publicadas.

Actores Externos:



La función del **Administrador de Desarrolladores** la realiza el primer desarrollador que se suscribe en la plataforma API. Desde su registro, se creará la organización a la que pertenecerá, así como a todos los demás desarrolladores invitados por ella. Además, este desarrollador heredará todas las características del rol de Desarrollador que se describe a continuación.

El rol del **Desarrollador** es crear aplicaciones que usen las API que ofrece la plataforma. Es su responsabilidad registrar las aplicaciones, obteniendo así las credenciales para consumir las API. También solicitará mejoras en el plan de acceso otorgado a la organización.

4.6 Resultados

Los resultados de las pruebas de estrés (Ver Anexo 1) indican que la plataforma soporta consistentemente las cargas pesadas, esto puede ser observado en la progresión lineal de los números durante el incremento de solicitudes concurrentes.

5 Prueba de Concepto

Este capítulo muestra la configuración de un cliente y su uso para consumir un servicio.

La organización entregó:

- Client_Id
- Client_Secret
- Scopes
- Auth URL
- Access Token URL:

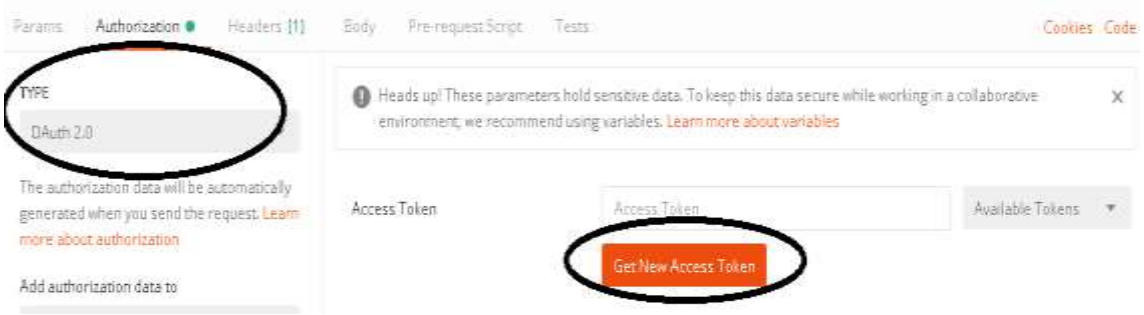
El desarrollador de la aplicación deberá entregar:

- El “CallBAck” de la misma.
- especificar si va a autenticar usuarios internos, externos o customer.
- Indicar los roles (grupos de AD) que requiere sean enviados en el assertion.

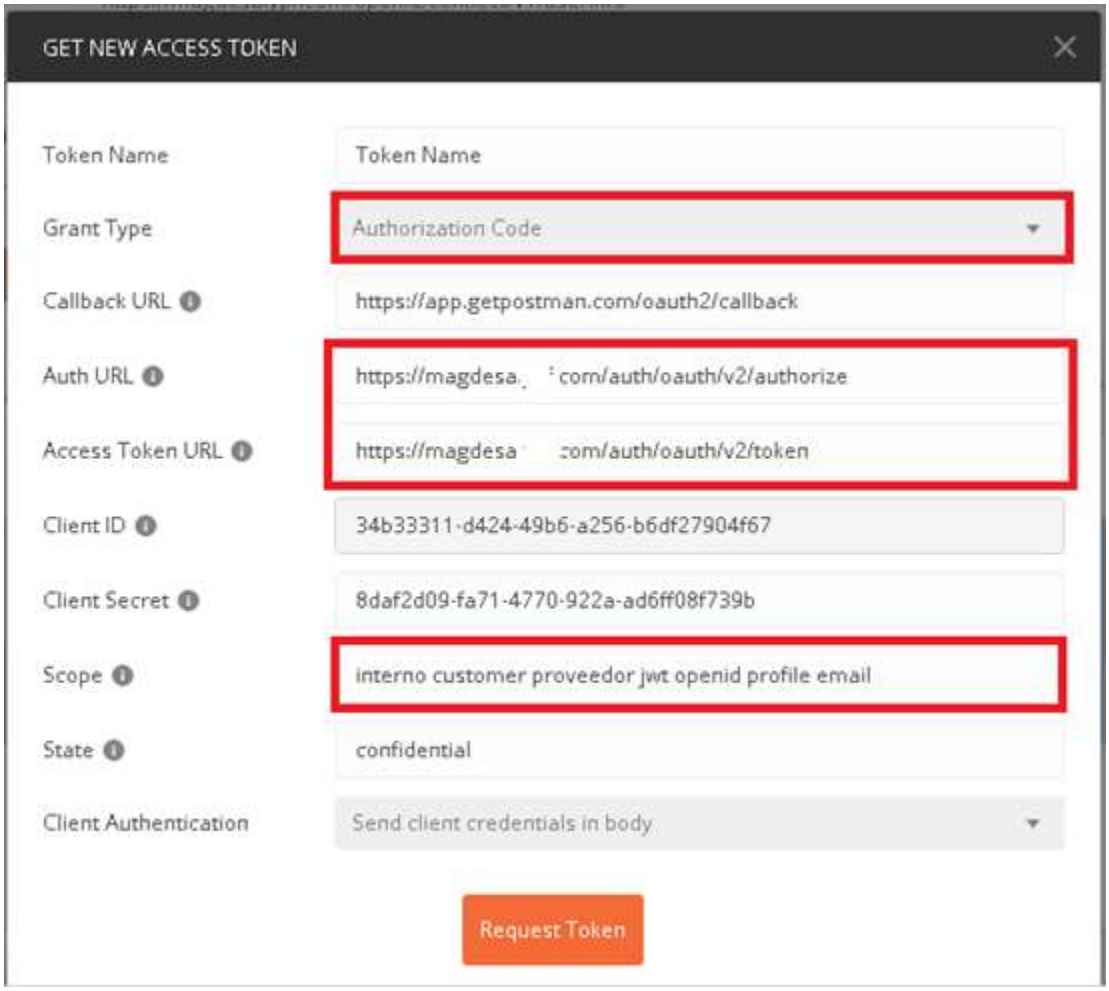
Nota: estos parámetros están descriptos anteriormente.

Autorización y Autenticación desde Postman[13]**:**

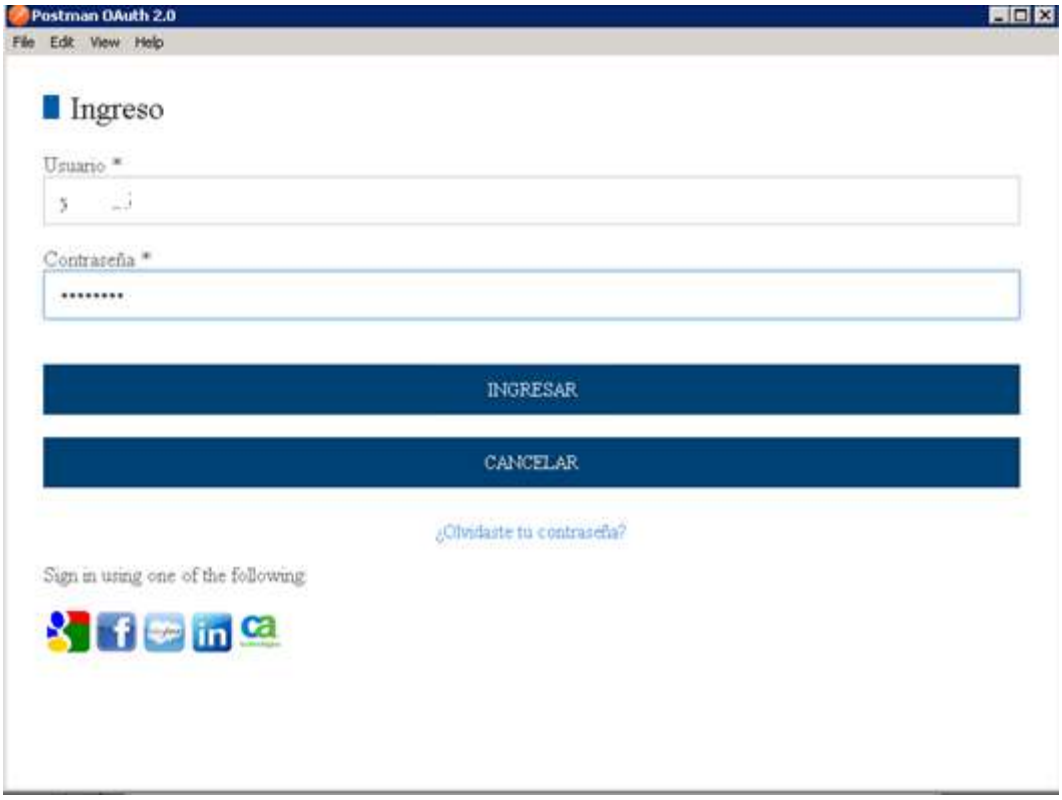
En postman[14] en la pestaña de Authorization debe ir OAuth 2.0 y luego ir a “Get New Access Token”



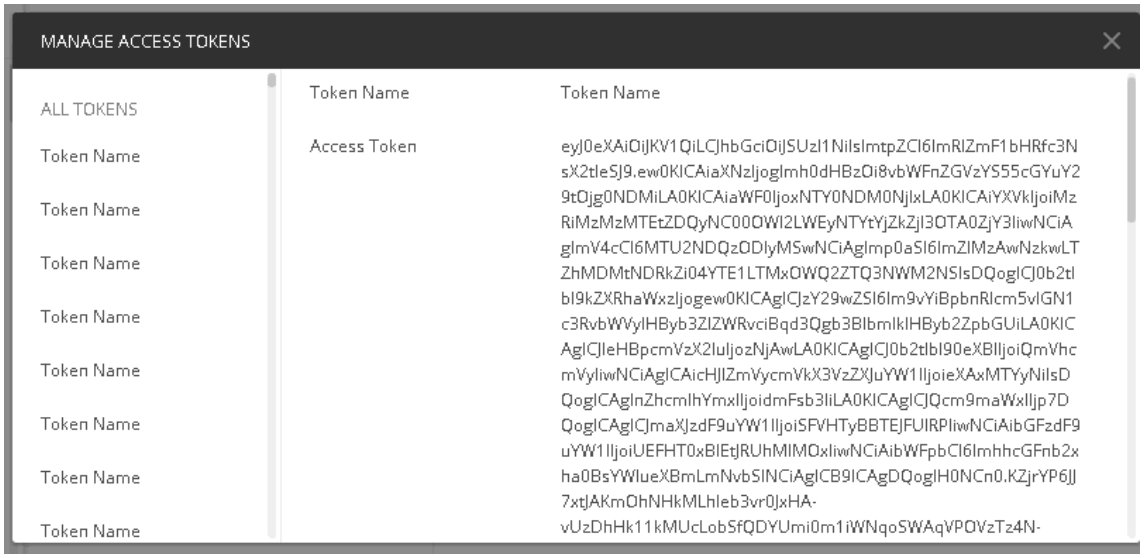
Nos visualizara una pantalla para elegir el tipo de flujo. Se selecciona Authorization Code y se completa la información que requiere este flujo como se ve en la figura:



Al requerir nos mostrara un login para poder ingresar y poder hacer la autenticación:



Si las credenciales son las correctas nos devolverá un Access_Token. Dado que entre los scopes se envió jwt, el sistema retornara un Access Token en formato JWT.



Cuando el usuario se encuentre autorizado y autenticado nos devolverá un Access_Token en formato JWT.

A partir de este punto se tienen dos opciones para visualizar la información del usuario:

La primera forma es la utilización del servicio Userinfo de OpenId. La Url depende del ambiente donde se esté utilizando, para esta prueba se utilizara el ambiente de desarrollo:

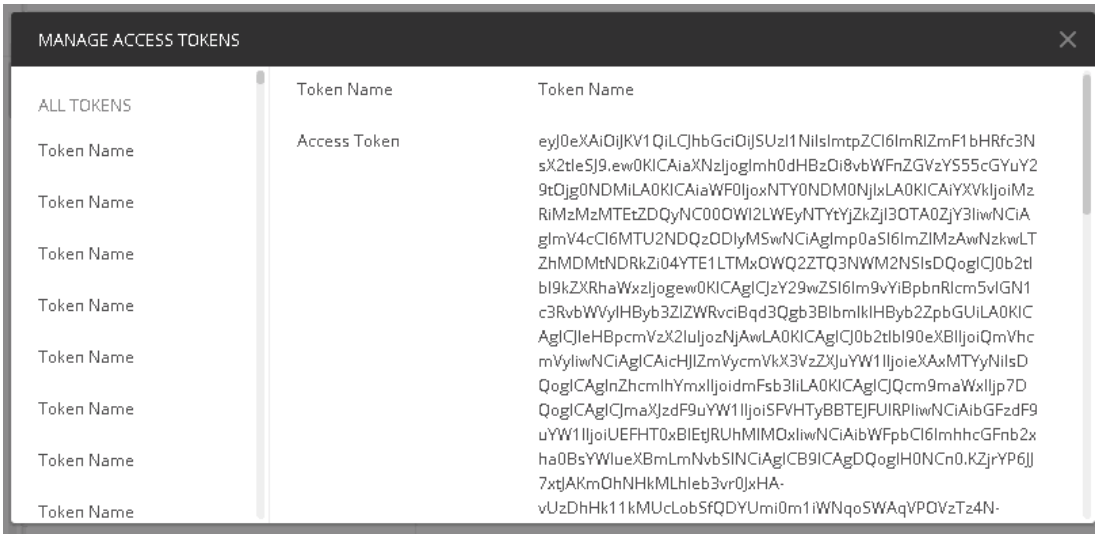
URL UserInfo: https://magdesa.organizacion.com/openid/connect/v1/userinfo

Cuando se tiene el Access_Token se envía al URL

Retorna la siguiente información:


```
{
    "sub": "-1MHi733vjarbKw4e8gtFvkXqf3dFySkypBMFCiS6jI",
    "given_username": "joseprueba001@mailinator.com",
    "uid": "ED100113",
    "first_name": "JOSEPRUEBA",
    "last_name": "PRUEBA",
    "mail": "joseprueba001@mailinator.com",
    "tipo_empleado": "Proveedor",
    "CUIT": "20-10101010-5",
    "roles": "IDM-B                    to-desa"
}
```


La segunda forma de tener acceso a los datos de usuarios solo es posible cuando el token es JWT. En este caso podremos validarlo y decodificarlo



En la prueba de concepto tomamos el Access token y lo decodificamos con la página https://jwt.io. Dicha WEB nos permite decodificar el JWT mostrando de la siguiente manera:

6 Conclusiones

En el presente trabajo se describe la puesta en marcha de una infraestructura de Autenticación y autorización para aplicaciones WEB y aplicaciones Móviles de una organización de primer nivel.

Es importante destacar que se trabajó en un ambiente de alta disponibilidad ya que la misma tiene requisitos de trabajo con una disponibilidad de 7 por 24 y centenares de APIS a ser protegidas.

Como framework de autenticación / autorización se seleccionaron los protocolos oauth2 y openid Connect que se implementaron en el API Gateway de CA[12]. La misma fue adaptada al ambiente de la organización por medio de su integración con los distintos directorios de usuario que posee la compañía. Usuarios internos, externos separados en proveedores y clientes y usuarios finales.

Los resultados de las pruebas de estrés indican que la plataforma soporta consistentemente las cargas pesadas, esto puede ser observado en la progresión lineal de los números durante el incremento de solicitudes concurrentes. Las pruebas concluyeron con tasas de éxito por encima del 98% en todos los escenarios, lo cual es sobresaliente. Se demuestra que no se pudo encontrar límites ni cuellos de botella evidentes en la infraestructura. En resumen, el ambiente podría, en un escenario de estrés, soportar un pico de alrededor de 600 usuarios concurrentes con 800 transacciones/segundo. Si en un futuro, se requiere aumentar las capacidades/performance, se podrían incorporar nodos al clúster o aumentar los recursos de procesamiento de los nodos actuales.

La implementación fue exitosa, y actualmente el sistema se encuentra en operación. La combinación de OpenID Connect y OAuth2 resultó un buen enfoque para asegurar las aplicaciones de la organización y permitió resolver los problemas de seguridad típicos de las aplicaciones móviles, nativas y web actuales.

7 Bibliografía

[1] «IdentityServer3: The big Picture». [En línea]. Disponible en: https://identityserver.github.io/Documentation/docsv2/overview/bigPicture.html. [Accedido: 23-ago-2019].

[2] D. Hardt <dick.hardt@gmail.com>, «The OAuth 2.0 Authorization Framework». [En línea]. Disponible en: https://tools.ietf.org/html/rfc6749. [Accedido: 23-ago-2019].

[3] D. Hardt y M. Jones, «The OAuth 2.0 Authorization Framework: Bearer Token Usage». [En línea]. Disponible en: https://tools.ietf.org/html/rfc6750. [Accedido: 23-ago-2019].

[4] J. Bradley, N. Sakimura, y M. B. Jones, «RFC7797 JSON Web Token (JWT)». [En línea]. Disponible en: https://tools.ietf.org/html/rfc7519. [Accedido: 23-ago-2019].

[5] «A Beginner's Guide to JWTs in Java», *Stormpath User Identity API*, 21-jun-2016. .

[6] J. Bradley, N. Sakimura, y M. B. Jones, «RFC7519 JSON Web Token (JWT)». [En línea]. Disponible en: https://tools.ietf.org/html/rfc7519. [Accedido: 28-ago-2019].

[7] «OpenID Connect | OpenID». [En línea]. Disponible en: https://openid.net/connect/. [Accedido: 16-jun-2019].

[8] «Final: OpenID Connect Core 1.0 incorporating errata set 1». [En línea]. Disponible en: https://openid.net/specs/openid-connect-core-1_0.html#UserInfo. [Accedido: 23-ago-2019].

[9] «OAuth 2.0: equilibrio y usabilidad en la securización de APIs», *Paradigma*, 29-oct-2018. [En línea]. Disponible en: https://www.paradigmadigital.com/dev/oauth-2-0-equilibrio-y-usabilidad-en-la-securizacion-de-apis/. [Accedido: 23-ago-2019].

[10] Scott Morrison, Rob Wilson, Matt McLarty, *Securing Microservice APIs*, O'Reilly Media, Inc. 2018.

[11] itsafer, «RTO y el RPO del Plan de Recuperación ante Desastres», *Itsafer*, 14-nov-2018. .

[12] «CA API Gateway Home - CA API Gateway - 9.4 - Documentación de CA Technologies». [En línea]. Disponible en: https://docops.ca.com/ca-api-gateway/9-4/en. [Accedido: 26-ago-2019].

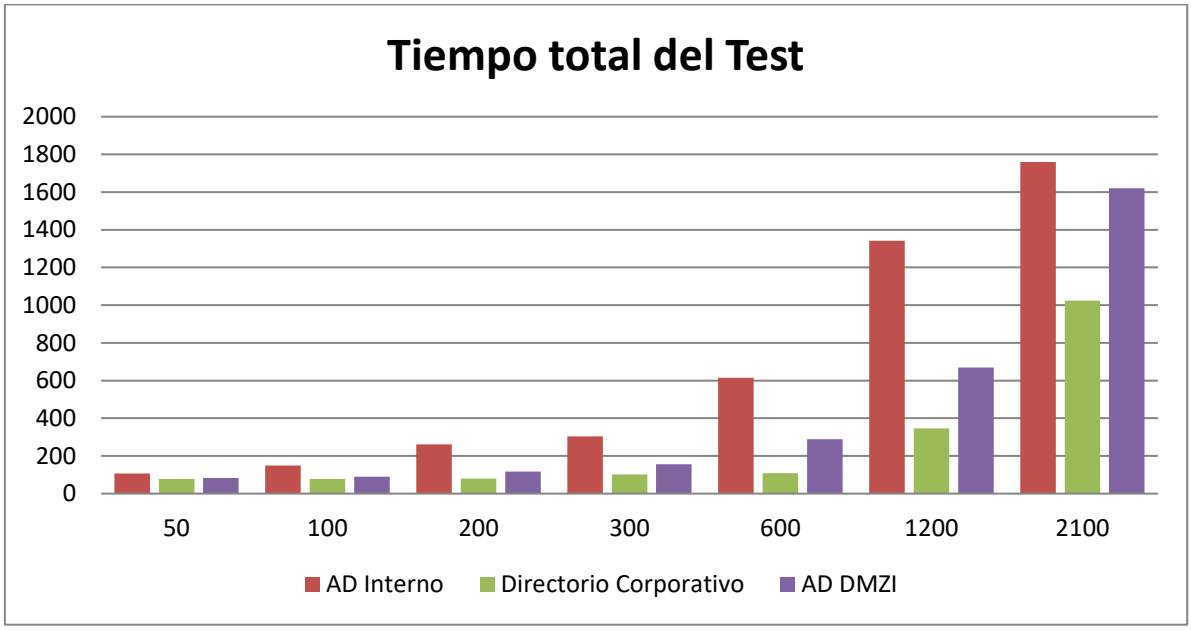
[13] «Postman | API Development Environment», *Postman*. [En línea]. Disponible en: https://www.getpostman.com. [Accedido: 26-ago-2019].

[14] «Authorization with Postman», *Postman Learning Center*. [En línea]. Disponible en: https://learning.getpostman.com/docs/postman/sending_api_requests/authorization/. [Accedido: 26-ago-2019].

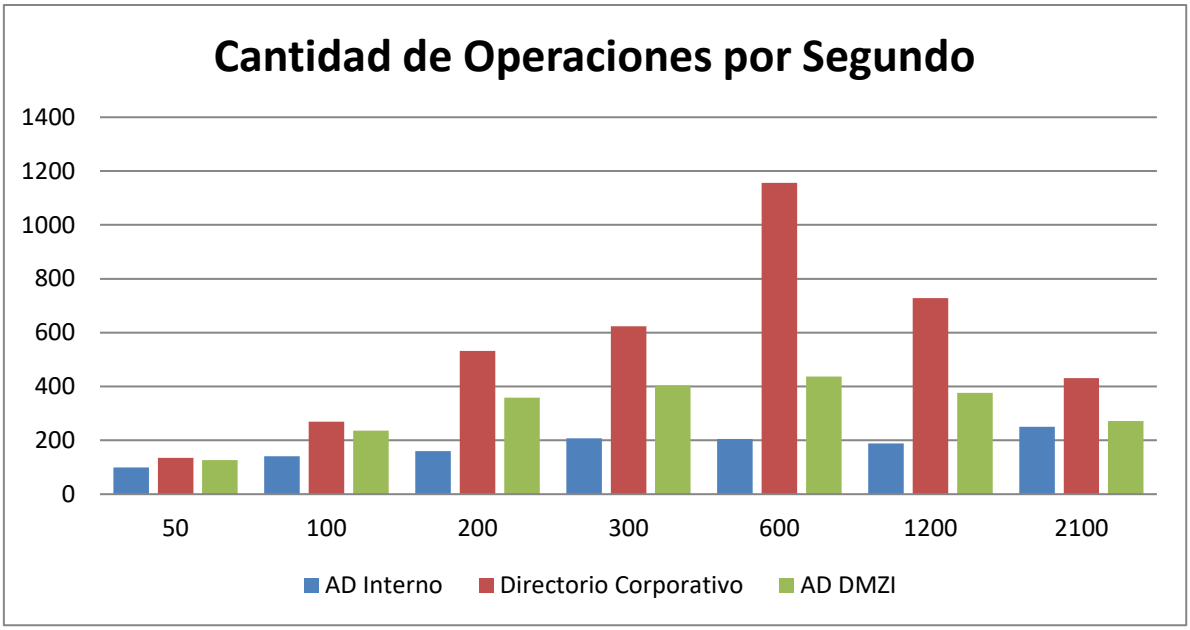
8 Anexo 1 Pruebas de Estrés

8.1 Datos de las Pruebas

Las pruebas se iniciaron con 50 usuarios concurrentes, autenticando mediante OAuth 2.0 con el flujo Resource Owner Password Credentials. Luego se aumentó gradualmente hasta llegar a los 2100 usuarios concurrentes.
En el siguiente gráfico se puede observar el tiempo en segundos de la duración de las pruebas dependiendo de la cantidad de usuarios concurrentes.



Como se puede observar, el directorio con el que menos tiempo tomó concluir la prueba es el Directorio Corporativo, en cualquier cantidad de usuarios.
En el siguiente gráfico se puede observar una comparación de la cantidad de operaciones por segundo dependiendo del directorio al que pertenece el usuario.



Se puede observar que el directorio con mejor performance es el Directorio Corporativo. El cual, en algunos casos duplicó la cantidad de operaciones por segundo de los demás directorios.

En cuanto a errores, en el AD Interno se obtuvo una precisión del 99,89%, en el Directorio Corporativo del 99,91% y en el AD DMZI del 99,92%, siendo este último el de mayor precisión.

Se observó también que la latencia, en algunos casos, puede ser considerada extremadamente alta.

8.2 Datos detallados de las pruebas

8.2.1 AD Interno

Usuarios	Operaciones	Tiempo (s)	Op/s	Latencia Min	Latencia Max	% Error	CPU #1	CPU #2
50	10500	106	99,0566	191	5890	0,00 %	5,80 %	5,00 %
100	21000	149	140,9396	195	12514	0,00 %	10,00 %	8,90 %
200	42000	262	160,3053	191	9268	0,00 %	12,70 %	10,60 %

300	63000	304	207,2368	188	75857	0,03%	39,00%	32,00%
600	126000	615	204,878	143	258896	0,00%	15,20%	18,40%
1200	252000	1341	187,9195	88	308999	0,14%	78,70%	77,10%
2100	441000	1759	250,7106	81	281410	0,02%	84,70%	83,20%

8.2.2 Directorio Corporativo

Usuarios	**Operaciones**	**Tiempo (s)**	**Op/s**	**Latencia Min**	**Latencia Max**	**% Error**	**CPU #1**	**CPU #2**
50	10500	78	134,6153846	182	2280	0,60%	3,50%	4,30%
100	21000	78	269,2307692	187	2976	0,00%	8,90%	10,00%
200	42000	79	531,6455696	179	4040	0,00%	11,10%	12,60%
300	63000	101	623,7623762	187	4736	0,03%	32,20%	30,00%
600	126000	129	976,744186	187	11802	0,00%	56,60%	62,90%
1200	252000	346	728,3236994	167	88827	0,04%	84,80%	84,60%
2100	441000	1024	430,6640625	97	162390	0,04%	90,20%	92,10%

8.2.3 AD DMZI

Usuarios	Operaciones	Tiempo (s)	Op/s	Latencia Min	Latencia Max	% Error	CPU #1	CPU #2
50	10500	83	126,5060241	186	2636	1,60 %	10,00%	9,60 %
100	21000	89	235,9550562	187	4533	0,40 %	10,20%	13,00%
200	42000	117	358,974359	174	7662	0,10 %	15,50%	17,70%
300	63000	156	403,8461538	187	12434	0,00 %	32,00%	34,00%
600	126000	288	437,5	179	77454	0,00 %	39,30%	36,10%
1200	252000	669	376,6816143	100	203821	0,00 %	40,80%	40,50%
2100	441000	1620	272,2222222	76	267291	0,01 %	89,10%	89,00%