Informe PPS

Universidad Nacional del Noroeste de la Provincia de Buenos Aires

Título:

Diseño, implementación y despliegue de autenticación multifactor integrado con Single Sign On utilizando protocolos y arquitectura orientada a servicios.

Carrera: Ingenieria en informatica

Práctica Profesional Supervisada

Estudiante: Ignacio Martin Citate Gomez

Tutor Docente: Carlos Andres Di Cicco

Tutor de Institución: Hugo Ramon

Fecha de presentación: Noviembre de 2023

Índice de contenidos

1. Introduccion

La UNNOBA (Universidad Nacional del Noroeste de la Provincia de Buenos Aires) provee a todos sus integrantes de múltiples soluciones de software que facilitan las actividades académicas y administrativas. En particular, la ProTIC (Prosecretaría TICs) se encarga de diseñar, implementar, gestionar y mantener dichas soluciones.

Uno de los ejes principales dentro de estas aplicaciones es la autenticación, autorización y accounting, conocido en seguridad informática como AAA.

La ProTIC resuelve esto de forma transversal utilizando un esquema llamado SSO (Single Sign On) [1], lo cual permite a los usuarios acceder a múltiples sistemas relacionados (pero independientes) sin necesidad de ingresar sus credenciales más de una vez.

A su vez, ofrece un portal llamado 'Login' que vincula al usuario con sus servicios disponibles.

Los datos relacionados a cada cuenta de usuario (credenciales, datos personales, autorización, servicios disponibles) son persistidos y accedidos mediante un protocolo llamado LDAP (Lightweight Directory Access Protocol) que provee, a efectos prácticos, una base de datos centralizada.

Actualmente, la única manera de validar la identidad de un usuario es mediante su email institucional y su contraseña, esto resulta aceptable en casos donde los usuarios hacen un uso responsable de sus credenciales, tienen contraseñas fuertes y realizan una rotación frecuente de las mismas.

Sin embargo, aun tomando estas precauciones, la seguridad de sus cuentas puede ser vulnerada mediante diversos ciberataques [2].

Dada esta situación, se propone aumentar la seguridad mediante un método conocido como MFA (Multi-Factor Authentication) [3], que requiere que el usuario utilice dos o más factores de verificación para acceder.

Adicionalmente, se busca integrar todas las interfaces y servicios internos en una única plataforma IAM (Identity and Access Management), para facilitar el mantenimiento de los sistemas, evitar lógica duplicada y reducir el acoplamiento de servicios internos, lo cual resultará en una solución más segura, robusta y estable.

2. Objetivos

Esta PPS (Práctica Profesional Supervisada) tiene como objetivo general el diseño, desarrollo, implementación y puesta en marcha de una plataforma IAM que proporcione mayor seguridad a las cuentas UNNOBA, y a su vez, centralice todas las interacciones internas entre servicios para mejorar la mantenibilidad y simplicidad de las tareas

pertinentes a la AAA. Para cumplir con dicho propósito, se establecen los siguientes objetivos específicos:

- Diseñar e implementar una plataforma IAM que siga una arquitectura SOA (Service Oriented Architecture) y permita centralizar el manejo de identidad y accesos a cuentas UNNOBA.
- Extender la actual solución de autenticación implementando un servicio independiente del protocolo actual, de modo que pueda generalizarse a diferentes esquemas.
- Diseñar, implementar y desplegar una solución de autenticación multifactor basada en OTP (One Time Password).
- Diseñar e implementar un servicio HTTP (HyperText Transfer Protocol) que sea el responsable de la interacción con LDAP, de modo que se estandarice y unifique la comunicación con la base de datos de usuarios.
- Diseñar e implementar un API (Application Programming Interface) Gateway que funcione como proxy de autenticación y punto de control, facilitando así el desarrollo de aplicaciones que se comunican con APIs internas y reduciendo la duplicación del código.
- Migrar y actualizar el servidor de LDAP a la nueva versión estable con el fin de aprovechar los cambios y mejoras recientes, además de mover la instancia a la nueva infraestructura cloud interna (OpenStack).

3. Plan de Trabajo y Carga Horaria

N°	Actividades	Tiempo de duración														
		Semanas														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	Investigación y análisis de la solución de autenticación anterior (SAML, LDAP)	•	•														
2	Diseño e implementación de API Gateway			•	•	•											
3	Migración y actualización de servidor LDAP							•	•	•							

4	Diseño e implementación de servicio HTTP de interacción con LDAP				•	•	•									
5	Diseño e implementación de nuevo servicio y aplicación web IAM que soporta MFA.								•	•	•	•	•	•	•	•
6	Elaboración de informe final										•	•	•	•	•	•

4. Descripción de la Práctica Profesional Efectuada

4.1. Estado del arte

En las grandes empresas tecnológicas se aprecian soluciones IAM extremadamente pulidas [4] (que siguen en proceso de mejora continua) en cuanto a la seguridad, usabilidad y análisis de datos. Si bien no existe un estándar único que coincida en todas estas organizaciones, si hay algunos aspectos comunes a tener en cuenta [5].

- Poseen sistemas de autenticación multifactor, estos factores pueden estar basados en datos biométricos de la persona, de posesión de hardware, o de posesión de software o una combinación de ellos. Siendo los datos biométricos la representación más fiel de la identidad del usuario hasta la fecha.

- Poseen esquemas de autenticación SSO, esto resulta primordial ya que se manejan una amplia gama de servicios en la misma organización, y resulta primordial que los usuarios se autentiquen una única vez y con las mismas credenciales para tener acceso a los mismos.

- Se mantienen al día ante las constantes nuevas vulnerabilidades que son descubiertas en el campo de la ciberseguridad; brute force, DoS (Denial of Service), DDoS (Distributed DoS), MITM, password/key compromise, session hijacking.

- Se busca tener un sistema de autorización centralizado (a nivel de acceso). Esto quiere decir que es requerido que se puedan asignar las aplicaciones a las que tiene permitido acceder a un usuario de forma centralizada, de modo que pueda ser manejado y monitoreado de forma más sencilla. Google lleva esto un poco más allá y propone un sistema de autorización centralizado a nivel global [6].

- Se implementan arquitecturas orientadas a servicios o de microservicios para una mayor separación de responsabilidades, desacoplamiento y reducción de complejidad.

- Preponderan la recolección y monitoreo de datos como base fundamental para la toma de decisiones y determinante para la evaluación del funcionamiento de la plataforma.
- Se considera la usabilidad y experiencia del usuario como primordial. En muchos casos se deja de lado la seguridad en pos de la usabilidad dado que si se compromete la usabilidad se generan pérdidas de dinero por abandono del producto o servicio en cuestión.

4.2. IAM en UNNOBA

El primer paso a realizar antes de tomar cualquier decisión fue analizar la solución de autenticación, autorización, gestión de usuarios y registro (IAM) que se encontraba en funcionamiento hasta el momento. Dicha solución de software se distingue por varias partes o módulos que se encargan de funciones específicas.

4.2.1. LDAP

El módulo fundamental se trata del almacenamiento de datos de usuario, el mismo se realiza utilizando una tecnología llamada LDAP (Lightweight Directory Access Protocol) [7]. Es un protocolo utilizado mayoritariamente para guardar datos en un directorio LDAP y autenticar a usuarios con acceso a este directorio. A su vez, provee el lenguaje de comunicación utilizado para enviar y recibir esta información de forma estándar. Una de las ventajas principales de LDAP es que funciona como punto central de acceso y modificación de datos.

Dentro del conjunto de datos de usuario que se persisten en LDAP se encuentran las credenciales (email y contraseña), lo que hace especialmente útil a este protocolo para proveer Same-Sign-On (no confundir con Single Sign-On), es decir, que los usuarios se autentican utilizando la misma contraseña en todos los sistemas, un ejemplo claro de esto es la autenticación a las redes WiFi (Wireless Fidelity) de la UNNOBA, que precisamente utiliza LDAP como agente autenticador.

Utilizar LDAP como única fuente de la verdad (término acuñado del inglés, source of truth) es muy beneficioso para los usuarios, dado que no deben recordar diferentes credenciales para diferentes sistemas, por ejemplo ingresar con una contraseña a SIU-Guarani y con otra a PlataformaED, esto sería muy engorroso, incluso si los usuarios decidieran ingresar la misma contraseña en ambos sistemas, al actualizar alguna de ellas deberían actualizar las restantes en los demás sistemas. No es muy controversial argumentar que la situación descrita previamente no ofrece una buena experiencia para el usuario (UX, del inglés User eXperience).

LDAP también ha sido aprovechado en la UNNOBA como fuente de autorización, esto es, para el manejo de los permisos de acceso a los diferentes sistemas. En LDAP además de los datos comunes al usuario (como el nombre, apellido, DNI, email y contraseña) se asocian los permisos de acceso a cada sistema ofrecido por la UNNOBA.

En resumen, la tecnología LDAP es utilizada para llevar a cabo dos grandes tareas, la persistencia de datos de autenticación y la autorización.

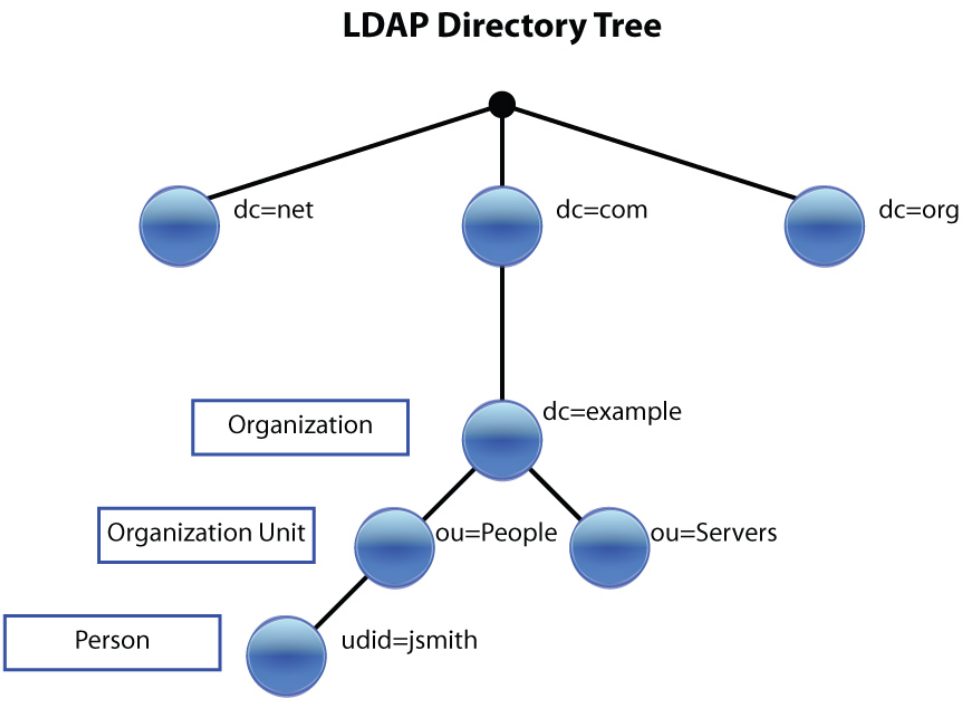

Fig 1. - Estructura de árbol LDAP.

4.2.2. SAML

Otro módulo destacado dada la importancia de su función se trata de SAML (del inglés, Security Assertion Markup Language) [8]. SAML es un estándar abierto que define el intercambio de datos en cuanto a autenticación y autorización entre partes, en particular entre lo que el protocolo define como Identity Provider (IdP) y Service Provider (SP).

En el caso de uso principal de SAML (y el utilizado por la UNNOBA), el usuario solicita un servicio a un SP (es decir, una aplicación web como puede ser SIU-Guaraní). El SP realiza una solicitud de autenticación al IdP que luego de que el usuario ingrese sus credenciales desemboca en un acceso exitoso (o no) al SP. El estándar no impone un método de autenticación específico en el IdP lo cual nos permite personalizar este comportamiento, a fin de cuentas, se utiliza LDAP como fuente de datos subyacente para autenticar y autorizar al usuario.

Al margen de los detalles inherentes al estándar SAML, lo que realmente nos interesa es su utilidad para posibilitar el esquema denominado Single Sign-On (SSO) de forma estandarizada.

SSO es una solución que permite que diferentes aplicaciones independientes (que pueden pertenecer a entidades distintas) utilicen la misma sesión de autenticación, de esta forma

evitando el ingreso repetitivo de credenciales. Las implementaciones de SSO como SAML son adoptadas frecuentemente por organizaciones a nivel global por motivos de seguridad y experiencia de usuario.

Desde el punto de vista de la seguridad, una ventaja introducida por SSO es que, dado que reduce la cantidad de credenciales necesarias para acceder a múltiples servicios a solo uno, hay menos posibilidades de que estas credenciales sean extraviadas o comprometidas.

Desde el punto de vista del usuario final, el hecho de aprovechar una misma sesión para múltiples aplicaciones mejora la experiencia del usuario porque reduce drásticamente la fatiga de ingreso de credenciales además de la carga asociada a recordar las mismas.

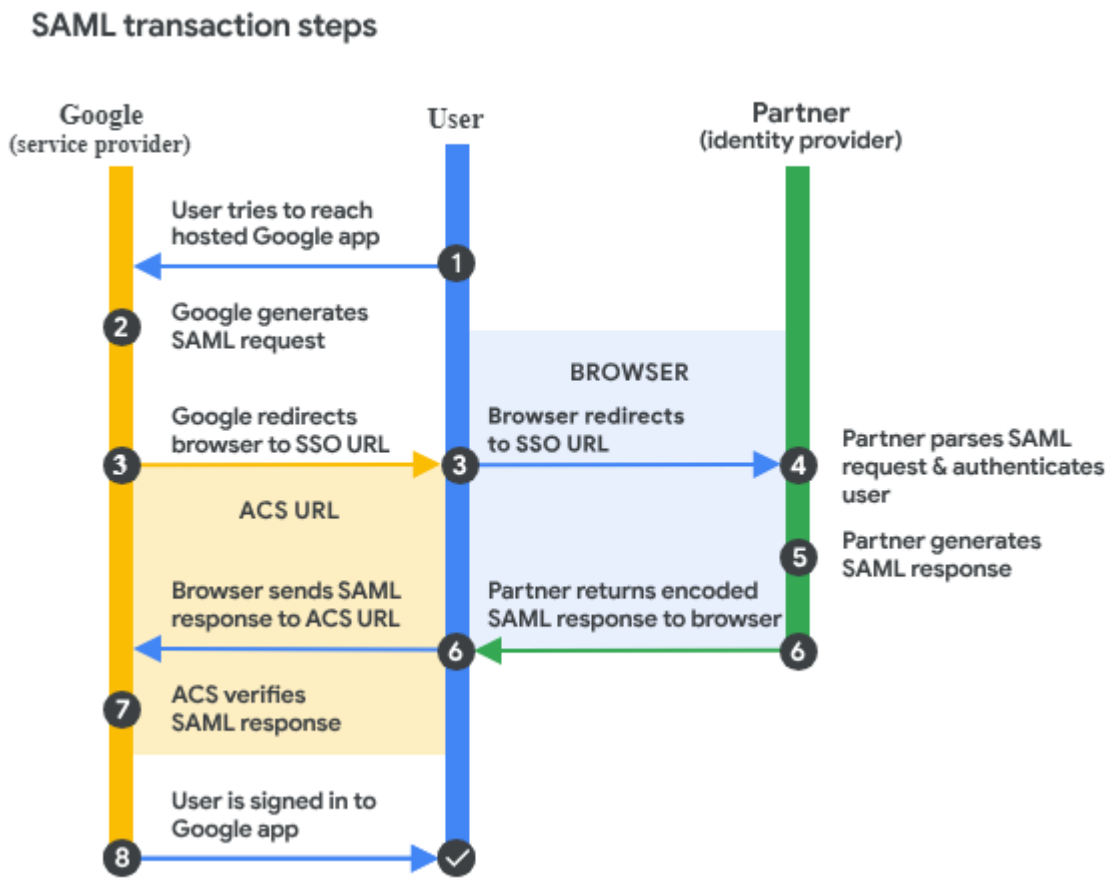

Fig 2. - Transaccion SAML.

4.2.3 Registro de acceso

Resulta sumamente importante llevar un registro de los accesos de los usuarios a los diferentes sistemas (donde, quien, como y cuando). Por diversas cuestiones como la auditoria, seguridad, análisis de datos, entre otros.

Al momento del análisis, se observa que esto se resuelve mediante un servicio HTTP que se encarga de guardar en la base de datos cada registro de acceso, junto con los datos del sistema al que ingresó. Lo positivo de esta solución es que persiste estos registros de manera centralizada en una única base de datos, en lugar de desperdigarlos por los diferentes sistemas. Sin embargo, necesitamos que cada sistema realice una llamada a este servicio cada vez que un usuario accede al sistema. Esto es manejable en el caso de los sistemas internos que son propiedad de la UNNOBA (aunque también existe la posibilidad de que se olvide su implementación en algún sistema, además de que es propenso a errores ya que hay que repetirlo en distintos lugares), dado que la ProTIC gestiona desarrollos de software propios, de terceros (en algunos casos con personalizaciones) y de código abierto, se presentan casos en donde no se tiene la posibilidad de modificar el código para registrar el acceso a estos (por ejemplo, los sistemas del SIU o Sistema de Información Universitaria, que se encarga de desarrollar software universitario a nivel nacional o Moodle, que es el software de código abierto que funciona como campus virtual).

4.3. Solución propuesta

A continuación, se describe la solución propuesta y efectuada para mejorar los aspectos mencionados anteriormente, de forma de poder avanzar hacia lo que se considera el estado del arte.

4.3.1 API Gateway (Gatekeeper)

La última versión del estándar SAML (2.0) data del año 2005, cabe imaginar que los cambios en la industria del desarrollo de software (o reduciéndose solo al desarrollo web) han sido vastísimas. Esto lleva a que muchas veces se encuentren ciertas limitaciones en el protocolo dado que el panorama en aquel entonces era radicalmente distinto.

En la UNNOBA se encontraron dificultades al tratar de integrar este protocolo al desarrollar nuevas aplicaciones web que poseen una API HTTP como Backend y una SPA (Single Page Application) desarrollada con algún framework de JavaScript en el Frontend. Estas dificultades surgen dado que SAML se basa en la interacción con el navegador web utilizando redirecciones HTTP (véase SAML Redirect Binding), pero al separar backend de frontend, no podemos depender del navegador web para realizar redirecciones. Esto significa que no podemos implementar SAML sin que nuestro backend quede expuesto a la red pública, ya que necesitamos que el servidor (que es el backend) realice la interacción con SAML de forma directa, pero esto contradice el propósito de tener el backend y frontend separados, ya que el usuario estaría interactuando directamente con el backend.

Esta fue la principal motivación que llevó a la creación de una API Gateway (llamada “Gatekeeper” por el autor).

Un API Gateway [9] actúa como mediador entre las aplicaciones cliente (frontend) y los servicios backend en una arquitectura de microservicios [10] o SOA [11]. Es una capa de

software que funciona como punto de entrada para varias APIs, lo que le permite realizar acciones que son comunes a todos los servicios como lo es la autenticación, autorización y accounting. Esto simplifica el desarrollo de nuevos servicios, ya que no necesitan rehacer la lógica que es común a todas ellas como la AAA, también permite que cada servicio haga una tarea específica sin tener que mezclar diferentes funcionalidades que no le corresponden.

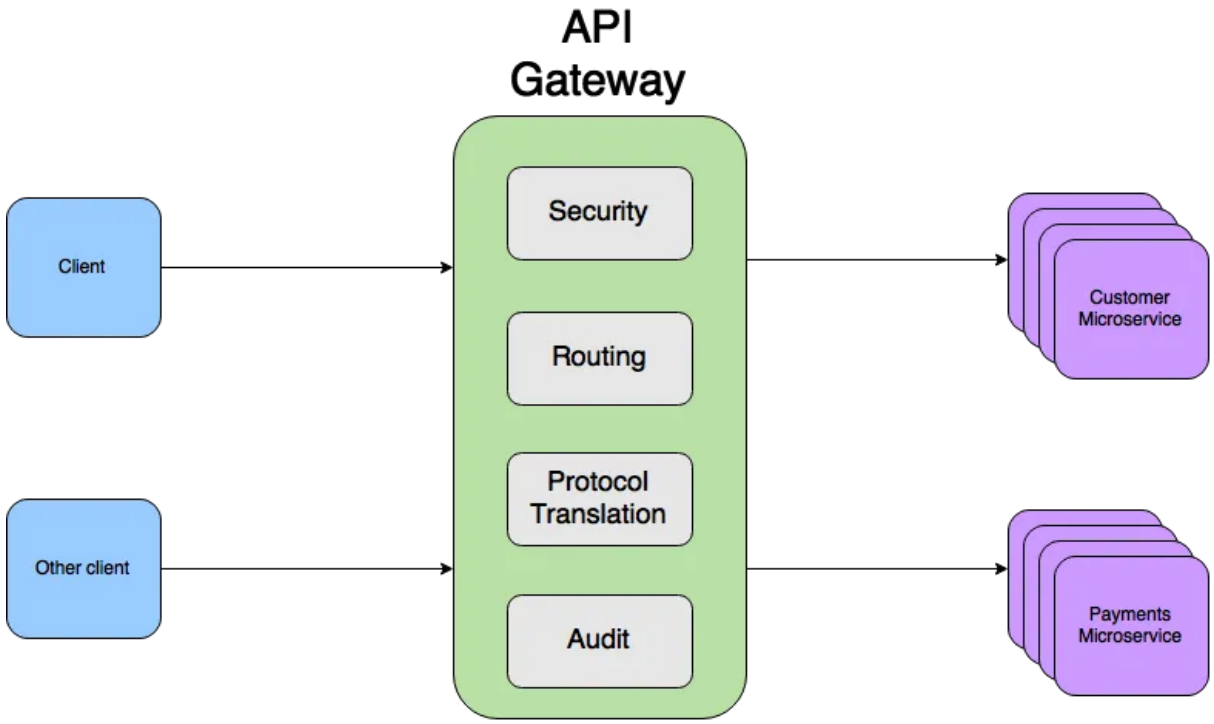

Fig 3. - Arquitectura de API Gateway.

En particular, Gatekeeper, permite configurar las rutas a cada API mediante un archivo de configuración en formato YAML, donde se asigna a cada API un prefijo. Por ejemplo, si se quisiera rutear una API HTTP que tiene como URI (Uniform Resource Identifier) base “http://apiejemplo.unnoba.edu.ar:8080” con el prefijo “/apiejemplo”, entonces todas las llamadas con dicho prefijo se redireccionará a esa API. (si se llama a “/apiejemplo/users” se traducira a “http://apiejemplo.unnoba.edu.ar:8080/apiejemplo”), cabe destacar que

Gatekeeper es un servidor HTTP que debe estar en la red pública, mientras que las APIs solo pueden accederse desde la red interna, por lo que no es posible acceder a ninguna de ellas sin pasar por Gatekeeper.

Previo al proxy de la solicitud HTTP se asegura de que el usuario se encuentra autenticado, de lo contrario se devuelve un error HTTP 401 (Unauthorized), esto es lo que le indica a nuestro frontend que debemos redireccionar al usuario al login de Gatekeeper, que a su vez es el que interactuara con el protocolo SAML. Luego de un acceso exitoso, Gatekeeper se encarga de redireccionar a la URL del frontend que inició el login.

Otro de los pasos previos al proxy de la solicitud HTTP es la inserción de un header personalizado a la request con el mail del usuario codificado para que el servicio backend pueda identificar a qué usuario corresponde la request (no existe la posibilidad de interceptar este header, ya que necesariamente la única forma de llegar a un backend es a través de Gatekeeper).

La creación de Gatekeeper permite a la UNNOBA crear aplicaciones que tienen frontend separado de backend (entre ellos “QR Generator” un sistema que sirve para tomar asistencia por QR, una aplicación de escritorio de firma digital de certificados y varias interfaces administrativas internas) además de agilizar a los desarrolladores la tarea de programar todo lo relacionado con AAA que es común a cualquier aplicación multiusuario.

4.3.2 Migración y actualización de LDAP

4.3.2.1. Motivación

Hay distintas implementaciones de LDAP, entre ellas se encuentra OpenLDAP [12], que es una implementación de código abierto ampliamente utilizada y presente en las distribución de Linux basadas en Debian, lo cual resulta particularmente conveniente para el caso de la UNNOBA, ya que sus servidores son principalmente Debian y Ubuntu.

Al momento del inicio del proyecto, el servidor de LDAP de UNNOBA se encontraba en la versión 2.4. Es decir, una versión atrás de la 2.5, que es la nueva versión LTS (Long Term Stable) que había salido hasta la fecha.

Además de ello, la ProTIC había migrado varias máquinas virtuales a una nueva infraestructura de nube interna llamada OpenStack [13], la cual facilita al equipo el mantenimiento de los servidores. Entre una de las máquinas que se encontraban pendiente de migración estaba el servidor de LDAP.

Teniendo en cuenta lo mencionado, se tomó la decisión de realizar la migración del servidor de LDAP a la par que su actualización a la versión 2.5. Además, se le dio prioridad a esto, ya que este servidor afecta directamente a todas las cuentas de la UNNOBA y por ende casi la totalidad de las interacciones que se realizan en los sistemas de la universidad.

4.3.2.2. Ejecución

Dada la naturaleza crítica del servidor de LDAP, es necesario tomar las máximas precauciones para evitar cualquier tipo de fallo, ya que implicaría una parada de la actividad en los sistemas casi por completo.

Como es de esperar, la UNNOBA posee backups realizados diariamente de las máquinas virtuales, además de entornos de test, donde se pueden realizar pruebas con respecto a los cambios realizados.

Según se investigó, la actualización de versiones de OpenLDAP era segura, ya que no influye en ningún cambio en los protocolos. La mayor dificultad recae en que existían cambios de configuración y de backend de almacenamiento que se utiliza. El backend de persistencia de datos que se utilizaba hasta la fecha quedó obsoleto en la nueva versión, por ende este debía ser reemplazado al realizar el cambio.

Esto implicaba copiar todos los datos hasta la fecha y sobreescribirlos en el nuevo servidor con la versión 2.5 instalada, además de realizar exhaustivas pruebas para comprobar el correcto funcionamiento.

También hay que tener en cuenta que la UNNOBA utiliza LDAP para asociar los cursos del sistema Moodle (o PlataformaED), con lo cual se probaron varios scripts de sincronización de cuentas existentes.

Además, la UNNOBA sincroniza las cuentas institucionales con cuentas de Google utilizando LDAP, por ende había que asegurarse que esto seguirá funcionando de manera correcta.

Otra de las cuestiones a tener en cuenta al migrar el servidor es el cambio de IP y por lo tanto de nombre de dominio (DNS).

Finalmente, el dia 20/09/2023 por la noche (donde hay menor tráfico), se realizó la restauración y copia de la base de datos junto con el cambio de DNS y migración de máquina virtual, es decir, todo lo pertinente para que el servidor nuevo de LDAP quede en funcionamiento y reemplace al viejo. Cabe destacar que el cambio fue exitoso, ya que no generó ningún error a posteriori y todo quedó funcionando de forma óptima con la última versión y en los nuevos servidores.

4.3.3. Servicio HTTP para LDAP

Con el fin de centralizar la comunicación con el servidor de LDAP, y en consecuencia con sus datos, se decidió desarrollar un microservicio interno que funciona como una API HTTP para que los sistemas que requieran consultar o modificar datos puedan hacerlo por este medio. Recordemos que la alternativa es que cada uno de los sistemas accedan directamente a los datos con el protocolo LDAP, y por lo tanto tengan las credenciales de acceso replicadas así como la implementación del protocolo (esto se traduce en la utilización de una biblioteca de código).

La alternativa propuesta ofrece mayor control sobre el flujo de datos entre sistemas, reduce la tendencia a fallos, facilita la mantenibilidad debido a la eliminación de lógica duplicada y permite monitorear el tráfico de cada servicio.

Algunas de las funcionalidades que ofrece este servicio son:

- Autenticar un usuario con sus credenciales.
- Agregar un nuevo servicio a un usuario (permisos de acceso).

- Actualizar datos personales como email alternativo y teléfono.
- Obtener datos de un usuario en particular por DNI o email.

Estos son los casos de uso esenciales y más utilizados, pero es posible extender esto a cualquier consulta, modificación o creación de datos en LDAP.

4.3.4. Plataforma IAM

4.3.4.1. SimpleSAMLphp

El primer aspecto a resolver para crear una plataforma IAM es el SSO, que como ya fue mencionado, se logra a partir del protocolo SAML. En particular, utilizando una biblioteca de código abierto, llamada SimpleSAMLphp [14], que como su nombre indica, está programada en lenguaje PHP. La misma ofrece varios módulos de autenticación posibles (auth source), que son configurables.

La configurada hasta el momento se trataba del módulo LDAP. en el que se especifican los datos correspondientes al servidor LDAP, y luego se encarga de utilizarlo como base de datos para autenticar a los usuarios. Sin embargo, esto no nos permite personalizar el proceso de autenticación a nuestro gusto, por ejemplo, para agregar un segundo factor de autenticación, o para registrar el acceso de los usuarios.

Ahondando en el código y la documentación de SimpleSAMLphp, se encontró que la misma tiene la opción de crear módulos personalizados [15] para utilizar como fuente de autenticación. Esto es muy conveniente ya que nos permite realizar cualquier proceso interno siempre y cuando obtengamos como respuesta final si la autenticación fue exitosa (o no), junto con sus datos (email, nombre, apellido, servicios permitidos) en el caso afirmativo.

Ahora bien, la idea consta en crear un modulo de SimpleSAMLphp (llamado simplesamlphp-unnoba), que se comunique con nuestra plataforma IAM por medio de HTTP por la red local (localhost) de modo que la latencia se reduzca lo máximo posible, y de esta forma autenticar a los usuarios, mostrarles la página de login, y ejecutar los procesos correspondientes.

A la par de esto, se detecto que la version de SimpleSAMLphp configurada en aquel momento era la 1.17, mientras que la versión estable más reciente era la 1.20, que según la documentación de SimpleSAMLphp, era la ultima que se planea lanzar. Además, la máquina virtual donde se encuentra alojada actualmente (openid.unnoba.edu.ar) está en la infraestructura vieja (Xen), con lo cual sería una buena oportunidad para migrarla a los nuevos servidores en OpenStack.

Como resultado, se desarrolló el modulo “simplesamlphp-unnoba” junto con la actualización a la ultima version 1.20 de SimpleSAMLphp en una maquina virtual nueva en OpenStack.

4.3.4.2. Golang

Una importante decisión a tomar, una vez que ya se conoce el dominio del problema y se tiene una idea clara de cómo se va a resolver, es la tecnología a utilizar. En especial, cual es el lenguaje de programación.

En la ProTIC se utiliza mayoritariamente el lenguaje Java, sin embargo, en el último tiempo se incursionó en el lenguaje de programación de Google, Go [16] (también conocido como Golang).

Go es un lenguaje de programación compilado y con tipado estático, diseñado en Google por Robert Griesemer, Rob Pike, y Ken Thompson. Es sintácticamente similar a C, pero también ofrece seguridad de memoria (del inglés, Memory Safety), garbage collection, tipado estructural y un modelo de concurrencia simple pero eficiente que es una de sus características más destacadas.

Además de esto, Go tuvo mucho éxito por motivos como su velocidad de compilación, velocidad de ejecución, fácil despliegue y su amplio ecosistema de herramientas y librerías. Uno de los “selling points” más destacados (inclusive por Rob Pike) es la simplicidad. Es un lenguaje simple, que tiene una baja curva de aprendizaje comparado con otros lenguajes más populares.

Otro punto importante es su librería estándar, como la mayoría de lenguajes Go tiene una librería estándar que ofrece funciones comunes a todos los lenguajes, sin embargo Go va un poco más allá y desarrolla una librería estándar más amplia que ofrece cuestiones que con otros lenguajes normalmente se deben recurrir a dependencias externas, entre ellos el paquete "net/http”, “html/template”, “crypto” y ”encoding/json”.

Una de las características más útiles (personalmente para el autor), es el manejo de dependencias, paquetes y el “build system” que ofrece Go. En primer lugar, esto ya viene predeterminado por el lenguaje, con lo cual no es necesario recurrir a herramientas externas, además es muy sencillo de utilizar y ahorra bastantes dolores de cabeza que pueden producirse en otros lenguajes como Python (véase pip, anaconda, etc) o Java (maven, gradle).

Por estos motivos (además de su éxito en la ProTIC con otros proyectos más pequeños), se decidió realizar el desarrollo de la plataforma IAM en este lenguaje.

4.3.4.3. Autenticación, 2FA y recuperación

Lo primero que se desarrolló fue la funcionalidad básica de inicio de sesión con email y contraseña (ya presente en la solución anterior). Para ello, cada vez que hay que autenticar al usuario con sus credenciales se hace uso del servicio HTTP de LDAP.

Una vez hecho esto, se trabajó en la autenticación de dos factores [17]. Se decidió utilizar como segundo factor un método llamado OTP, se trata de una clave de un solo uso (descartable), que se pide al usuario luego de que ingresa correctamente su contraseña. La capa de seguridad extra está basada en que para generar este OTP es necesario que el usuario tenga un dispositivo móvil (también es posible realizarlo con extensiones del navegador u otras aplicaciones en una PC). De todas formas, lo que se busca con esto es que la persona tenga que tener acceso a un segundo factor, en este caso el dispositivo que genera los OTPs para poder verificar su identidad.

Existen distintos tipos de OTP, en particular se eligió el TOTP (Time-based OTP), que como su nombre indica, está basado en el tiempo. Un TOTP (definido en el RFC 6238) [18] es un algoritmo que genera un OTP utilizando el tiempo actual como fuente de unicidad.

A nivel de seguridad significa que si un TOTP se compromete, sólo será válido por un tiempo limitado.

Para generar un TOTP se necesita de una clave secreta compartida entre el cliente y el servidor, con lo cual es de suma importancia la protección de esta clave secreta, ya que puede ser utilizada por un atacante, por ejemplo por que la base de datos de autenticación resulta expuesta. Una forma de prevenir esto es tratar estas claves del TOTP como si se tratara de una segunda contraseña, es decir, tomando las precauciones de encriptado pertinentes y por tanto asegurando la clave real ante una potencial exposición de la base de datos.

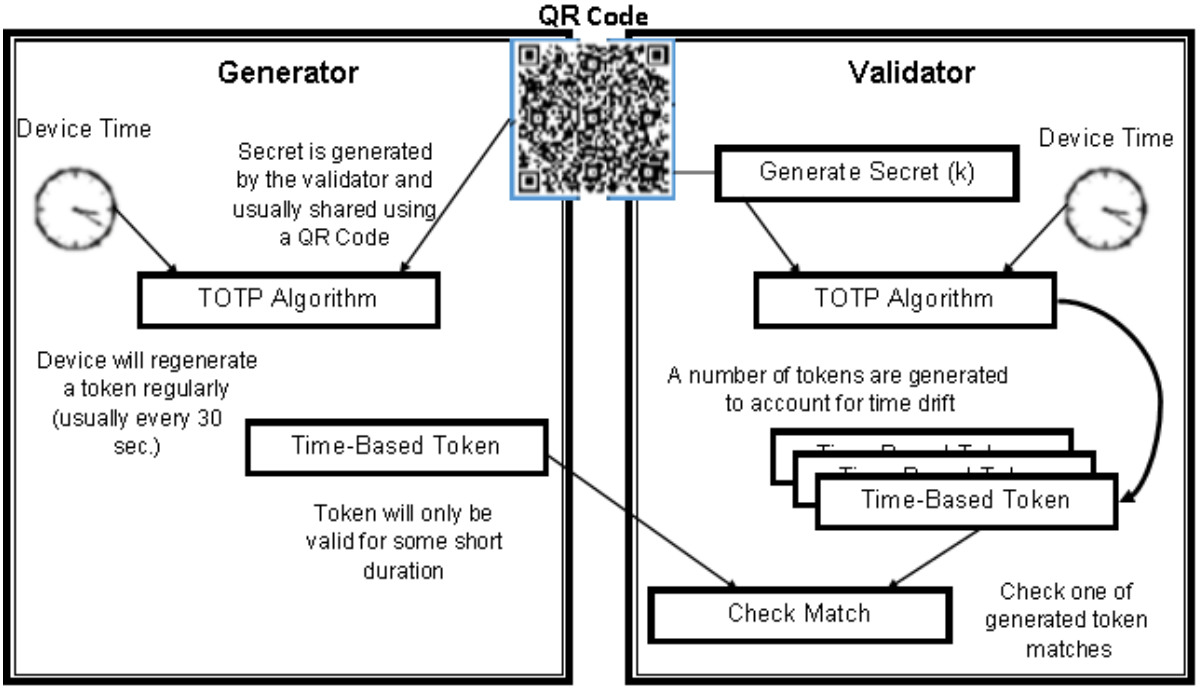

Fig 4. - Algoritmo TOTP.

El usuario debe tener la capacidad de iniciar sesión sin ningún segundo factor (si no lo desea), y en caso de que quiera habilitarlo para incrementar la seguridad de su cuenta debería poder realizar su respectiva configuración, además de poder activarla y desactivarla.

Para configurar la autenticación de dos factores el usuario debe instalar una aplicación como Google Authenticator en su dispositivo móvil, una vez hecho esto, deberá utilizar la aplicación para escanear el código QR que se muestra en pantalla (que tiene la información del TOTP) y finalmente confirmar ingresando el OTP generado en la aplicación. Una vez hecho esta serie de pasos, el usuario está en condiciones de utilizar la autenticación de dos factores en sus próximos accesos.

Ante la dependencia de un dispositivo móvil para confirmar la identidad del usuario, surge un nuevo problema. ¿Qué ocurre si el usuario extravía su dispositivo móvil? En ese caso, el usuario tendrá la posibilidad de ingresar un código de recuperación que se genera al momento de configurar la autenticación de dos factores.

Si el usuario pierde acceso a su dispositivo móvil, debe ingresar alguno de estos códigos, y tendrá la posibilidad de configurar la autenticación de dos factores, o desactivarla si así lo desea.

4.3.4.4. Post-Autenticacion

Es común que se requiera realizar ciertas acciones luego de autenticar al usuario, como por ejemplo, preguntarle si desea configurar la nueva autenticación de dos factores, actualizar sus datos personales, completar una encuesta, entre otras.

Debido a que esta necesidad surge con frecuencia en el ámbito académico, se agregó a la plataforma la posibilidad de mostrar al usuario cualquier información pertinente previo al acceso original que fue solicitado por el mismo. De todas formas, esto ocurrirá sólo en casos muy puntuales, de modo de no avasallar al usuario cada vez que necesita acceder a algún sistema.

4.3.4.5. ReCAPTCHA

ReCAPTCHA [19] es un servicio gratuito que protege los sitios web de abuso o spam, mayoritariamente causado por bots. Utiliza un avanzado motor de análisis de riesgos, junto con machine learning. En particular, la versión más reciente reCAPTCHA v3, apunta a no entorpecer la experiencia del usuario. Esta versión limita la interacción calculando una puntuación acorde al comportamiento presente e histórico del usuario. Es entonces cuando se determina mediante este puntaje si el usuario es efectivamente un humano.

En este sentido, se hace uso de reCAPTCHA v3 en la página de inicio de sesión del usuario, con el fin de evitar posibles ataques e incrementar la seguridad del sistema, (por ejemplo por pruebas de fuerza bruta con bots). Si se detecta que la entidad que está iniciando sesión no es un humano, automáticamente se denegará el acceso (potencialmente marcando la IP como sospechosa).

4.3.4.6. Registros y dispositivos de acceso

Con el objeto de tener más información sobre los accesos de los usuarios, detectar posibles accesos sospechosos e inclusive analizar el tráfico de los sistemas para tomar decisiones estratégicas, se registran los inicios de sesión (e intentos de inicio).

Un dato importante respecto a los accesos que no se posee al menos hasta el momento de esta implementación es información referente al dispositivo desde el que se accede. Entiéndase por dispositivo al conjunto formado por:

- ID del usuario.
- User-Agent (contiene información del navegador y el sistema operativo).
- Direccion IP.

El User-Agent es un header que se envía en las solicitudes HTTP.

Entonces se registra para cada acceso:

- Fecha y hora.
- Dispositivo desde el que se accede.
- Host al que intenta acceder (que identifica unívocamente al sistema).
- Si es un acceso exitoso o no.
- Método de dos factores utilizado (si aplica).

Para visualizar estos datos, se desarrolla un dashboard administrativo que muestra con gráficos y tablas los mismos para su análisis detallado.

El registro de los dispositivos de acceso permite a los usuarios detectar accesos sospechosos y bloquear el acceso desde el mismo en el caso de que se reconozca que tal dispositivo no le pertenece. Por ejemplo, ante un hackeo de cuenta, el usuario tiene la posibilidad de bloquear el dispositivo del atacante.

Aun así, se busca prevenir estos ataques con la ya mencionada autenticación de dos factores, que reduciría enormemente los riesgos de que tal cosa ocurriera.

4.3.4.7. Redis (sesiones persistentes)

Un aspecto importante de una plataforma IAM son las sesiones. Una sesión permite identificar las interacciones de un usuario con las aplicaciones a la que corresponde tal sesión. En particular, las sesiones se implementan con cookies [20]. Las cookies son pequeños archivos de texto que se guardan en el navegador. Una cookie de sesión es una cookie creada por el servidor que no puede ser otorgada a otro servidor. Se crea un “session ID” (identificador de sesión) que se genera de manera aleatoria, justamente para identificar a la sesión en cuestión y relacionarla con sus datos. Estos datos se conforman

por ejemplo por un dato identificador del usuario como su email, el tiempo de expiración de la cookie, el tiempo de creación, entre otros.

Hay diferentes maneras de persistir estas sesiones en el servidor. Una de ellas, y quizá la más sencilla es guardarlas en memoria RAM. El problema que esto acarrea es que a pesar de ser muy rápido y sencillo, es volátil y por lo tanto, las sesiones no sobreviven a un reinicio del servidor.

Para lograr que las sesiones sean persistentes ante estos reinicios utilizaremos Redis [21]. Se trata de una base de datos en memoria open source, que ofrece tiempos de respuesta rápidos. La particularidad que la distingue es que ofrece persistencia en disco con snapshots realizadas periódicamente para luego poder recuperarse ante una potencial caída.

Redis tiene una interfaz de comandos que permite escribir y leer datos basados en un par clave-valor. Para guardar las sesiones se utilizará el ID de sesión como clave y los datos propios de la sesión como valor.

4.4. Trabajo futuro

Es evidente que no es posible abarcar todas las funcionalidades deseadas en un único proyecto dado que los recursos temporales y humanos casi siempre resultan escasos. Por ello, se listan algunas de las cosas que pueden realizarse a futuro para mejorar las cuestiones abarcadas por el proyecto para alcanzar algo semejante a lo mencionado en el análisis del estado del arte.

- Implementar nuevos factores de autenticación, como por ejemplo WebAuthn [22], que en este momento aún se encuentra en una etapa prematura de su desarrollo, por lo que se decidió postergarlo.
- Agregar más gráficos/estadísticas relacionadas con el registro de acceso de los usuarios a medida que vayan surgiendo las necesidades de análisis.
- Crear SDKs (Software Development Kit) o bibliotecas de código internas con el fin de simplificar aún más la integración de la autenticación para diferentes lenguajes como Java, Go y Javascript.
- Aplicar más medidas de seguridad, estando atento a las nuevas vulnerabilidades o posibles ataques descubiertos (por ejemplo por fundaciones como OWASP [23]).

5. Conclusiones

Como se ha mostrado a lo largo de este informe, hemos logrado explicar cómo se afrontaron y finalmente alcanzaron los objetivos planteados para esta práctica profesional.

Se logró migrar y actualizar el servidor de LDAP de forma exitosa, acomodándose a la nueva infraestructura de OpenStack y aprovechando las mejoras de la nueva versión.

A su vez, se puso en funcionamiento el API gateway “Gatekeeper” que efectivamente redujo el costo y tiempo de desarrollo de nuevas aplicaciones.

Gracias a la implementación del nuevo servicio HTTP que interacciona con LDAP, se pudo focalizar el flujo sobre el mismo, de modo que todas las aplicaciones que lo consumen pasen por un único punto de acceso, sin tener que repetir funcionalidades en múltiples bases de código.

Uno de los consumidores de este servicio es la propia plataforma IAM, que ofrece una solución unificada para el manejo de identidad y acceso a las cuentas UNNOBA. Dicha plataforma, otorga una serie de mejoras en cuanto a seguridad, auditoría y rendimiento respecto a la anterior. Las mismas se alcanzaron a través de la posibilidad de agregar autenticación multifactor, recuperación de cuentas y registros de acceso. Incluso se aportaron mejoras de mantenibilidad que favorecen a los programadores de estos sistemas, por ejemplo con la post-autenticación, que permite tener todas las acciones a realizar luego de una acción exitosa en el punto central de la misma, en lugar de tenerla desperdigada en los diferentes sistemas.

En definitiva, la ejecución de esta práctica profesional resultó en un aporte positivo para la ProTIC y consecuentemente para la UNNOBA. Con el deseo de que pueda extenderse para mejorar la seguridad de las cuentas, facilitar el desarrollo a los programadores y analizar el tráfico en profundidad para tomar decisiones.

6. Bibliografia

1. De Clercq, J. (2002, September). Single sign-on architectures. In *International Conference on Infrastructure Security* (pp. 40-58). Berlin, Heidelberg: Springer Berlin Heidelberg.
2. Thomas, K., Li, F., Zand, A., Barrett, J., Ranieri, J., Invernizzi, L., ... & Bursztein, E. (2017, October). Data breaches, phishing, or malware? Understanding the risks of stolen credentials. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security* (pp. 1421-1434).
3. Abhishek, K., Roshan, S., Kumar, P., & Ranjan, R. (2013). A comprehensive study on multifactor authentication schemes. In *Advances in Computing and Information Technology: Proceedings of the Second International Conference on Advances in Computing and Information Technology (ACITY) July 13-15, 2012, Chennai, India-Volume 2* (pp. 561-568). Springer Berlin Heidelberg.
4. Indu, I., Anand, P. R., & Bhaskar, V. (2018). Identity and access management in cloud environment: Mechanisms and challenges. *Engineering science and technology, an international journal*, *21*(4), 574-588.

5. Konstantinidis, G. (2021). Identity and access management for e-government services in the European Union–state of the art review.
6. Pang, Ruoming, et al. "Zanzibar:{Google's} Consistent, Global Authorization System." *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. 2019.
7. RFC 4511: Lightweight Directory Access Protocol (LDAP): the protocol. (2006, June 8). IETF Datatracker. https://datatracker.ietf.org/doc/html/rfc4511.
8. RFC 7522: Security Assertion Markup Language (SAML) 2.0 Profile for OAUth 2.0 Client Authentication and Authorization grants. (2015, May 19). IETF Datatracker. https://datatracker.ietf.org/doc/html/rfc7522
9. Sanjay Gadge, Vijaya Kotwani (2017). Microservice Architecture: API Gateway Considerations https://www.globallogic.com/wp-content/uploads/2017/08/Microservice-Architecture-API-Gateway-Considerations.pdf
10. What are microservices? (n.d.). Microservices.io. https://microservices.io/index.html
11. What is SOA? - SOA Architecture Explained - AWS. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/what-is/service-oriented-architecture/
12. OpenLDAP Software 2.5 Administrator's Guide: Introduction to OpenLDAP Directory Services. (n.d.). https://www.openldap.org/devel/admin/intro.html
13. Open source cloud computing platform software - OpenStack. (n.d.). OpenStack. https://www.openstack.org/software/
14. SimpleSAMLphp home - SimpleSAMLphp. (n.d.). https://simplesamlphp.org/
15. SimpleSAMLphp documentation. (n.d.). https://simplesamlphp.org/docs/stable/simplesamlphp-modules.html
16. The Go programming language. (n.d.). https://go.dev/
17. Multifactor Authentication - OWASP Cheat Sheet Series. (n.d.). https://cheatsheetseries.owasp.org/cheatsheets/Multifactor_Authentication_Cheat_Sheet.html
18. RFC 6238: TOTP: Time-Based One-Time Password Algorithm. (2011, May 13). IETF Datatracker. https://datatracker.ietf.org/doc/html/rfc6238
19. ReCAPTCHA. (n.d.). reCAPTCHA. https://www.google.com/recaptcha/about/
20. IBM documentation. (2021, March 6). https://www.ibm.com/docs/en/sva/9.0?topic=cookies-session-concepts
21. Redis. (n.d.). Redis. https://redis.io/
22. A demonstration of the WebAuthn specification. (n.d.). WebAuthn.io. https://webauthn.io/
23. OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation. (n.d.). https://owasp.org/

7. Acrónimos

Acrónimo	Descripción
UNNOBA	Universidad Nacional del Noroeste de la Provincia de Buenos Aires
ProTIC	Prosecretaría TICs
AAA	Autenticación, autorización y accounting
SSO	Single Sign On
LDAP	Lightweight Directory Access Protocol
MFA	Multi-Factor Authentication
IAM	Identity and Access Management
PPS	Práctica Profesional Supervisada
SOA	Service Oriented Architecture
OTP	One Time Password
API	Application Programming Interface
HTTP	HyperText Transfer Protocol
API	Application Programming Interface
MITM	Man In The Middle
DoS	Denial of Service
DDoS	Distributed Denial of Service

UX	User eXperience
DNI	Documento Nacional de Identidad
SAML	Security Assertion Markup Language
IdP	Identity Provider
SP	Service Provider
YAML	YAML Ain't Markup Language
URI	Uniform Resource Identifier
DNS	Domain Name System
PC	Personal Computer
TOTP	Time Based OTP
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
ID	Identificador
RAM	Random Access Memory
WiFi	Wireless Fidelity

8. Agradecimientos

Agradezco profundamente a:

- Mi familia por apoyarme incondicionalmente en todo el trayecto universitario y en general en todos mis emprendimientos personales y profesionales.
- Mis amigos, que siempre se encuentran presentes.
- Mis compañeros de trabajo y mentores de la UNNOBA por el enorme aprendizaje realizado y el espléndido ambiente laboral que se vive diariamente.
- Mis docentes, enfáticamente con los cuales he logrado mantener una relación más cercana, por sus enseñanzas y las oportunidades ofrecidas.
- La UNNOBA por ofrecerme una posibilidad de crecimiento que espero haber aprovechado al máximo.