Desarrollo de un sistema de verificación de identidad manual, offline y diferida basada en autenticación fuerte para la Universidad de Río Negro

Trabajo final de carrera

Licenciatura en Sistemas

Autor: Dirazar, Federico

Director: Mg. Lugani, Carlos Fabián

Agradecimientos

En primer lugar, a Sandra y Delio por todo su apoyo a lo largo de mi carrera universitaria.

En segundo lugar, a Zaida por alentarme, inspirarme a lograr metas más grandes y sus valiosos aportes en materia de traducción.

En tercer lugar, a Carlos Lugani por su incansable predisposición a contribuir en la concreción de este trabajo.

Por último, al LIA que siempre me recibió con puertas abiertas para resolver consultas técnicas.

Índice

Agradecimientos **1**
Índice **2**
Resumen **3**
Abstract **4**
Introducción **5**
Problema a resolver y justificación **7**
Solución propuesta **9**
Metodología de trabajo 9
Obtención de requerimientos 12
Desarrollo 25
Descripción de la solución 28
Configuración 36
Disponibilidad de fuentes 39
Conclusiones **40**
Líneas futuras de trabajo 40
Bibliografia **42**

Resumen

Este trabajo detalla el proceso de desarrollo de una solución compuesta por una aplicación web y una aplicación móvil suplementaria, basado en el artículo *Proceso de registro e identificación sin control automático para la Universidad Nacional de Río Negro* de (Lugani, 2022).

La solución se basa en una aplicación Android generadora de tokens que identifican personas mediante el uso de su huella dactilar y que previamente deben haber realizado un registro presencial donde presentan su DNI.

Dicho registro se efectúa en la aplicación web, junto con la verificación del token que se realiza bajo demanda por un usuario autorizado y no requiere de conexión a internet; la suspensión de personas que inhabilita los tokens generados por una persona específica en un período de tiempo determinado y el alta de nuevos usuarios autorizados que utilizan la aplicación web.

Este enfoque busca resolver algunas problemáticas específicas producto del incremento en ofertas educativas a distancia de nivel universitario y el teletrabajo por parte de personal universitario, como la contabilización de asistencia en una videollamada, votaciones mediadas por videollamadas, verificación de identidad del estudiante a punto de rendir un examen a distancia e incluso posibilita un método para verificar solicitudes de consulta o modificación de información que se realizan presencialmente en oficinas de las instituciones universitarias, en especial en casos en donde no se requiere de realizar la autenticación en el momento de la identificación y la misma se puede realizar en diferido en otro momento.

La solución propuesta cuenta con una arquitectura basada en una API REST, que presta los servicios necesarios a ambas aplicaciones y gestiona mediante Keycloak los roles de los usuarios autorizados que utilizan la aplicación web.

Palabras clave: verificación de identidad, educación a distancia, teletrabajo.

Abstract

The current work details the development process of a solution consisting of a web application and a companion mobile app, based on the research paper *Proceso de registro e identificación sin control automático para la Universidad Nacional de Río Negro* by (Lugani, 2022).

The solution is based on a token-generating Android application. Users must have completed an in-person registration in which they show their ID to enable token generation. These tokens are backed by the user's fingerprint.

This registration is performed in the web application, along with on-demand offline token verification by an authorized user, suspensions which reject tokens generated by a specific user in a set timeframe and new authorized web application users.

This approach aims to solve some specific issues due to the rise in university-level distance learning offers and remote work by university staff, such as taking roll call and voting via video conferences; verifying students' identities when they're sitting for a synchronous remote exam; and even enables a way to verify student requests regarding query or updates to student information which is normally carried out in person on university premises, especially in cases where authentication is not required at the time of identification and it can be done deferred at another time.

This solution's architecture is based on a REST API which provides services to both the web and mobile apps, and manages Keycloak's authorized user roles who are allowed to utilize the web application.

Keywords: identity verification, distance learning, remote work.

Introducción

Hoy en día, cada vez más actividades se realizan de manera virtual, lo que implica desafíos abordados por la Gestión de Identidad y Acceso (IAM, por sus siglas en inglés de Identity and Access Management). Este trabajo se limita a la gestión de identidad, puesto que describe el desarrollo del sistema de software planteado en *Proceso de registro e identificación sin control automático para la Universidad Nacional de Río Negro* de (Lugani, 2022).

Cloudflare (s.f.-b) propone que “La gestión de identidad y acceso (abreviado: IAM o IdAM) es un modo de saber quién es un usuario y qué tiene permiso para hacer”. Además, Cloudflare (s.f.-a) define la autenticación como “[...] el proceso de verificar la identidad de alguien o algo”.

A su vez, Cloudflare (s.f.-a) manifiesta que “Las características que un sistema de autenticación comprobará se denominan ‘factores’”, y luego establece los tres factores más utilizados, (a) algo que sabe la persona, (b) algo que tiene la persona, y (c) algo que la persona es. Concretamente, algo que sabe la persona es “una información secreta que solo debe tener la persona real”, algo que tiene la persona “comprueba si la persona posee un objeto físico que se le ha entregado o que se sabe que tiene” y “algo que la persona es” se refiere al factor que “evalúa las cualidades inherentes a una persona” según Cloudflare (s.f.-a).

Según Pearson IT Certification (2011), un ejemplo de algo que sabe la persona es el uso de contraseñas, algo que tiene la persona se refiere a tokens físicos o una tarjeta inteligente, mientras que algo que la persona es consiste del escaneo de huellas dactilares; la geometría de las manos; escaneo del iris o retina; caligrafía y análisis de voz.

Sobre el factor “algo que tiene la persona”, hay dos distinciones, tokens blandos y tokens duros. Los tokens blandos se envían a un dispositivo como un celular mediante mensaje de texto o una aplicación específica, mientras que los tokens duros son objetos físicos pequeños que se deben conectar al dispositivo mediante USB, bluetooth u otro puerto para realizar la autenticación (Cloudflare, s.f.-a).

Los riesgos asociados a algo que sabe la persona son: el olvido de la contraseña, la escritura con posterior extravío o el hecho de que sea adivinada por un atacante. Por otro

lado, algo que tiene la persona sufre riesgo de extravío o de intercepción. Con respecto a algo que la persona es, el riesgo de falsificación depende del tipo de biometría utilizada y puede depender también de la precisión del escáner utilizado.

Por lo tanto, se asignan los distintos niveles de fortaleza de los tres factores, algo que sabe la persona, algo que tiene la persona y algo que la persona es, como débil, medio y fuerte, respectivamente.

El sistema propuesto por Lugani (2022) consta de dos aplicaciones, una web para la gestión de personas y otra móvil para la generación de tokens numéricos similares a las aplicaciones TOTP[1] como Google Authenticator, Microsoft Authenticator y Twilio Authy Authenticator, entre otras. Las principales características de la verificación de los tokens es que se realiza de manera manual, no en línea, no automáticamente y no requiere de interfaz con otros sistemas (Lugani, 2022, p. 3).

La comprobación de tokens se realiza manualmente porque está a cargo de usuarios autorizados que utilizan la aplicación web; no se realiza en línea debido a que el algoritmo se basa en la sincronización de relojes; no se realiza automáticamente porque al ser manual, se realiza bajo demanda; y no requiere de interfaz con otros sistemas porque la solución abarca todo el proceso, desde el registro y la generación de tokens, hasta la verificación de los mismos.

Una ventaja que brinda la verificación manual de los tokens es que se pueden utilizar los servicios de verificación de identidad sin la necesidad de modificar aplicaciones existentes, facilitando su adopción y ahorrando costos de desarrollo. Es más, se pueden utilizar los tokens en aplicaciones de terceros donde no se posea acceso al código fuente.

La aplicación móvil debe instalarse en un dispositivo capaz de desbloquearse con autenticación biométrica, por ejemplo, con un lector de huella dactilar o reconocimiento facial, ya que la aplicación móvil “requiere autenticación fuerte para habilitar la entrega del token” (Lugani, 2022, pp. 4-5). De este modo, la integridad de los tokens se basa en autenticación fuerte, asegurando que los tokens generados en un teléfono celular hayan sido solicitados por la persona en cuestión.

[1] TOTP: acrónimo en inglés de "Time-Based One-Time Password" (Contraseña de un solo uso Basada en el Tiempo). Es un mecanismo de autenticación que genera contraseñas de un solo uso, que son solo válidas por un tiempo, generalmente un minuto. A menudo se utilizan como segundo factor de autenticación para fortalecer la seguridad de sistemas informáticos y aplicaciones.

Problema a resolver y justificación

En Argentina, al año 2021, había un total de 793 ofertas educativas de pregrado, grado y posgrado a distancia, sin diferenciar entre gestión estatal o privada (Departamento de Información Universitaria [DIU], 2022, p. 48). En ese mismo año, se registraron 235 565 estudiantes, 99 160 nuevos inscriptos y 19 826 egresados para ofertas de pregrado, grado y posgrado en modalidad a distancia (DIU, 2022, p. 52).

Por otro lado, según datos del DIU (2022), en el año 2022, 216.279 personas estaban empleadas en el sistema universitario argentino en concepto de docente, preuniversitario, autoridad superior o no docente (p. 62).

Un desafío para las universidades es la correcta identificación de las personas intervinientes, tanto en el caso de estudiantes, como en el caso del personal de la institución. Independientemente, al tratarse de teletrabajo o de educación a distancia, la necesidad es la misma, contar con un método de verificación de identidad.
El sistema formulado por Lugani (2022), presenta una alternativa viable para este propósito, sin demandar costos de desarrollo y aprovechando software en uso por la institución.

Por ejemplo, tomando el caso de una aplicación de videollamadas, se pueden ingresar los tokens en tiempo real mediante el chat para comprobar la identidad de los usuarios conectados. Específicamente este escenario toma relevancia en reuniones virtuales donde autoridades de la universidad se encuentran en una votación, o en el caso de estudiantes, para la toma de asistencia. También resulta propicio su uso con el fin de verificar la identidad de estudiantes al momento de rendir un examen a distancia, donde usualmente se observa el DNI en cámara.

De igual manera, se pueden utilizar los tokens para identificarse en una aplicación de chat, en el caso de que se requiera gestionar algún trámite administrativo por parte del estudiante que contacta a soporte. Esto facilita el trabajo del personal administrativo, asegurando que la solicitud es auténtica y permite diseñar esquemas donde trámites presenciales puedan realizarse a distancia. Para ilustrar, la solicitud de un certificado de alumno regular o de un certificado analítico debe efectuarse presencialmente, presentando el DNI en el departamento de Alumnos. Sin embargo, se podría simplificar y agilizar esta gestión, ya que la generación de tokens siempre requiere el registro presencial previo donde se observa el DNI. De este modo, solo con una visita presencial al momento de registro, se habilita la

gestión a distancia. Este método de verificación mediante tokens cobra mayor relevancia si las solicitudes se refieren a modificación de datos de un alumno, en vez de una simple consulta.

Asimismo, en el caso de una clase o videoconferencia híbrida, donde se conectan usuarios presencialmente y virtualmente, o virtualmente dos grupos de personas reunidas en distintas sedes de la universidad, con tan solo registrar los tokens de los presentes, se contabiliza fácilmente la asistencia de todos los participantes.

Solución propuesta

Metodología de trabajo

Una de las decisiones a tomar que más impactan el desarrollo del proyecto es la elección de un modelo de Ciclo de Vida del Desarrollo de Software (SDLC, por sus siglas en inglés de Software Development Life Cycle). Sommerville define un SDLC como un conjunto de actividades relacionadas que culmina en la producción de un sistema de software (2016, p. 44).

Dos de los modelos de ciclos de vida descritos por Sommerville (2016) son el modelo en cascada y el modelo iterativo. El modelo en cascada separa las actividades de especificación, desarrollo, validación y evolución en fases distintas, llevadas a cabo secuencialmente. Por su parte, el modelo iterativo entrelaza las actividades de especificación, desarrollo y validación, que producen versiones llamadas incrementos, añadiendo nuevas funcionalidades a la versión anterior. Se basa en la idea de desarrollar una implementación inicial, obtener retroalimentación del cliente y/o de los usuarios y crear varias versiones hasta obtener el sistema que se necesita (pp. 45-49).

Sommerville (2016), afirma que como las empresas de hoy en día operan en un entorno global que sufre de interminables cambios, surgieron las llamadas metodologías ágiles, que se caracterizan por entrelazar las actividades de especificación, diseño e implementación; producir mínima documentación de diseño; desarrollar el sistema en incrementos; utilizar comunicación informal; y por un amplio uso de herramientas que automatizan el proceso, como herramientas de testing automático o gestores de configuración (Sommerville, 2016, pp. 73-74).

Un postulado del manifiesto ágil indica “Individuos e interacciones sobre procesos y herramientas” (Beck et al., 2001). Respecto a esta frase, se toma la noción de libertad en la comunicación entre las partes interesadas por sobre procesos de comunicación prescriptivos. Por lo tanto, se buscó la mejor manera de avanzar en incrementos, priorizar requerimientos según dependencias, mantener una comunicación constante entre las partes interesadas y responder a posibles cambios de enfoque, considerando que se contó con un único desarrollador. Luego de revisar la bibliografía referida a los SDLCs, se optó por la metodología ágil de gestión de proyectos Kanban.

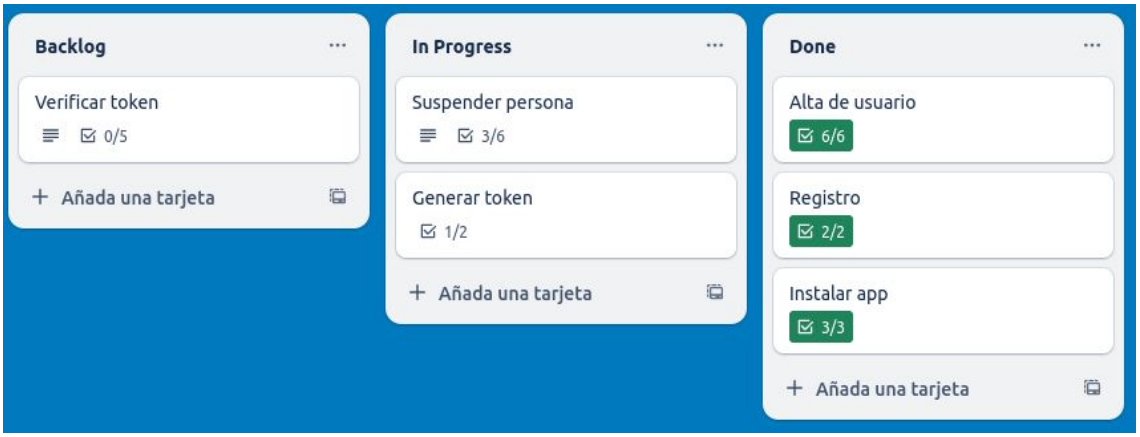
Kanban es un sistema de señalización basado en tarjetas desarrollado por Toyota para coordinar la reposición de partes en sus líneas de ensamble (Rasmusson, 2010, p. 174). En Kanban, el trabajo se limita por un concepto llamado Trabajo En Curso (WIP, por sus siglas en inglés de Work In Progress), lo que define que un equipo solo tiene permitido trabajar en un número finito de cosas a la vez.

En el caso de este proyecto, el equipo de trabajo estuvo compuesto de un solo desarrollador, por lo que el WIP se asumió de 3, principalmente porque hay dependencia entre los casos de uso identificados. Esto permitió utilizar los avances en la definición e implementación de un caso de uso para guiar el desarrollo de otros casos de uso relacionados. Se tomó el orden en que los casos de uso ocurren en el sistema para luego permitir probar nuevas funcionalidades incrementalmente.

La herramienta de gestión utilizada fue Trello (https://trello.com/), un tablero Kanban colaborativo online.

Figura 1

Tablero Trello utilizado



Como se ve en la Figura 1, las tarjetas utilizadas llevan los nombres de los casos de uso definidos previamente, y se agrupan en tres columnas, “Backlog”, donde residen tarjetas que están pendientes; “In Progress”, donde se agrupan tarjetas en progreso y “Done”, donde están comprendidas las tarjetas finalizadas. También cabe destacar que en la columna “In Progress”, hay dos tarjetas, cuya cantidad no excede el WIP de 3. Por otro lado, en la columna “Backlog” se ubica la tarjeta “Verificar Token”, que requiere de la finalización de las tarjetas en la columna “In Progress” para avanzar.

Figura 2

Trello: contenido de la tarjeta "Verificar token"

En la Figura 2 se observa un ejemplo de la información que contiene la tarjeta "Verificar token", que incluye una división del caso de uso en tareas más pequeñas.

Sobre los beneficios de utilizar la metodología Kanban, Rasmusson indica que como el objetivo de este método es el que el trabajo fluya, no es necesario preocuparse por iteraciones y se pueden abordar tareas complejas que tomarían más de una iteración en otras metodologías (2010, pp. 175-176).

Se utilizaron correos electrónicos y mensajería instantánea para mantener canales abiertos de comunicación con el propietario del sistema. Sin embargo, se llevaron a cabo reuniones presenciales para discutir avances, priorizar requerimientos y esclarecer la visión del proyecto. Además, se usaron Git y GitLab particularmente para la administración online de

los distintos repositorios privados de código. Por otro lado, los diagramas presentados se confeccionaron mediante diagrams.net, una aplicación online de diagramado.

Obtención de requerimientos

En (Lugani, 2022) se propuso una aplicación móvil que genera tokens, la aplicación central que para comprobarlos y las características de la verificación (manual, offline, diferida). Cabe destacar que la primera actividad del proyecto se basó en refinar lo estipulado mediante un proceso de obtención de requerimientos.

Sommerville (2016, p. 102) define los requerimientos como descripciones de lo que el sistema debe hacer: el servicio que ofrece y las restricciones en su operación.

Según Pressman y Maxim (2020, p. 24), el objetivo de la obtención de requerimientos es entender lo que las partes interesadas quieren del software que se va a construir.

Sommerville (2016) propone dos métodos para llevar a cabo el relevamiento, las entrevistas y la observación. Particularmente el autor define dos tipos de entrevistas, las abiertas y las cerradas. Las entrevistas cerradas consisten en preparar de antemano el cuestionario, mientras que en una entrevista abierta la conversación es sumamente casual, sin guión. En este trabajo se utilizaron entrevistas abiertas, porque no existe un sistema que se pueda observar.

Se mantuvieron varias entrevistas con el propietario del sistema. Se tomaron notas mediante un procesador de textos en una computadora personal.

En esas entrevistas se abordaron las siguientes preguntas

1. ¿Cuáles son las operaciones que permite el sistema?
2. ¿Quiénes son los actores que utilizan el sistema?
3. ¿De cuántas aplicaciones requiere el sistema y en qué plataformas se necesitan?
4. ¿Qué datos de entrada tiene el algoritmo que genera el token?
5. ¿Qué estados se deben mantener sobre los usuarios del sistema?
6. ¿Cómo es el proceso de registro?
7. ¿Cómo es el proceso de instalación de la aplicación móvil?
8. ¿Cómo se pide el token?

9. ¿Cómo se verifica el token?

A continuación se detalla la información recabada en las entrevistas:

Las operaciones que permite el sistema son: alta de usuarios, registro de una persona, generación de token, comprobación de token y suspensión de una persona.

Los actores que utilizan el sistema se dividen en usuarios y personas. Las personas utilizan el servicio, registrándose y generando tokens para su identificación en otros sistemas. Por otro lado, los usuarios tienen los roles de Administrador, Autoridad o Consulta. Los usuarios Administrador pueden agregar nuevos usuarios Autoridad o Consulta, los usuarios Autoridad están a cargo del registro y las suspensiones de las personas, mientras que los usuarios Consulta verifican los tokens.

El sistema requiere de dos aplicaciones, una web y otra móvil. La aplicación web es utilizada por los usuarios Administrador, Autoridad y Consulta. La aplicación móvil la emplea cada persona registrada para generar sus tokens.

El proceso de registro consiste en que una persona se presenta presencialmente en la oficina donde se encuentra el usuario Autoridad. El usuario Autoridad ingresa el nombre, apellido, fecha de nacimiento y número de DNI de la persona. Acto seguido, le toma una fotografía del DNI y luego toma otra del rostro de la persona. Por último, la persona ingresa una contraseña y el usuario Autoridad presiona el botón de “Aceptar”.

La persona instala la aplicación en su dispositivo móvil, luego ingresa su DNI y la contraseña que escribió en la oficina del usuario Autoridad. Por último, confirma la operación mediante su huella dactilar.

Finalizado el registro en la aplicación móvil de manera exitosa, se puede ingresar la huella para pedir un token. Al cabo de un minuto, se solicita la huella nuevamente. También se requiere la huella nuevamente cuando se desbloquea el celular luego de apagada la pantalla.

El algoritmo que genera el token requiere el DNI de la persona, la fecha, la hora, y una semilla. Una semilla es un número secreto y único para cada instalación de la aplicación móvil, obtenido cuando se instala la aplicación por primera vez en su celular.
Las personas pueden tener como estado ‘dado de alta’ y ‘suspendido’.

Por su parte, el usuario Consulta ingresa en la aplicación web un token, una fecha, una hora, nombre y apellido de la persona a verificar. Si es válido, se observa en pantalla el DNI del usuario con un mensaje de éxito. En caso contrario, presenta que el token no es válido.

Una vez finalizadas las entrevistas, se recopiló la información obtenida y se elaboraron casos de uso para documentar los requerimientos del sistema. Como resultado de esta actividad se obtuvo una especificación sobre el sistema a desarrollar.

Particularmente se describieron 6 casos de uso, (a) registro, (b) instalar app, (c) suspender persona, (d) alta de usuario, (e) generar token y (f) verificar token. Además, se identificaron 4 actores: Administrador, Autoridad, Consulta y Persona. Dichos casos de uso presentan una visión global del sistema, detallada pero no exhaustiva, en la cual no hubo grandes divergencias respecto a los lineamientos de diseño planteados por (Lugani, 2022), que funcionaron como cimiento para el desarrollo inicial.

Sobre los casos de uso, Sommerville (2016) comenta:

> Los casos de uso son una manera de describir las interacciones entre usuarios y un sistema usando un modelo gráfico y texto estructurado [...] en su forma más simple, un caso de uso identifica a los actores involucrados en una interacción y nombra el tipo de interacción. Luego se añade información adicional describiendo la interacción con el sistema (p. 125)

Tabla 1

Plantilla de casos de uso

Actor principal	
Actores secundarios	
Objetivo	Acción que soporta el caso de uso
Precondiciones	Condiciones que se asumen o deben cumplirse para lograr el escenario de éxito del caso de uso
Disparador	Evento que inicia el caso de uso
Escenario de éxito	Los pasos que componen el caso de uso. 1. Paso 1 2. Paso 2 3. Paso 3
Excepciones	Errores o acciones del usuario, asociadas a un paso del escenario de éxito, que divergen del orden establecido. 1. Error en el paso 1 2. Acción del usuario distinta al paso 2
Observaciones	1. Observación 1 2. Observación 2

Nota. Adaptado de *Software Engineering: A Practitioner's Approach (9na edición)* (p. 136), por R. S. Pressman y B. R. Maxim, 2020, McGraw-Hill Education.

En la Tabla 1 se observa la plantilla utilizada para la documentación de los casos de uso.

Tabla 2

Caso de uso: registro

Actor principal	Autoridad
Actores secundarios	Persona
Objetivo	Registrar una Persona en el sistema
Precondiciones	1. El usuario Autoridad está registrado en el sistema. 2. El usuario Autoridad mantiene una sesión activa en el sistema. 3. La Persona está presente físicamente y tiene su DNI en mano.
Disparador	La Persona se acerca a las instalaciones y solicita el registro en el sistema.
Escenario de éxito	1. El usuario Autoridad ingresa en el sistema el nombre, apellido, fecha de nacimiento y número de DNI de la Persona. 2. El sistema muestra el campo “foto persona” donde el usuario Autoridad selecciona y sube una fotografía del rostro de la persona. 3. El sistema muestra el campo “foto dni” donde el usuario Autoridad selecciona y sube una fotografía del DNI de la Persona. 4. El usuario Autoridad le proporciona el teclado a la Persona, la que ingresa una contraseña y luego lo devuelve. 5. El usuario Autoridad presiona el botón de “Aceptar”. 6. El sistema guarda los datos de la Persona con un estado “dado de alta”

Excepciones	2. Ocurre un error al subir la imagen. El sistema muestra el error y solicita repetir la operación. 3. Ocurre un error al subir la imagen. El sistema muestra el error y solicita repetir la operación. 4. La Persona ingresa una contraseña que no cumple con los lineamientos de seguridad de contraseñas. El sistema muestra el error junto al campo de contraseñas. 5.a Se detecta al menos un campo vacío. El sistema deshabilita el botón de “Aceptar”. 5.b Se detecta al menos un campo con formato no válido. El sistema deshabilita el botón de “Aceptar”. 5.c Se detecta al menos un campo con caracteres no válidos. El sistema deshabilita el botón de “Aceptar”. 5.d No hay conexión a internet. El sistema informa el error.
Observaciones	-

En la Tabla 2 se presenta el caso de uso “registro”, que representa el primer paso en la interacción de una persona con el sistema. Esta actividad es necesaria para luego permitir la instalación y uso de la aplicación móvil.

Tabla 3

Caso de uso: instalar app

Actor principal	Persona
Actores secundarios	-
Objetivo	Instalar la aplicación y registrar el dispositivo móvil.
Precondiciones	El celular de la Persona presenta lector de huella dactilar.
Disparador	La Persona instala la aplicación en su dispositivo móvil
Escenario de éxito	1. La Persona ingresa su DNI y la contraseña que escribió en el momento de registro en el formulario de la aplicación móvil. 2. La Persona autoriza la operación mediante su huella dactilar. 3. La Persona presiona el botón de “Aceptar”. 4. El sistema borra la contraseña de la Persona. 5. El sistema le envía la semilla de la Persona a la aplicación.
Excepciones	3. No hay conexión a internet. La aplicación móvil muestra un mensaje de error y solicita reintentar la operación.
Observaciones	-

La Tabla 3 describe la acción que toma una persona registrada en el sistema para configurar inicialmente la aplicación, permitiendo la posterior generación de tokens.

Tabla 4

Caso de uso: Suspender persona

Actor principal	Autoridad
Actores secundarios	-
Objetivo	Deshabilitar la verificación de tokens para una Persona en particular.
Precondiciones	1. El usuario Autoridad está registrado en el sistema. 2. El usuario Autoridad mantiene una sesión activa en el sistema. 3. La Persona está registrada en el sistema con el estado “dado de alta”.
Disparador	El Autoridad accede a la sección de suspensión de personas.
Escenario de éxito	1. El usuario Autoridad selecciona una Persona. 2. El usuario Autoridad ingresa una razón para la suspensión. 3. El usuario Autoridad ingresa un rango de fechas de inicio y fin de la suspensión. 4. El Autoridad presiona el botón “Aceptar” para confirmar la operación.
Excepciones	2. El Autoridad accede a otra sección. El sistema muestra la pantalla accedida y se pierde el progreso en el formulario de suspensiones. 3.a El rango de fechas ingresado no es válido. El sistema indica el error junto al campo. 3.b El usuario Autoridad puede elegir no

	ingresar una fecha de fin. El sistema hace efectiva la nueva suspensión en la fecha de inicio indicada, pero por un tiempo indefinido. Posteriormente un usuario Autoridad puede anular dicha suspensión, donde el sistema establece la fecha actual como la fecha de fin de suspensión. 4. El Autoridad presiona el botón de “Cancelar”. El sistema muestra la pantalla previa.
Observaciones	1. El sistema debe registrar todos los períodos de suspensión de Personas. 2. Las razones de suspensión pueden ser: a. Voluntad propia b. Pedido de un superior c. Fallecimiento

La Tabla 4 relata la acción tomada por un usuario Autoridad, que suspende un usuario por alguna razón disponible, lo que rechaza tokens generados dentro un período específico de tiempo. Las suspensiones se explicarán más en detalle en el apartado de descripción de la solución.

Tabla 5

Caso de uso: alta de usuario

Actor principal	Administrador
Actores secundarios	-
Objetivo	Dar de alta un usuario en el sistema, ya sea Autoridad o Consulta.
Precondiciones	1. El usuario Administrador está registrado en el sistema. 2. El usuario Administrador mantiene una sesión activa en el sistema.
Disparador	El Administrador accede a la sección para dar de alta usuarios.
Escenario de éxito	1. El Administrador ingresa email y contraseña del usuario a dar de alta. 2. El Administrador selecciona un rol de los disponibles. 3. El Administrador presiona el botón de “Aceptar”.
Excepciones	3.a Se detecta al menos un campo vacío. El sistema deshabilita el botón “Aceptar”. 3.b Se detecta al menos un campo con formato no válido. El sistema deshabilita el botón “Aceptar”. 3.c Se detecta al menos un campo con caracteres no válidos. El sistema deshabilita el botón “Aceptar”. 3.d El email ingresado ya está en uso. El sistema muestra un mensaje de error y la operación no se completa. 3.e El Administrador presiona el botón de “Cancelar”. El sistema muestra la pantalla anterior.
Observaciones	-

La Tabla 5 narra el proceso de alta de nuevos usuarios autorizados que luego utilizarán el sistema web, por parte de un usuario Administrador. No se pueden dar de alta otros usuarios con el rol de Administrador mediante la aplicación web.

Tabla 6

Caso de uso: generar token

Actor principal	Persona
Actores secundarios	-
Objetivo	Generar un token.
Precondiciones	1. La Persona se encuentra registrada en el sistema. 2. La Persona completó el proceso de instalación de la app.
Disparador	La Persona accede a la sección de generar tokens de la aplicación.
Escenario de éxito	1. La Persona accede utilizando su huella dactilar. 2. La aplicación genera el token y lo presenta en pantalla.
Excepciones	1. No se reconoce la huella dactilar. La aplicación móvil solicita reintentar la operación. 2. Ocurre un error al generar el token. La aplicación móvil informa el error.
Observaciones	Al cabo de un minuto, se solicita la huella nuevamente. También se requiere la huella nuevamente cuando se desbloquea el celular luego de apagada la pantalla.

La Tabla 6 describe los pasos que realiza una persona registrada para obtener tokens.

Tabla 7

Caso de uso: Verificar token

Actor principal	Consulta
Actores secundarios	-
Objetivo	Comprobar la validez del token.
Precondiciones	El usuario Consulta está registrado en el sistema. El usuario Consulta mantiene una sesión activa en el sistema.
Disparador	El usuario Consulta recibe un token y debe verificarlo.
Escenario de éxito	1. El usuario Consulta ingresa el token, el nombre y el apellido de la Persona. 2. El usuario Consulta ingresa la fecha y la hora de generación de token. 3. El usuario Consulta presiona el botón “Verificar”. 4. El sistema presenta un mensaje de éxito junto con el DNI de la Persona.
Excepciones	3.a Se detecta al menos un campo vacío. El sistema deshabilita el botón “Aceptar”. 3.b Se detecta al menos un campo con un formato no válido. El sistema deshabilita el botón “Aceptar”. 3.c Se detecta al menos un campo con caracteres no válidos. El sistema deshabilita el botón “Aceptar”. 3.d No hay conexión a internet. 4. El token no es válido. El sistema muestra el mensaje informando que el token no es válido.
Observaciones	-

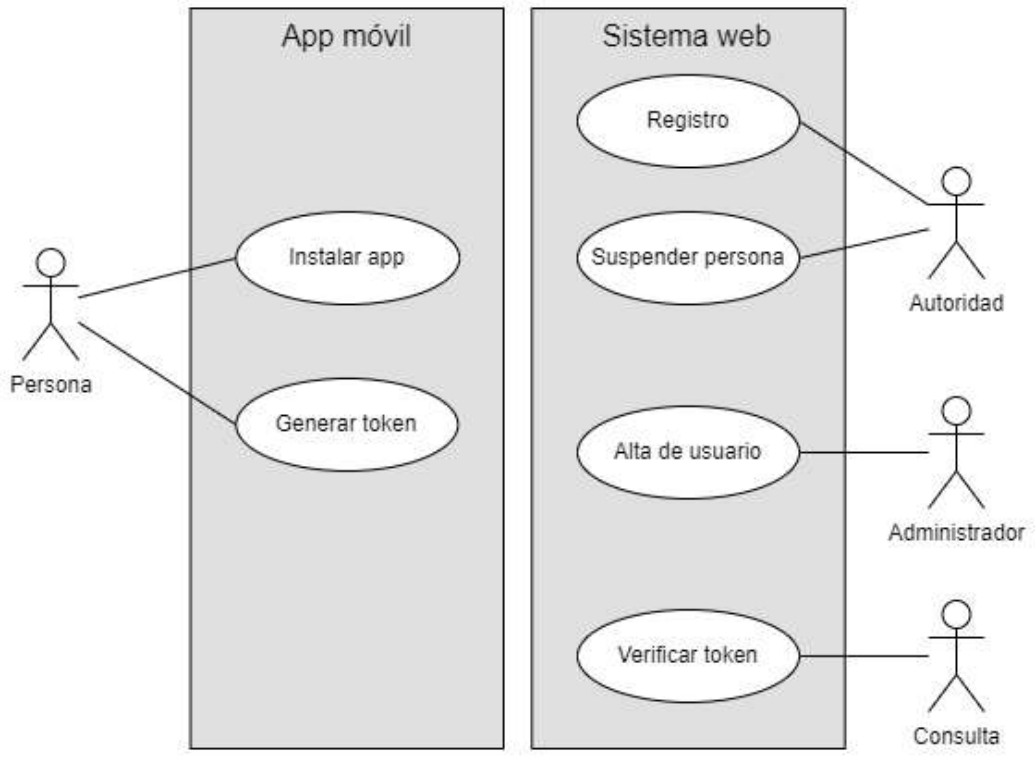
La Tabla 7 hace referencia al proceso de verificación de tokens, realizado bajo demanda por un usuario Consulta.

Luego de haber precisado los casos de uso, se confeccionó el diagrama de casos de uso, del cual Sommerville (2016) establece:

> Los casos de uso son documentados utilizando un diagrama de casos de uso de alto nivel. El grupo de casos de uso representa todas las posibles interacciones que serán descritas en los requerimientos del sistema. Los actores en el proceso, quienes pueden ser humanos u otros sistemas, son representados como hombres palo. Cada tipo de interacción se representa como una elipse nombrada. (p.125)

Figura 3

Diagrama de casos de uso



La Figura 3 representa el diagrama de casos de uso resultante del análisis previo. La relación del actor con el o los casos de uso que inicia se establece mediante una línea

recta. Las distintas aplicaciones, “sistema web” y “aplicación móvil” están representadas mediante rectángulos grises, que encuadran casos de uso iniciados en dicha plataforma. La distribución de los casos de uso no indica secuencialidad alguna.
El manifiesto ágil postula “Software funcionando sobre documentación extensiva” (Beck et al., 2001). Sin embargo, al contar con un diseño preliminar, funcionalidades definidas en gran medida junto con el orden en que deben ejecutarse y los actores correspondientes, se determinó conveniente la documentación en casos de uso. Con respecto a esta decisión, también pesó el hecho de que todos los casos de uso son iniciados por personas físicas, bajo demanda, y no por otros sistemas de manera automática. De esta manera, los documentos de casos de uso presentados anteriormente aportaron un detalle pormenorizado de las tareas a realizar, mientras que el diagrama de casos de uso contribuyó con una visión general de cómo encajan las distintas funcionalidades en la solución.

Desarrollo

Se llevó a cabo la implementación del sistema concebido por (Lugani, 2022), añadiendo los ligeros cambios que surgieron de las definiciones concretas detalladas en los casos de uso.

En pos de brindar un sistema de software fácilmente mantenible, se consideraron las arquitecturas y los lenguajes de programación de otros sistemas de software actualmente desplegados y en uso en la institución, por lo que se optó por utilizar la herramienta JHipster (https://www.jhipster.tech/).

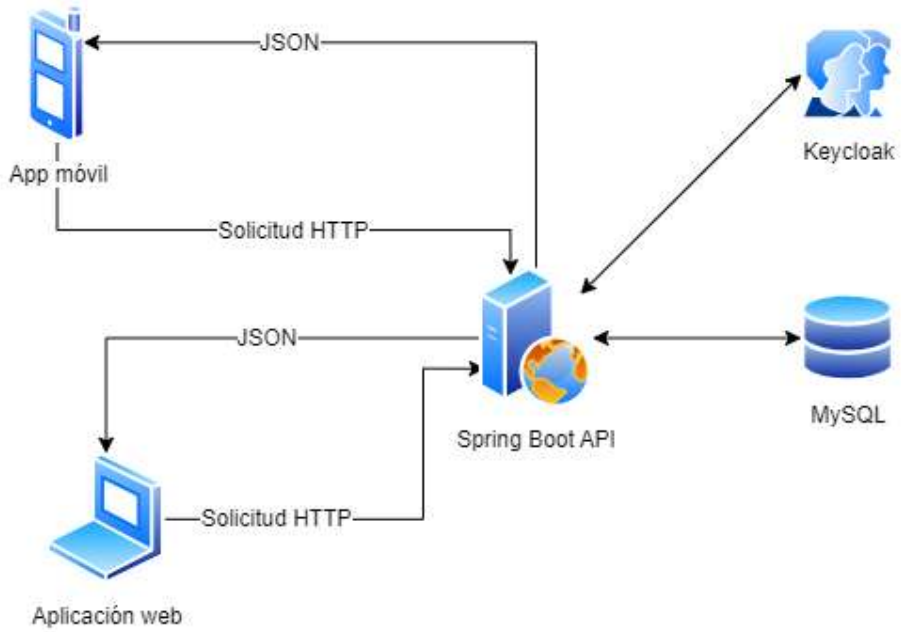
JHipster es un generador de código, que permite crear proyectos según la configuración especificada, entre los que se pueden elegir el método de autenticación, la base de datos, la tecnología de backend y la tecnología de frontend. Se escogió una base de datos MySQL, un backend basado en Spring Boot, un frontend web basado en Angular y Keycloak como servicio de autenticación. Como resultado se obtuvo un proyecto full-stack previamente configurado, que agiliza considerablemente el desarrollo.

Además de generar un proyecto según la arquitectura deseada, se pueden definir las entidades pertinentes mediante el Lenguaje de Dominio de JHipster (JDL, por sus siglas en inglés de JHipster Domain Language).

Al utilizar esta herramienta, se agiliza el desarrollo y se obtiene un repositorio de código generado siguiendo buenas prácticas de programación. Se utilizó JHipster v7.9.3, que configuró el proyecto para utilizar MySQL 8.0.30, Keycloak 19.0.1 y Spring Boot 2.7.3.

Figura 4

Diagrama de arquitectura del sistema



En la Figura 4 se observa el diagrama de arquitectura del sistema propuesto. Keycloak (https://www.keycloak.org/) es un gestor de identidad y acceso basado en Java. En esta arquitectura, se utiliza para establecer los roles de los distintos usuarios, Administrador, Autoridad y Consulta. Los usuarios autorizados inician sesión mediante Keycloak para acceder a la aplicación web, donde realizan las actividades que corresponden a su rol.

El backend consta de una API REST desarrollada con el framework Spring Boot. Dicha API se comunica con la aplicación web, la aplicación móvil, la base de datos MySQL y el servidor Keycloak haciendo uso de la lógica necesaria para proveer los servicios definidos. Además, establece qué operaciones pueden realizar los distintos usuarios dependiendo de su rol.

Por otro lado, la aplicación móvil es utilizada por las personas para generar sus tokens y solo requiere de conexión a internet al momento de ingresar la contraseña que se estableció al momento de registro presencial, no cuando se generen tokens. Ya que el sistema propuesto requiere de autenticación fuerte, se utilizaron las librerías de Expo (https://expo.dev/) para implementar autorización mediante huella dactilar y almacenamiento

seguro. La particularidad de Expo es que permite probar aplicaciones sin necesidad de instalar manualmente o generar los APK utilizando el cliente Expo (https://expo.dev/client), lo que permite acelerar el desarrollo al brindar funcionalidades para lidiar con errores. La versión de React Native utilizada es la 0.71.73, mientras que la versión de Expo es la 48.0.9

En enero del año 2024, los dos sistemas operativos más utilizados en dispositivos móviles en Argentina eran Android con el 90,75% y iOS con el 9,05%, lo que representa el 99,80% del mercado (StatCounter, s.f.-c). StatCounter es un servicio de analíticas web que registra la actividad de los usuarios en distintas páginas web que lo utilizan (https://statcounter.com/). De esta manera se puede extraer información del tipo de dispositivo que accede a los más de un millón y medio de sitios miembros, contabilizando más de cinco mil millones de visitas mensuales alrededor de todo el mundo (StatCounter, s.f.-a).

En el caso de Android, bajo la clase BiometricsPrompt del paquete android.hardware.biometrics, a partir de Android 9, API 28 en adelante, se incluye el escaneo de huella dactilar, mientras que el reconocimiento facial se integró bajo la misma clase en la versión Android 10, API 29 en adelante (Android Open Source Project, s.f.).

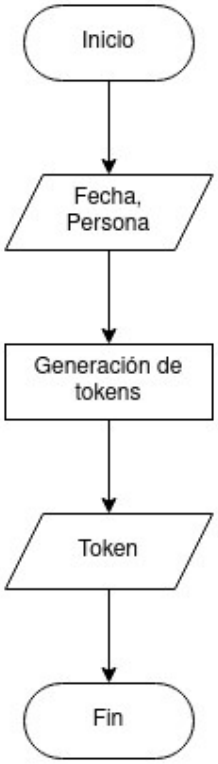
Según un análisis de StatCounter que diferencia las versiones de Android que generan tráfico web desde Argentina, en enero del año 2024, el 37.06% se produjo en la versión 13.0, el 18.33% en la 11.0, el 17.01% en la 12.0, el 9.12% en la 10.0 y el 5.23% en la versión “9.0 Pie” (StatCounter, s.f.-b). Los dispositivos que corrían Android 9.0 Pie en adelante acumularon el 86,75% del mercado.

Por lo tanto, se consideraron las estadísticas de StatCounter y la disponibilidad de dispositivos para priorizar el diseño, el desarrollo y las pruebas; actividades que se realizaron exclusivamente sobre un celular con escáner de huella dactilar que corre Android 9.

Descripción de la solución

Las dos piezas fundamentales que establecen la correctitud e integridad de la solución son los algoritmos de generación y verificación de tokens. En este apartado se explicarán como operaciones de caja negra mediante diagramas de flujo.

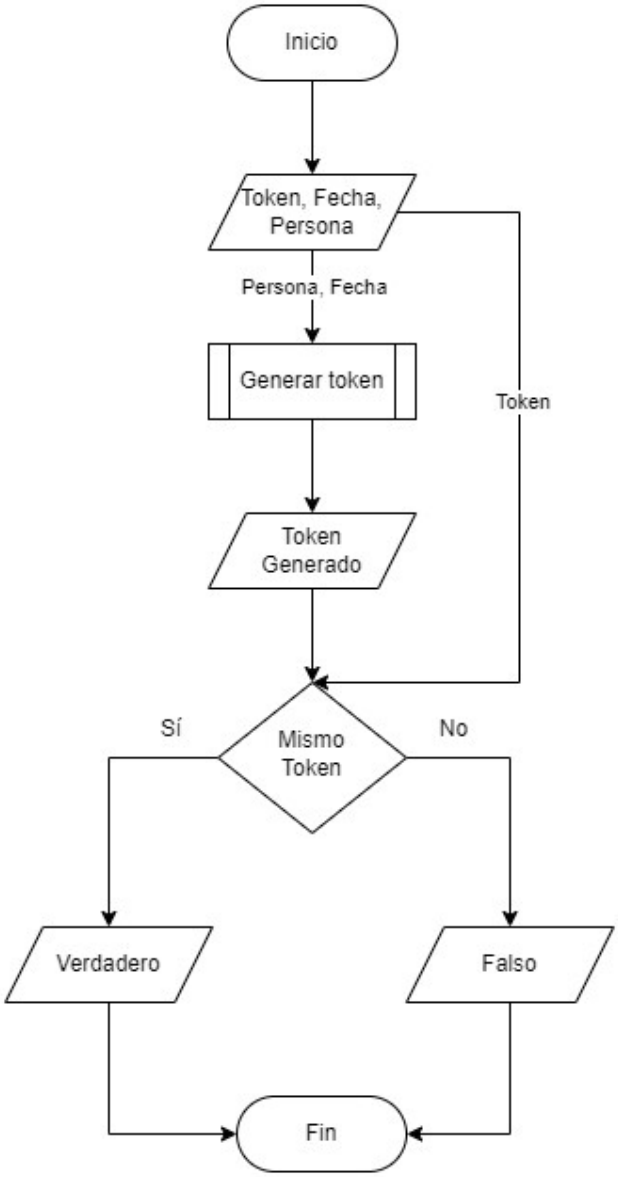
Figura 5



Generar token

El algoritmo presentado en la Figura 5, hace uso del DNI de la persona; la semilla de la persona; la fecha y la hora con exactitud al minuto. Por lo tanto, se utiliza el DNI junto con la semilla, que identifican unívocamente a una persona y a una instalación en un dispositivo particular, además de una fecha y hora exactas al minuto.

Figura 6



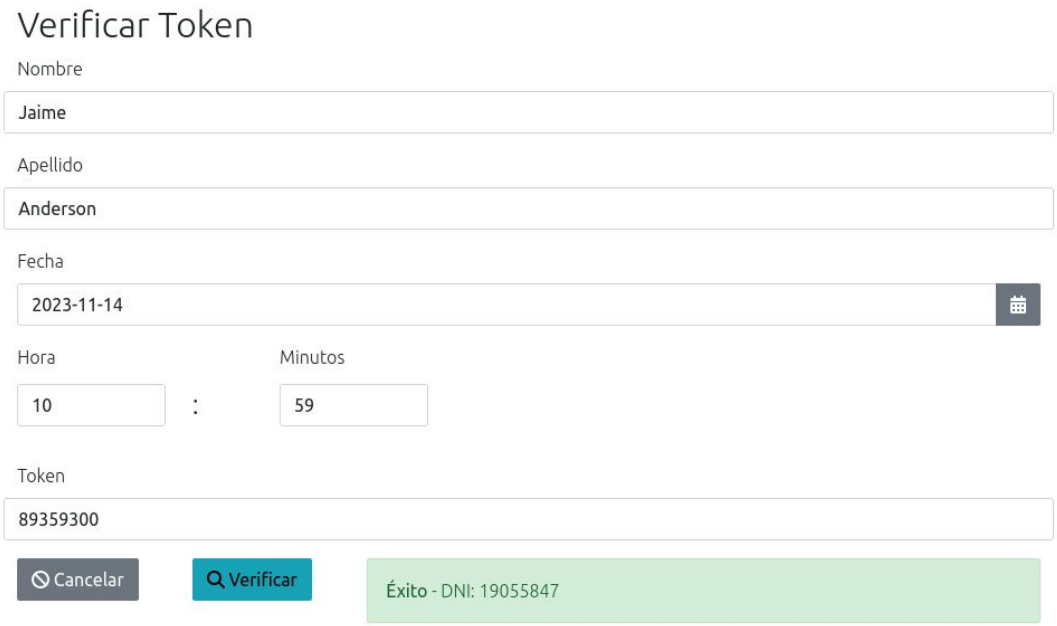
Verificar token

El proceso descrito en la Figura 6 utiliza la generación de tokens para comparar un token particular con un token generado, ya que los mismos datos de entrada generan siempre el mismo token. El resultado es un valor booleano, por lo que si el token generado es el mismo que el token ingresado, la verificación retorna verdadero y en caso contrario, falso. De esta manera se puede comprobar la integridad de un token, el cual representa la actividad de una persona registrada en una fecha y hora exacta.

Al momento de verificar un token, los usuarios Consulta ingresan los datos requeridos en el formulario que se observa en la Figura 7.

Figura 7

Formulario de verificación de tokens



La Figura 7 presenta el formulario de la aplicación web para la verificación de los tokens. Para simplificar el uso por parte de los usuarios Consulta, se ingresa nombre y apellido de la persona, reemplazando el DNI; la fecha; hora y minutos; y por último, el token a verificar. Cabe destacar que la única manera de verificación presente en esta solución requiere que el usuario Consulta conozca el dueño del token a verificar, el token en cuestión, además de la fecha y la hora de generación del mismo. Solo el hecho de conocer un token no presenta utilidad alguna, ya que no existe forma de conocer el DNI de la persona, la semilla, la fecha o la hora de generación a partir del token.

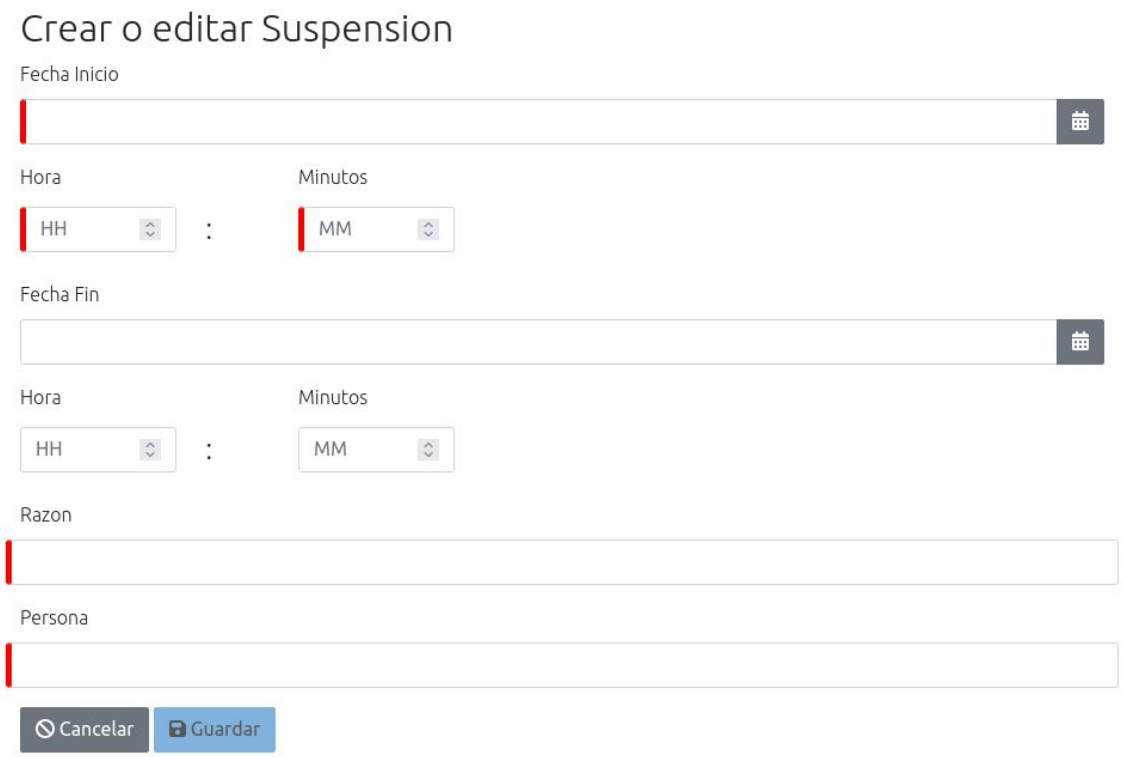
En consecuencia, la solución se basa en la sincronización de relojes, ya que una instalación específica de la aplicación móvil en un dispositivo genera tokens para una sola persona, y a su vez, la aplicación web conoce los datos de todas las personas para poder verificar sus tokens. De esta manera, es necesario que las personas que utilizan la aplicación móvil configuren su dispositivo en la zona horaria definida por la aplicación central, que comprende la web y la API REST. Por lo tanto, la aplicación móvil no requiere de conexión a internet durante la generación, y en el caso de la aplicación web, asumiendo que el resto de los servicios, Keycloak; API REST y MySQL se ejecuten en la misma red local, tampoco requeriría de conexión a internet.

Dado que la solución depende de la sincronización de relojes, resulta conveniente definir los conceptos de suspensiones y rango de autenticación.

Una suspensión es el mecanismo para inhabilitar tokens de una persona, representado por un rango de fechas. Particularmente, los tokens cuya fecha y hora de generación se encuentren comprendidos dentro del período de la suspensión, serán rechazados por el algoritmo de verificación. Los usuarios Autoridad gestionan las suspensiones mediante la aplicación web, utilizando el formulario presentado en la Figura 8.

Figura 8

Formulario de suspensión



En la figura 8 se observan los datos necesarios para suspender una persona, particularmente se necesita una fecha de inicio junto con su hora y minutos; una razón de suspensión, la persona a suspender y, opcionalmente una fecha de fin junto con su hora y minutos. En el caso de que sea proporcionada la fecha de fin, el módulo verificador reconocerá como inválidos todos los tokens con hora y minuto correspondientes al período

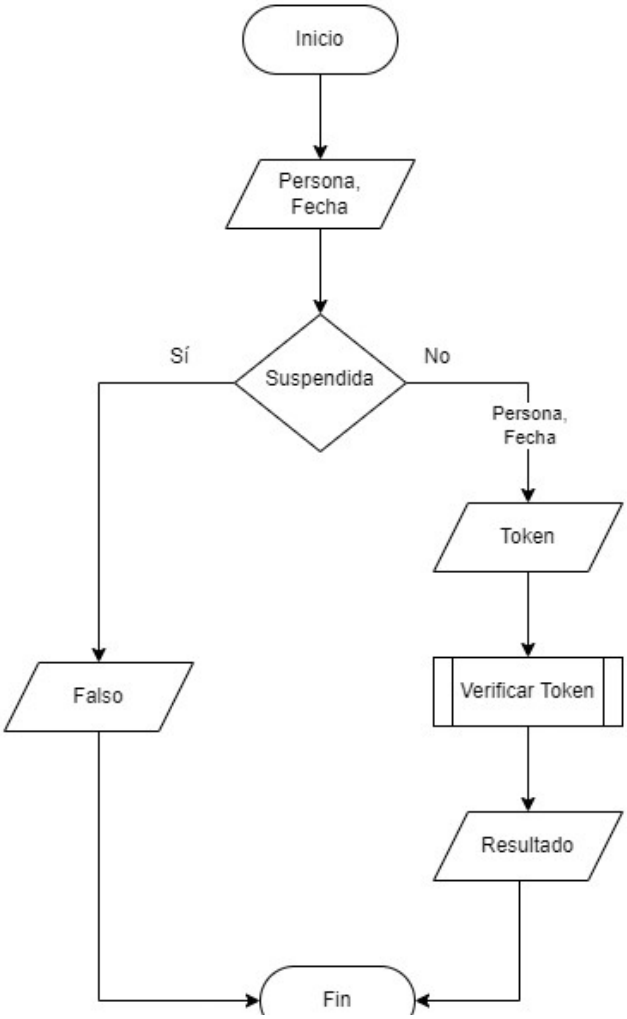
de suspensión. En el hipotético caso que la suspensión culmine en una fecha arbitraria a las 11:00, el token de las 11:00 será el primero en volver a contar como válido.

En cuanto a las suspensiones creadas sin una fecha de fin, se asume una suspensión por término indefinido. En la figura 8 se observa el botón “Anular suspensión” que se habilita al editar una suspensión no finalizada y la termina en el mismo instante que se procesa la solicitud; los tokens generados en el minuto inmediatamente posterior a la anulación ya serán válidos.

Las suspensiones se asignan por persona, por lo tanto, una persona tiene una o muchas suspensiones, que se almacenan en la base de datos. Ya que la aplicación web centraliza la gestión de suspensiones, la aplicación móvil solo genera tokens bajo demanda, sin siquiera estar al tanto que la persona se encuentra suspendida.

Figura 9

Verificar token considerando suspensiones

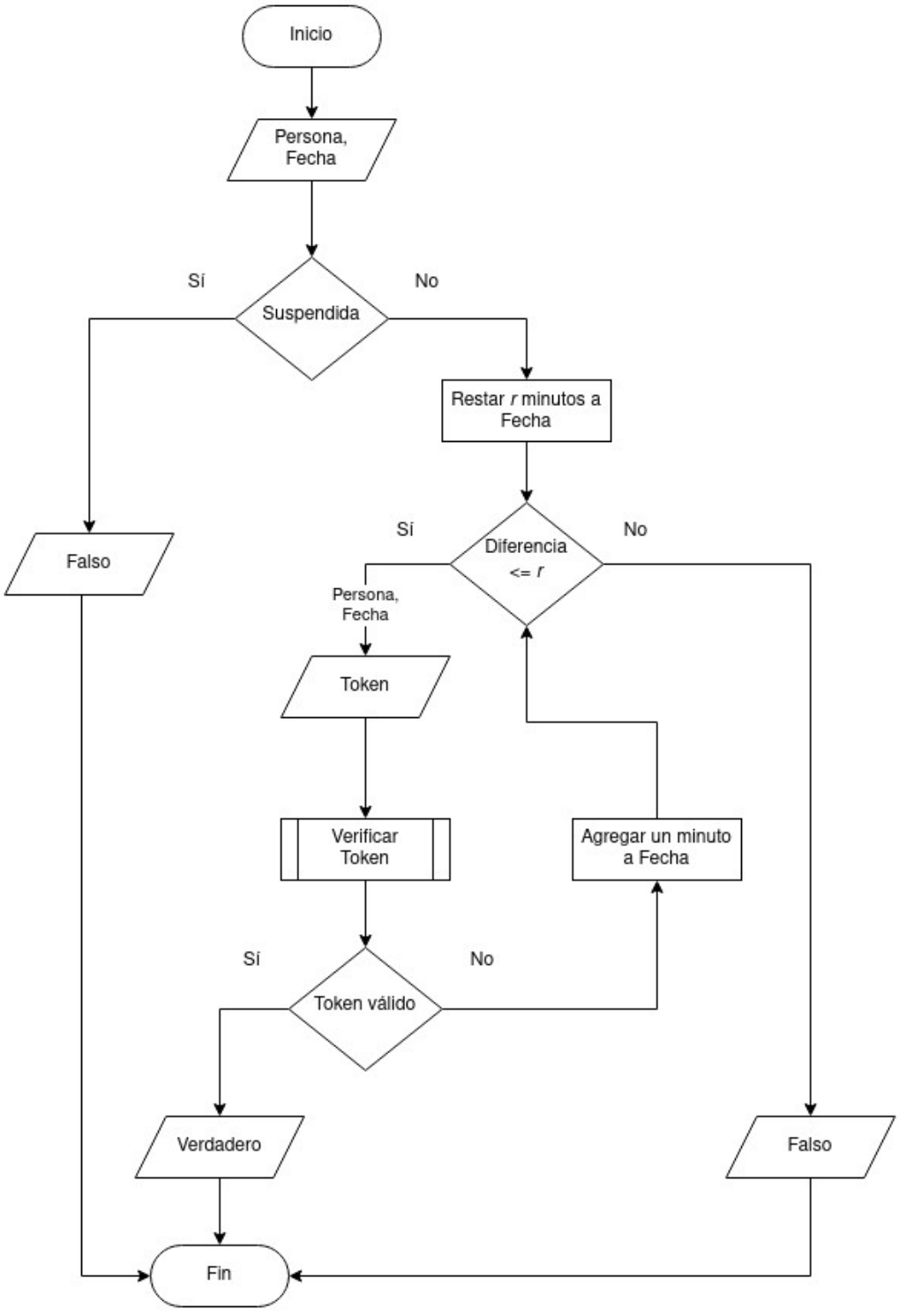


Al observar la Figura 9, se observa que el token puede ser rechazado tanto porque la fecha y hora están comprendidas dentro del período de suspensión o porque falló efectivamente la verificación de token.

Por otro lado, el rango de autenticación se expresa como un número entero positivo y se configura mediante la clase *src/main/java/ar/edu/unrn/saboff/web/rest/constants/TokenConstants.java* del código fuente de la API. Este rango, representado por un valor *r*, permite considerar válido un token, *r* minutos antes y *r* minutos después de su generación. Este rango representa los minutos contiguos a la hora exacta de generación para los cuales un token se considera válido.

Figura 10

Verificar token considerando suspensiones y rango de autenticación



Este concepto se ilustra en la Figura 10, donde se considera la fecha y hora ingresadas en el formulario de la Figura 8, al momento de verificar un token. Particularmente, la aplicación

web genera todos los tokens para la persona en cuestión que correspondan con la fecha ingresada, pero variando los minutos contiguos que se encuentren dentro del rango de autenticación. En caso de que un token de los generados sea el mismo que el ingresado, se considera este último como válido.

Por ejemplo, con un rango de autenticación de 1, el módulo de verificación de tokens considerará válidos tokens que se ingresen con 1 minuto de diferencia respecto a un token específico. Esto permite que si una persona tarda en ingresar un token generado, no se lo tome como inválido.

Tabla 8

Ejemplo con r = 1

Hora exacta	Token	Token válido
10:57	89271345	No
10:58	45689210	Sí
10:59	72310986	Sí
11:00	54872319	Sí
11:01	18763492	No

En la Tabla 8 se observa un ejemplo para un token generado en una fecha determinada a las 10:59, 72310986, cuya fila se encuentra sombreada. Con un rango de autenticación de 1, para las horas correspondientes a la misma fecha de generación pero con una diferencia de 1 minuto, el token se reconoce como válido.

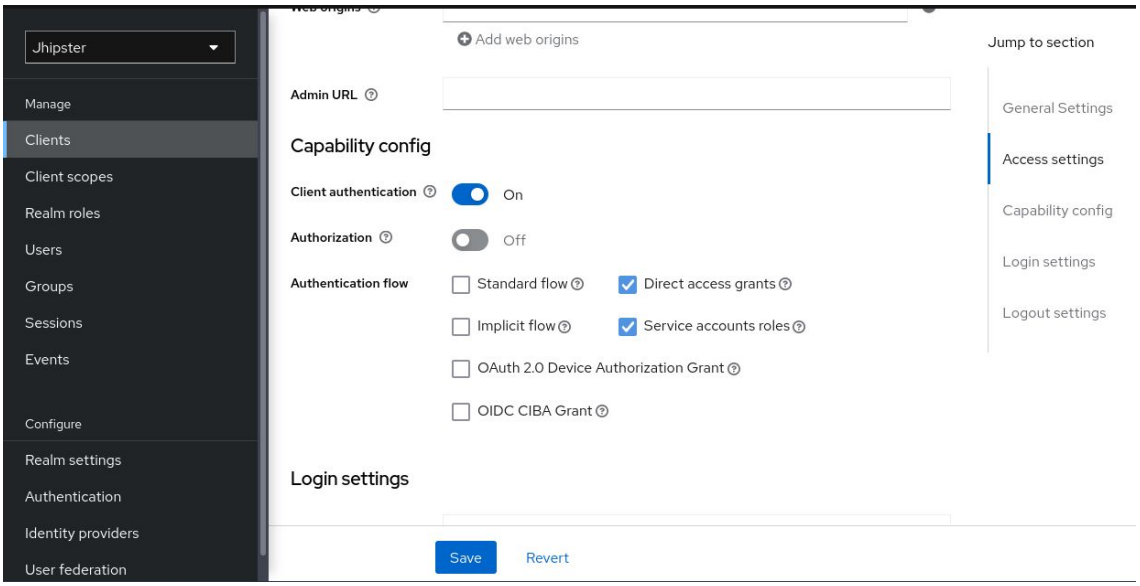
Configuración

Cuando se intente desplegar una instancia de este sistema en la institución, se debe tener en cuenta la configuración específica del reino (realm) de Keycloak. Estos ajustes en particular permiten a los usuarios Administrador agregar nuevos usuarios, con los roles Autoridad y Consulta, mediante la aplicación web. Dentro del reino a utilizar se debe crear un nuevo cliente OIDC[2] (https://openid.net/), para que Keycloak reconozca las solicitudes que realice la API REST.

[2] OIDC: acrónimo en inglés de "OpenID Connect". Es un protocolo de autenticación estandarizado que facilita la identificación de usuarios. Utiliza servidores de autorización y evita la gestión de contraseñas por parte de los desarrolladores. Generalmente permite realizar un login mediante

Figura 11

Keycloak: capacidades del cliente



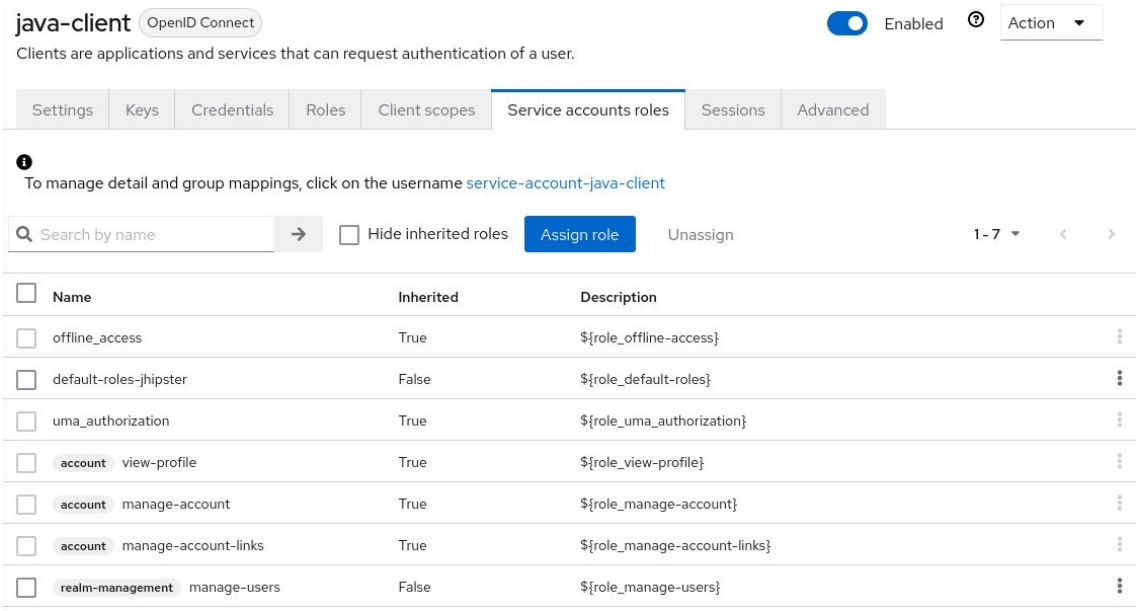
En la Figura 11 se observan las capacidades del cliente necesarias. Dicho cliente, llamado “java-client”, pertenece al reino JHipster.

Una cuenta de servicio (service account), permite operar sobre los recursos del servidor Keycloak, usuarios y roles, mediante clientes autorizados. Los roles de cuentas de servicio son ajustes que brindan permisos especiales a clientes específicos.

servicios de terceros como Google y Facebook entre otros. Además, define estándares de autenticación biométrica como reconocimiento facial y escaneo de huella dactilar.

Figura 12

Keycloak: roles de cuenta de servicio

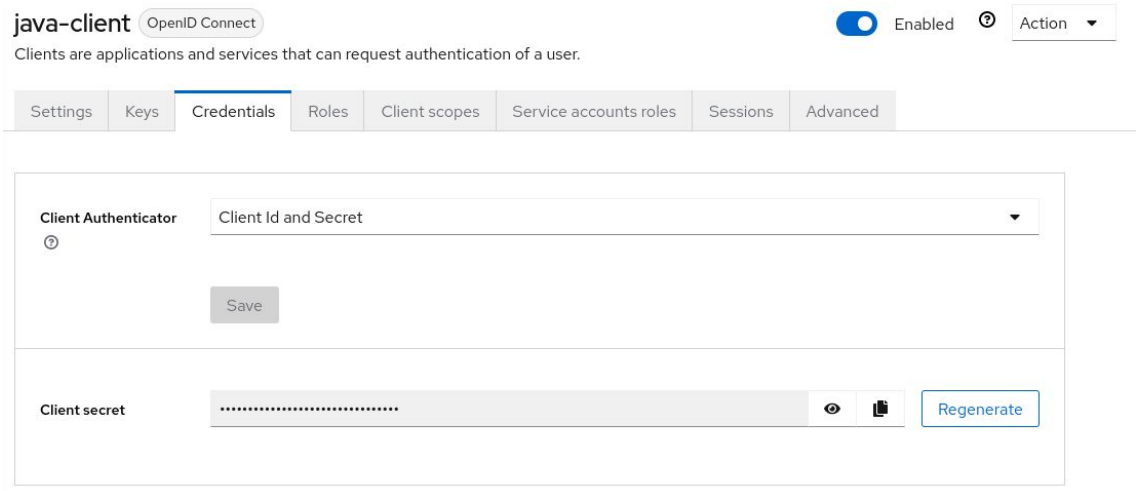


En la Figura 12 se presentan los roles de cuenta de servicio asignados al cliente “java-client”. Una vez asignado el rol “manage-users” de la categoría “realm-management”, el cliente tendrá el nivel de autorización necesaria para gestionar los usuarios del servidor Keycloak, posibilitando así la creación de nuevos usuarios.

Ya que el cliente se encuentra configurado, se debe copiar el secreto del cliente y pegarlo en una variable de entorno llamada “KEYCLOAK_CLIENT_SECRET”.

Figura 13

Keycloak: credenciales del cliente



La Figura 13 expone la pestaña de credenciales del cliente “java-client”. El campo “Client secret” contiene el secreto del cliente, que se debe utilizar para autenticar el cliente con el servidor Keycloak, brindando otra capa de seguridad a la solución.

Por último, se debe ingresar la configuración presentada en la Figura 14, en el archivo *src/main/resources/config/application.yml* presente en el proyecto de la API.

Figura 14

Application.yml

```
…
spring:
security:
oauth2:
client:
provider:
oidc:
        issuer-uri: http://localhost:9080/realms/jhipster
registration:
oidc:
client-id: web_app
        client-secret: web_app
        scope: openid, profile, email, offline_access # last one for refresh tokens
java-client:
issuer-uri: http://localhost:9080
realm: jhipster
client-id: java-client
grant-type: client_credentials
…
```

En la Figura 14 se observa un resumen del archivo application.yml, utilizado por JHipster para configurar el backend basado en Spring Boot. Se encuentra resaltada la configuración que se debe agregar al archivo. La entrada “java-client” debe estar con el mismo nivel de indentación que “spring.security.oauth2.client”, por lo que resulta en “spring.security.oauth2.java-client”.

Disponibilidad de fuentes

El código fuente de la solución descrita en este trabajo se encuentra dividido en tres proyectos: api, frontend y mobile. Cada uno de los proyectos consiste de un repositorio de código distinto, todos alojados en repositorios privados en la plataforma GitLab (https://gitlab.com/).

Conclusiones

Se logró el objetivo propuesto, ya que la aplicación funciona de forma adecuada. Se implementaron exitosamente tanto la generación como la verificación de tokens, los dos casos de uso esenciales para el correcto funcionamiento de la solución. Además, se desarrollaron satisfactoriamente las operaciones restantes en lo que respecta a la gestión de: los usuarios autorizados, las personas que utilizan la aplicación móvil y sus suspensiones. De este modo se obtuvo una solución independiente, que dispone de todos los servicios necesarios para su uso.
JHipster fue un pieza fundamental para el desarrollo de este sistema, debido a que rápidamente genera los proyectos backend y frontend, además de configurarlos para proveer una experiencia de desarrollo sin mayores interrupciones. También, permite definir entidades en un archivo JDL, que luego se traducen en CRUDs tanto en el frontend como en el backend. Sin embargo, cada vez que se requiera modificar la definición de entidades, se debe tener sumo cuidado de no sobreescribir los avances en el código fuente. Sobre este punto se demostró la utilidad de Git como software de versionado, permitiendo deshacer, rehacer cambios y experimentar en otras ramas con facilidad, disminuyendo este riesgo. A medida que fue avanzando el desarrollo, luego de modificar los archivos JDL, se obvió la generación de entidades de JHipster para evitar conflictos.

Líneas futuras de trabajo

El desarrollo de este sistema se planteó como un PMV (Producto Mínimo Viable) donde solo se implementan las funcionalidades básicas definidas en base a la elicitación de requerimientos mencionada previamente. Por lo tanto, es posible realizar mejoras o añadir nuevas funcionalidades para complementar los servicios brindados por las distintas aplicaciones.

En primer lugar, compilar la aplicación para iOS utilizando Expo. Si bien se utiliza el mismo código fuente ya que React Native es un framework multiplataforma, es probable que se requieran de optimizaciones al utilizar el reconocimiento facial o touch id de iOS. Es necesario contar con un dispositivo móvil con iOS para realizar pruebas y detectar posibles errores que solo ocurren en hardware al generar una versión completa.

En segundo lugar, generar una configuración para que todos los servicios, Keycloak, MySQL, backend y frontend se ejecuten en Kubernetes. Si bien JHipster produce imágenes

de Docker a partir de los proyectos, todavía requiere de intervención manual en la configuración para un despliegue exitoso en Kubernetes.

Por último, automatizar el registro, tomando fotografías del DNI y del rostro de la persona, para luego validarlas mediante el Sistema de Identificación Digital (https://www.argentina.gob.ar/interior/renaper/sid-sistema-de-identidad-digital). Este tipo de prácticas comúnmente utilizadas por aplicaciones móviles destinadas a operaciones bancarias o financieras permitirían realizar el registro de estudiantes que cursen sus estudios plenamente a distancia en localidades lejanas a las sedes de la Universidad.

Bibliografía

1. Beck, K., et al. (2001) Manifiesto por el Desarrollo Ágil de Software. https://agilemanifesto.org/iso/es/manifesto.html
2. Biometrics. (s.f.). Android Open Source Project. https://source.android.com/docs/security/features/biometric
3. Cloudflare. (s.f.-a). *¿What is authentication?*. https://www.cloudflare.com/learning/access-management/what-is-authentication/
4. Cloudflare. (s.f.-b). *¿Qué es la gestión de identidad y acceso?*. https://www.cloudflare.com/es-es/learning/access-management/what-is-authentication/
5. Departamento de Información Universitaria. (2022). *Síntesis de información estadísticas universitarias 2021-2022*. https://www.argentina.gob.ar/sites/default/files/sintesis_2021-2022_sistema_universitario_argentino_-_ok.pdf
6. Lugani, C. F. (2022). *Proceso de registro e identificación sin control automático para la Universidad Nacional de Río Negro*. Simposio de Informática en el Estado (SIE 2022) - JAIIO 51 (Modalidad virtual y presencial (UAI), octubre 2022). http://sedici.unlp.edu.ar/handle/10915/151760
7. Pearson IT Certification. (6 de Junio de 2011). *Understanding the Three Factors of Authentication*. https://www.pearsonitcertification.com/articles/article.aspx?p=1718488
8. Rasmusson, J. (2010). *The Agile Samurai: How agile masters deliver great software* (S. D. Pfalzer, Ed.; 1ra edición.). The Pragmatic Bookshelf.
9. StatCounter. (s.f.-a). FAQ | *Statcounter Global Stats*. https://gs.statcounter.com/faq#methodology

10. StatCounter. (s.f.-b). Mobile Android Version Market Share Argentina. https://gs.statcounter.com/android-version-market-share/mobile/argentina/#monthly-202302-202401
11. StatCounter. (s.f.-c). *Mobile Operating System Market Share Argentina*. https://gs.statcounter.com/os-market-share/mobile/argentina/#monthly-202302-202401