Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América
Facultad de Ingeniería de Sistemas e Informática
Escuela Profesional de Ingeniería de Sistemas

Implementación de un microservicio de autenticación con un Identity Server para una aplicación de banca móvil de una fintech americana

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el Título Profesional de Ingeniero de Sistemas

AUTOR

Paul Cristian PERCCA JULCA

ASESOR

Robert Elías ESPINOZA DOMÍNGUEZ

Lima, Perú

2022

Referencia bibliográfica

Percca, P. (2022). *Implementación de un microservicio de autenticación con un Identity Server para una aplicación de banca móvil de una fintech americana*. [Trabajo de suficiencia profesional de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática, Escuela Profesional de Ingeniería de Sistemas]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	PAUL CRISTIAN PERCCA JULCA
Tipo de documento de identidad	DNI
Número de documento de identidad	76380358
URL de ORCID	https://orcid.org/0000-0003-0299-0610
Datos de asesor	
Nombres y apellidos	ROBERT ELÍAS ESPINOZA DOMÍNGUEZ
Tipo de documento de identidad	DNI
Número de documento de identidad	08136325
URL de ORCID	https://orcid.org/0000-0002-3456-8302
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	JORGE LUIS CHÁVEZ SOTO
Tipo de documento	DNI
Número de documento de identidad	08675814
Miembro del jurado 1	
Nombres y apellidos	MARIO AGUSTÍN HUAPAYA CHUMPITAZ
Tipo de documento	DNI
Número de documento de identidad	15386891
Datos de investigación	
Línea de investigación	No aplica
Grupo de investigación	No aplica
Agencia de financiamiento	Propio

Ubicación geográfica de la investigación	País: Perú Departamento: Lima Provincia: Lima Distrito: Cercado de Lima Jr. Carlos Amezaga No. 375 Universidad Nacional Mayor de San Marcos Latitud: -12.0564232 Longitud: -77.0843327
Año o rango de años en que se realizó la investigación	2021
URL de disciplinas OCDE	2.02.04 -- Ingeniería de sistemas y comunicaciones https://purl.org/pe-repo/ocde/ford#2.02.04

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
Escuela Profesional de Ingeniería de Sistemas

Acta Virtual de Sustentación
del Trabajo de Suficiencia Profesional

Siendo las 16:45 horas del día 8 de enero del año 2022, se reunieron virtualmente los docentes designados como Miembros de Jurado del Trabajo de Suficiencia Profesional, presidido por el Lic. Chávez Soto Jorge Luis (Presidente), Mg. Huapaya Chumpitaz Mario Agustín (Miembro) y el Lic. Espinoza Domínguez Robert (Miembro Asesor), usando la plataforma Meet (https://meet.google.com/jjy-yahj-fza), para la sustentación virtual del Trabajo de Suficiencia Profesional intitulado: ***"IMPLEMENTACIÓN DE UN MICROSERVICIO DE AUTENTICACIÓN CON UN IDENTITY SERVER PARA UNA APLICACIÓN DE BANCA MÓVIL DE UNA FINTECH AMERICANA",*** *por el Bachiller* ***Percca Julca Paul Cristian;*** *para obtener el Título Profesional de Ingeniero de Sistemas.*

Acto seguido de la exposición del Trabajo de Suficiencia Profesional, el Presidente invitó al Bachiller a dar las respuestas a las preguntas establecidas por los miembros del Jurado.

El Bachiller en el curso de sus intervenciones demostró pleno dominio del tema, al responder con acierto y fluidez a las observaciones y preguntas formuladas por los señores miembros del Jurado.

Finalmente habiéndose efectuado la calificación correspondiente por los miembros del Jurado, el Bachiller obtuvo la nota de ***18 DIECIOCHO.***

A continuación el Presidente de Jurados el Lic. Chávez Soto Jorge Luis, declara al Bachiller ***Ingeniero de Sistemas****.*

Siendo las 17:32. horas, se levantó la sesión.

Presidente
Lic. Chávez Soto Jorge Luis

Miembro
Mg. Huapaya Chumpitaz Mario Agustin

Miembro Asesor
Lic. Espinoza Domínguez Robert

Dedicatoria

Este trabajo se lo dedico a mi madre Carmen Julca Lavado por su apoyo incondicional y por el gran esfuerzo que hizo por mi, en mi educación, a mi padre Alfredo Percca Tarapa, que me guía desde el cielo, y a mi hermana Heidy, quienes han sido el soporte a lo largo de mi vida universitaria.

Agradecimiento

A mi familia por su apoyo constante.

A mi asesor Robert Espinoza por su paciencia y guía.

A las empresas donde he laborado, por darme la oportunidad de crecer como profesional.

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMATICA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Implementación de un microservicio de autenticación con un identity server para una aplicación de banca móvil de una fintech americana

Autor: **Percca Julca, Paul Cristian**

Asesor: **Espinoza Domínguez, Robert Elías**

Título: **Trabajo de Suficiencia Profesional para optar el Título Profesional de Ingeniero de Sistemas**

Fecha: **Enero del 2022**

Resumen

El presente trabajo de suficiencia profesional trata sobre el diseño e implementación de un microservicio que expone servicios Rest, a través de los cuales, los usuarios de una aplicación bancaria pueden autenticarse mediante un Identity Server, una vez el usuario está autenticado, el microservicio se encarga de sincronizar la autenticación con un monolito usando JSON Web Tokens, el microservicio fue desarrollado en Java, con Spring Boot como framework. Se llevó a cabo en una fintech americana cuyo producto es una aplicación móvil bancaria totalmente personalizable en Android y iOS, la cual, bancos de USA usan para brindar sus servicios financieros. El desarrollo de este proyecto se llevó a cabo usando la metodología ágil Scrum, en la cual el autor asumió el rol de Sr. Java Developer. La solución planteada fue implementada y cubrió el objetivo principal que reemplazar el anterior método de autenticación, para permitir al usuario tener una experiencia consistente de autenticación a través de las diversas aplicaciones web y móviles que la fintech ofrece.

Palabras claves: JWT, Identity Server, framework, Scrum.

NATIONAL UNIVERSITY OF SAN MARCOS FACULTY OF SYSTEMS ENGINEERING AND INFORMATICS

PROFESSIONAL SCHOOL OF SYSTEMS ENGINEERING

Implementation of a microservice of authentication with an identity server for a mobile banking application of an american fintech

Author: **Percca Julca, Paul Cristian**

Advisor: **Espinoza Domínguez, Robert Elías**

Title: **Work of Professional Sufficiency to opt for the Professional Title of Systems Engineer**

Date: **January del 2022**

Abstract

The present work of professional sufficiency deals with the design and implementation of a microservice that exposes Rest services, through which users of a banking application can authenticate through an Identity Server, once the user is authenticated, the microservice takes care of synchronize the authentication with a monolith using JSON Web Tokens, the microservice was developed in Java, with Spring Boot as a framework. It was carried out in an American fintech whose product is a fully customizable mobile banking application on Android and iOS, which US banks use to provide their financial services. The development of this project was carried out using the agile Scrum methodology, in which the author assumed the role of Mr. Java Developer. The proposed solution was implemented and covered the main objective of replacing the previous authentication method, to allow the user to have a consistent authentication experience through the various web and mobile applications that this fintech offers.

Keywords: JWT, Identity Server, framework, Scrum.

Índice general

Índice de figuras

Índice de tablas

Introducción

El presente Informe Profesional describe la implementación de un microservicio de Autenticación como primer paso de una migración en una aplicación bancaria, desde una arquitectura monolítica a una arquitectura orientada a microservicios.

La fintech americana para la cual se realizó el proyecto, inicialmente era una startup, sin embargo debido a su gran éxito y a la contratación de sus servicios, una aplicación móvil bancaria totalmente personalizable, por parte de bancos importantes de USA, hizo que esta arquitectura dejara de adecuarse a las necesidades del negocio, evitando que se realizaran escalamientos y haciendo cada vez más difícil el mantenimiento y la creación de nuevas features en este monolito. Por otro lado, esta fintech americana ofrece varios productos de software financieros, y para acceder a ellos, los usuarios de los bancos usaban credenciales distintas a los que se utilizaban en la aplicación móvil.

Es entonces que se decidió llevar a cabo la implementación de un microservicio de Autenticación, para permitir el escalamiento horizontal mediante la creación de instancias de este microservicio, de acuerdo a las necesidades del negocio, así como brindar una experiencia consistente de autenticación a los usuarios, usando las mismas credenciales en los diferentes productos que ofrece la compañía, como se describe en este documento.

La creación del microservicio mencionado anteriormente fue parte de una nueva funcionalidad de autenticación, y para ello realizaron cambios en las aplicaciones android y iOS, esta labor se llevó a cabo a través de un esfuerzo conjunto de los equipos de Globant y la fintech americana.

Este documento describe las actividades realizadas en el trabajo de suficiencia profesional y está estructurado de la siguiente manera:

En el Capítulo I se describe cronológicamente la trayectoria profesional del autor, los roles y funciones que realizó a lo largo de su carrera, así como la formación educativa.

En el Capítulo II se resume y contextualiza el entorno y se describe a Globant, la consultora donde se realizó este trabajo profesional. Incluye la misión, visión, organigrama de la empresa, experiencia profesional y funciones desempeñadas por el autor durante la implementación del proyecto.

En el Capítulo III se describe la situación problemática, la solución planteada, los objetivos definidos, así como las actividades realizadas para llevar a cabo este proyecto.

En el Capítulo IV, el autor presenta una reflexión crítica sobre su participación en el proyecto, así como los aportes realizados y las lecciones aprendidas creando un microservicio.

En el Capítulo V se describen las conclusiones y recomendaciones del autor en base a la experiencia adquirida trabajando en el proyecto en mención.

También se incluyen las fuentes de información y el glosario.

Nota: Dado que existe un acuerdo de confidencialidad entre el autor del informe y la consultora de software Globant, y ya que el proyecto es un producto financiero que usan decenas de bancos en USA, el nombre de la compañía para la cuál se desarrolló el proyecto se mantendrá en el anonimato en el presente informe.

Capítulo I
Trayectoria profesional

El autor es un Bachiller egresado de la Facultad de Ingeniería de Sistemas e Informática de la Universidad Nacional Mayor de San Marcos con más de 6 años de experiencia brindando soluciones de software y ayudando a empresas a escalar con la ayuda de la tecnología. Se ha especializado en el desarrollo de Software en la industria de finanzas, telecomunicaciones y comercio. Entre sus principales habilidades destacan la capacidad analítica, el liderazgo, el rápido aprendizaje, ser autodidacta, amante de los desafíos, ha participado en varios proyectos web y de aplicaciones móviles usando tecnologías Java y Cloud.

En la tabla 1, 2, 3 y 4, se describe la experiencia profesional del autor en las empresas Globant, Belatrix, Verizon y Applying Consulting respectivamente.

Tabla 1: Experiencia Profesional - Globant

Globant	
Agosto 2020 – Actualidad	
Cargo:	Sr. Java Developer
Funciones:	• Implementación de microservicios para un proyecto bancario que permite diferentes métodos de autenticación como autenticación de 2 pasos y autenticación biométrica, usando Java y Spring Boot. • Diseño e implementación de servicios Rest usando Open Api. • Implementación de cambios de mantenimiento en un proyecto monolítico para solucionar errores de producción.

Fuente: Elaboración propia

Tabla 2: Experiencia Profesional - Belatrix

Belatrix	
Diciembre 2018 – Agosto 2020	
Cargo:	Java Developer
Funciones:	• Implementación de microservicios para un proyecto bancario que permite a los usuario a abrir cuentas. bancarias, usando Java y Spring Boot. • Implementación de cambios de mantenimiento en un proyecto monolítico para solucionar errores de producción.

Fuente: Elaboración propia

Tabla 3: Experiencia Profesional - Verizon

Verizon	
Octubre 2017 – Noviembre 2018	
Cargo:	Full Stack Developer
Funciones:	• Liderar una migración de una aplicación frontend e-commerce de Angular JS 1.5 a Angular 5. • Migración del proceso de *deployment* de la aplicación e-commerce de Cloud Foundry a Aws EC2. • Implementación de nuevos reportes en la parte backend del e-commerce usando Spring Boot y Jasper Reports.

Fuente: Elaboración propia

Tabla 4: Experiencia Profesional - Applying Consulting

Applying Consulting	
Setiembre 2015 – Mayo 2017	
Cargo:	Full Stack Developer
Funciones:	• Diseño e implementación de aplicaciones móviles y web para clientes de diferentes industrias como Pandero, Inkafarma, P&G. • Diseño y modelado las bases de datos de estos proyectos. • Participación de la toma de requerimientos mediante reuniones de con los clientes.

Fuente: Elaboración propia

En la tabla 5, se describe la formación académica del autor como posgrado, pregrado y colegio.

Tabla 5: Formación Académica Profesional

Maestría en Informática con Mención en Ciencias de la Computación	
Institución:	Pontificia Universidad Católica del Perú Escuela de Posgrado
Periodo:	2018 - 2019
Bachiller en Ingeniería de Sistemas	
Institución:	Universidad Nacional Mayor de San Marcos Facultad de Ingeniería de Sistemas e Informática
Periodo:	2012 - 2017

Secundaria	
Institución:	I.E.P. Galileo Galilei
Periodo:	2006 - 2010

Fuente: Elaboración propia

En la tabla 6, se describe la formación complementaria del autor como certificaciones y cursos.

Tabla 6: Formación Académica Complementaria

Cloud Developer Nanodegree	
Institución:	Udacity
Periodo:	2021
iOS Developer Nanodegree	
Institución:	Udacity
Periodo:	2020
Android	
Institución:	Academia Móvil
Periodo:	2016

Fuente: Elaboración propia

Capítulo II
Contexto en el que se desarrolló la experiencia

2.1. Empresa – actividad que realiza

Globant es una empresa de tecnología y desarrollo de software argentina, considerada un unicornio, ya que su valuación supera los mil millones de dólares, es una empresa con más de 20000 profesionales y está presente en 18 países trabajando con compañías importantes a nivel mundial como Google, Rockwell Automation, Electronic Arts y Santander, entre otras.

2.2. Visión

"Queremos desafiar el status quo y convertirnos en la mejor empresa en la creación de viajes digitales, combinando lo mejor en ingeniería, innovación y diseño." (Comparably, 2021)

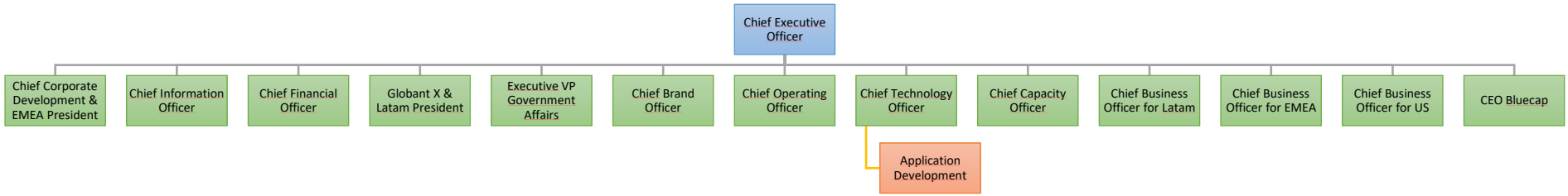
2.3. Misión

Globant está comprometido con la satisfacción del cliente continuando con la idoneidad y eficacia en la consecución de los objetivos de nuestra Política de Calidad. Esta política proporciona la base para establecer las metas para evaluar la efectividad de nuestro Sistema de Gestión de Calidad. (Comparably, 2021)

2.4. Organización de la empresa

En la figura 1 se muestra el organigrama ejecutivo de Globant, el autor forma parte del área de Application Development.

Figura 1: Organigrama de Globant



Fuente: Elaboración propia

2.5. Área, Cargo y Funciones desempeñadas

El autor de este trabajo de suficiencia profesional se desempeñó el rol de Sr. Java Developer en el equipo de Authentication and Fraud Control del área de desarrollo de aplicaciones de Globant.

Las funciones desempañadas se muestran a continuación.

- Apoyo al tech leader en la investigación del funcionamiento del proyecto, para el diseño de la nueva solución.
- Diseño e implementación de código en los microservicios.
- Participar en los Code Review del equipo.
- Realizar cambios en la aplicación monolítica.
- Realizar tareas de mantenimiento y solución de bugs en las aplicaciones.
- Entrenar a los nuevos miembros del equipo en la explicación del proyecto.
- Bridar demos internas y externas de los avances del proyecto.

2.6. Experiencia profesional realizada en la organización

El autor realizó múltiples iniciativas dentro de la organización tales como:

- Creación de documentación y diagramas de la arquitectura de la aplicación.
- Ingeniería inversa de la base de datos del proyecto, para una mejor visibilidad.
- Implementación de proyectos de integration testing, para las diferentes features del proyecto.
- Implementación de un microservicio para la detección de fraude usando un producto de Mastercard.
- Implementación de un microservicio para la apertura de cuentas bancarias desde la app móvil.

Capítulo III
Actividades desarrolladas

3.1. Situación problemática

La fintech americana tiene como producto una aplicación bancaria móvil, los clientes de esta fintech son bancos de USA, la aplicación posee diferentes features como son Online Account Opening (que permite a los clientes nuevos abrirse una cuenta desde la aplicación), Biometric Authentication, (que permite la autenticación del usuario usando Touch Id y Face Id en caso de dispositivos iOS o Fingerprint en caso de Android) , Apple Pay (que integra la aplicación móvil con la billetera nativa de Apple), entre otras sin embargo esta aplicación presenta una arquitectura monolítica, que cuenta ya con más de 10 años, esta arquitectura ha generado muchos problemas de dependencia, por lo cual se ha decidido migrarla a una arquitectura de micro servicios, y la primera feature a ser migrada fue la de Authenticación.

En esta arquitectura existe un intermediario entre los bancos y el servidor monolito, llamada HUB, las credenciales de los usuarios no son guardadas en la base de datos del servidor monolito sino que la poseen los bancos, con la finalidad de eliminar el acoplamiento que existe y la gran dependencia del servidor monolito por el HUB, se ha decidido usar un Identity Server para la autenticación, de manera que se brinde una experiencia consistente de autenticación en las diferentes plataformas que esta fintech ofrece.

3.1.1. Definición del Problema

El principal problema radica en que se tiene una aplicación bancaria con alto grado de acoplamiento y dependencia dentro de su arquitectura, lo cual conlleva a que no se pueda realizar un escalamiento horizontal.

3.2. Solución

Entonces, para reducir el acoplamiento de la aplicación, se implementó un microservicio, que maneja la autenticación del usuario, y

que brinda una experiencia similar a las de las otras plataformas a los clientes, logrando el primer paso para la migración total de la arquitectura monolítica a orientada a microservicios.

3.2.1. Objetivos

A continuación, se describen el objetivo general y los objetivos específicos del proyecto.

3.2.1.1. General

Implementar un microservicio que permita el desacoplamiento de la feature de Autenticación del monolito.

3.2.1.2. Específicos

- Integrar un Identity Server para la autenticación de la aplicación bancaria.
- Permitir nuevos métodos de autenticación como son OTP y authenticator, que el Identity Server ofrece.
- Brindar una experiencia consistente de autenticación al usuario entre las diferentes plataformas que la fintech ofrece.
- Sincronizar la autenticación del usuario tanto en el microservicio como en el monolito.

3.2.2. Alcance Funcional

El proyecto solo contempla la migración del microservicio de Autenticación, la integración de los servicios Rest para la autenticación con el Identity Server, la sincronización a través de un JWT con el servidor monolito y el HUB, así como la Autenticación biométrica con Touch Id y Face Id en caso de dispositivos iOS y Fingerprint en el caso de

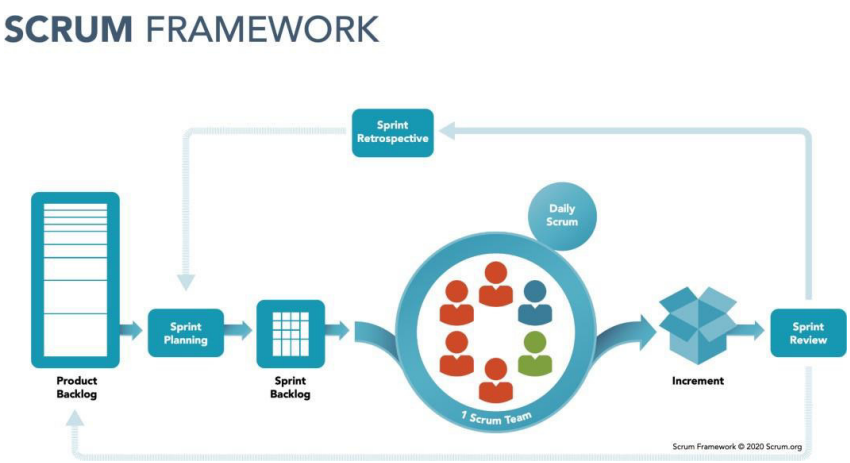
dispositivos Android. No se contempla la detección de Fraudes durante el proceso de autenticación, ya que para ello existe otro microservicio, no tratado en el presente proyecto.

3.2.3. Etapas y metodología

Durante la implementación de este proyecto se utilizó la metodología ágil Scrum.

En la figura 2 se muestra el flujo de trabajo de la metodología Scrum.

Figura 2: Flujo de Trabajo Scrum



Fuente: Adaptado de Scrum.org (Scrum, 2021)

De acuerdo con el diagrama se realizaron las siguientes tareas en cada una de las etapas:

3.2.3.1. Inicio

Durante esta etapa se identificó al equipo scrum, al Product Owner y Scrum Master. Se llevo a cabo una planificación a grandes rasgos de cómo se desarrollarían los sprints.

3.2.3.2. Planificación y Estimación

Se llevaron a cabo las reuniones de planificación al iniciar cada sprint, en el cual se realizó la estimación de las historias de usuario y tareas usando planning poker.

3.2.3.3. Implementación

Se llevaron a cabo la implementación de las historias de usuario, para llevar un control de las tareas e identificar dependencias o historias bloqueadas, se llevó a cabo daily meetings de 15 minutos, donde los miembros del equipo, comentaban el status de las historias, y si se encontraban bloqueados.

3.2.3.4. Revisión y Retrospectiva

Al finalizar cada sprint se llevaba a cabo la Review y Retrospective meeting, ambas por separado, en la Review meeting se realizaba una demo con el Product Owner donde se mostraba el avance del equipo durante el sprint, en la Retrospective meeting, se mencionaba aspectos que sucedieron durante el sprint, aspectos que deberían continuar sucediendo y aspecto a tratar de eliminar para el mejor performance del equipo.

3.2.3.5. Lanzamiento

Durante esta etapa se llevó a cabo de despliegue de la aplicación a un ambiente de UAT, donde los QAs probarían la aplicación y se daría la certificación de calidad.

3.2.4. Fundamentos Utilizados

3.2.4.1. Arquitectura Monolítica

La arquitectura monolítica es la estructura tradicional de las aplicaciones de software, donde todas las funcionalidades de un proyecto existen en una única base de código, donde todos los aspectos del software operan como una sola unidad. Las grandes desventajas de seguir esta arquitectura es el gran acoplamiento que existe en cada uno de sus módulos, lo que conlleva a que se vuelva difícil de mantener, además de que es necesario desplegar toda la aplicación, incluso para un pequeño cambio, y que, si una sola parte de la aplicación se enfrenta a una gran carga de tráfico, se necesita desplegar instancias de toda la aplicación en varios servidores, lo cual es ineficiente ya que consume recursos innecesariamente. (Geeks for Geeks, 2021)

3.2.4.2. Arquitectura de Microservicios

Los microservicios son un tipo de arquitectura usada para diseñar aplicaciones. A diferencia de las arquitecturas tradicionales y monolíticas, los microservicios desglosan la aplicación en funciones principales. Donde cada función es un servicio, que pueden ser diseñados e implementados independientemente. Lo que permite que funcionen separados y que de haber algún fallo, fallen por separado sin afectar a los demás servicios. (Red Hat, 2021)

3.2.4.3. Escalamiento Horizontal

El escalado horizontal se refiere al aprovisionamiento de servidores adicionales para satisfacer sus necesidades, a menudo dividiendo las cargas de trabajo entre servidores para limitar la cantidad de solicitudes que recibe cualquier servidor individual. El escalado horizontal en la computación en la nube significa agregar instancias adicionales en lugar de pasar a un tamaño de instancia mayor. (CloudCheckr, 2021)

3.2.4.4. Escalamiento Vertical

El escalamiento vertical se refiere a agregar más CPU, memoria o recursos de E / S a un servidor existente o reemplazar un servidor por uno más potente. En un centro de datos, los administradores lograban tradicionalmente el escalado vertical comprando un servidor nuevo y más potente y descartando o reutilizando el anterior. Los arquitectos de la nube de hoy pueden lograr el escalado vertical de AWS y el escalado vertical de Microsoft Azure cambiando el tamaño de las instancias. Los servicios en la nube de AWS y Azure tienen muchos tamaños de instancia diferentes, por lo que el escalado vertical en la computación en la nube es posible para todo, desde las instancias EC2 hasta las bases de datos RDS. (CloudCheckr, 2021)

3.2.4.5. OAuth

OAuth es un estándar que las aplicaciones pueden utilizar para proporcionar a las aplicaciones cliente un "acceso delegado seguro". OAuth funciona a través de HTTPS y autoriza dispositivos, API, servidores y aplicaciones con tokens de acceso en lugar de credenciales.

OAuth es una forma de obtener acceso a datos protegidos desde una aplicación. Es más seguro que pedir a los usuarios que inicien sesión con contraseñas. (Oauth, 2021)

3.2.4.6. Identity Server

IdentityServer es un servidor de autenticación que implementa los estándares OpenID Connect (OIDC) y OAuth 2.0 para ASP.NET Core. Está diseñado para proporcionar una forma común de autenticar solicitudes a todas sus aplicaciones, ya sean terminales web, nativas, móviles o API. IdentityServer se puede utilizar para implementar el inicio de sesión único (SSO) para múltiples aplicaciones y tipos de aplicaciones. Se puede utilizar para autenticar a los usuarios reales a través de

formularios de inicio de sesión e interfaces de usuario similares, así como autenticación basada en servicios que generalmente implica la emisión, verificación y renovación de tokens sin ninguna interfaz de usuario. IdentityServer está diseñado para ser una solución personalizable. Por lo general, cada instancia se personaliza para adaptarse a las necesidades de una organización individual y / o un conjunto de aplicaciones. (Microsoft, 2021)

3.2.4.7. JWT

JSON Web Token (JWT) es un estándar abierto (RFC 7519) que define una forma compacta y autónoma de transmitir información de forma segura entre las partes como un objeto JSON. Esta información se puede verificar y confiar porque está firmada digitalmente. Los JWT se pueden firmar usando un secreto (con el algoritmo HMAC) o un par de claves pública / privada usando RSA o ECDSA. (JWT, 2021).

3.2.4.8. Key Pair

Una llave privada y una llave pública son parte del cifrado que codifica la información. Ambas claves funcionan en dos sistemas de cifrado llamados simétricos y asimétricos. El cifrado simétrico (cifrado de clave privada o cifrado de clave secreta) utiliza la misma clave para cifrado y descifrado. El cifrado asimétrico utiliza un par de claves como la clave pública y la privada para una mayor seguridad donde el remitente del mensaje cifra el mensaje con la clave pública y el receptor lo descifra con su clave privada. (ScienceDirect, 2021)

3.2.4.9. Scrum

Scrum es un marco ligero que ayuda a las personas, los equipos y las organizaciones a generar valor a través de soluciones adaptables para problemas complejos. Los co-creadores de Scrum, Ken Schwaber y Jeff

Sutherland, han escrito The Scrum Guide para explicar Scrum de manera clara y sucinta. Esta guía contiene la definición de Scrum. Esta definición consta de las responsabilidades, los eventos, los artefactos y las reglas que los unen de Scrum. (Scrum, 2021).

3.2.5. Implementación de las áreas, procesos, sistemas y buenas prácticas

Las actividades que se realizaron en cada Sprint se detallaran a continuación.

3.2.5.1. Numero de Sprints

El proyecto se realizó en 6 sprints.

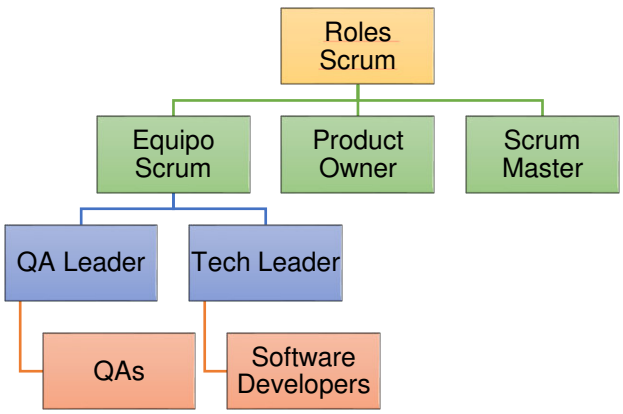
3.2.5.2. Duración de cada sprint

Cada sprint tuvo una duración de 2 semanas.

3.2.5.3. Roles Scrum involucrados

En la figura 3 se muestra el equipo Scrum y los roles que lo conforman.

Figura 3: Roles Scrum



Fuente: Elaboración propia

El equipo fue multicultural, contando con miembros de Argentina, Perú, Estados Unidos y Filipinas.

En la figura 7 se muestra los miembros del equipo y sus respectivas nacionalidades.

Tabla 7: Equipo de Desarrollo

Rol	Número de Personas
Product Owner	1 (Estados Unidos)
Scrum Master	1 (Argentina)
Tech Leader	1 (Argentina)
QA Leader	1 (Argentina)
Java Developers	1 (Perú) 1(Filipinas)
QAs	2 (Argentina) 1 (Perú)
iOS Developers	2 (Argentina) 1 (Perú)
Android Developers	2 (Argentina) 1 (Perú)

Fuente: Elaboración propia

3.2.5.4. Cronograma del proyecto

En la tabla 8 se muestra el cronograma del proyecto, con los sprints realizados, desde el 1 de junio del 2020 al 21 de agosto del 2020.

Tabla 8: Cronograma del proyecto

Sprint	Fecha Inicio	Fecha Fin
Sprint 1	01/06/2020	12/06/2020
Sprint 2	15/06/2020	26/06/2020
Sprint 3	29/06/2020	10/07/2020
Sprint 4	13/07/2021	24/07/2021
Sprint 5	27/07/2021	07/08/2021
Sprint 6	10/08/2021	21/08/2021

Fuente: Elaboración propia

3.2.5.5. Product Backlog

En la tabla 9 se muestra la lista de las historias de usuario, sprint por sprint, donde se identifican las tareas de creación de endpoints para las funcionalidades de autenticación con credenciales y autenticación biométrica.

Tabla 9: Historias de Usuario por Sprint

Sprint	Historia de Usuario
Sprint 1	• Realizar la creación del Proyecto Spring Boot como microservicio y definición de endpoints. • Implementar el endpoint para obtener iniciar el flujo de autenticación y obtener la configuración de la institución financiera (/start).

	• Implementar el endpoint de autenticación con credenciales completas, usuario y password (/creds).
Sprint 2	• Implementar la funcionalidad de recordar usuario. • Implementar cambios en el monolito para la sincronización de la sesión, luego de una autenticación con credenciales exitosa. • Implementar la funcionalidad de cierre de sesión para el Identity Server y el monolito (/logout).
Sprint 3	• Implementar el endpoint para aceptar términos y condiciones (/terms). • Implementar el endpoint para seleccionar el dispositivo al cual se enviará el codigo OTP (otpchoice). • Implementar el endpoint para validar el código OTP (otpvalidation).
Sprint 4	• Implementar la funcionalidad de configuración del key del authenticator • Realizar cambios en el endpoint de validar el OTP, para validar el código del authenticator.
Sprint 5	• Implementar el endpoint de habilitar/ deshabilitar la autenticación biométrica (/setupbiometric). • Implementar el endpoint de inicio de la autenticación biométrica (/startdevice). • Implementar el endpoint de fin y validación de la autenticación biométrica (/enddevice).

Sprint 6	• Refrescar el token de autenticación del Identity Server, luego de una autenticación biométrica exitosa. • Implementar cambios en el monolito para la sincronización de la sesión, luego de una autenticación biométrica exitosa.

Fuente: Elaboración propia

3.2.5.6. Sprints

A continuación, se muestra el trabajo realizado en cada uno de los sprints.

3.2.5.6.1. Sprint 1

En el primer Sprint se implementaron las siguientes historias de usuario:

- **Realizar la creación del Proyecto Spring Boot como microservicio y definición de endpoints.**

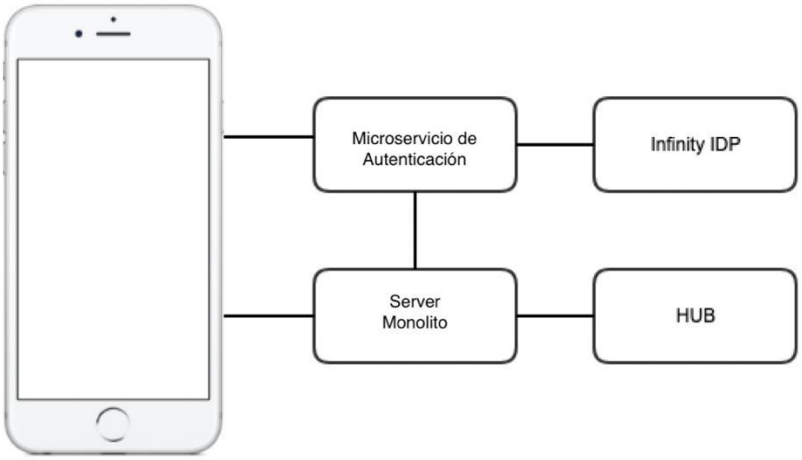
Se realizó la creación de un repositorio usando Git como gestor de dependencias en Bitbucker como se observa en el Anexo 2, se creó un proyecto usando Java 8 y Sprint Boot 2.4, la configuración para el despliegue en containers de Openshift se realizó usando Helms para la configuración de secrets, Jenkins como herramienta de CI/CD como se observa en el Anexo 3 y Maven como gestor de dependencias.

Durante este proceso, se llevó a cabo una revisión de la documentación del Identity Server, donde se encontró un wizzard con los endpoints que este Identity Server proveía, la secuencia de llamadas e identificación de parámetros a enviar en cada paso de la autenticación.

Se definió los endpoints usando Open Api 3 para la documentación de estos, se tomó en consideración los parámetros de entrada y salida, así como los mensajes de error en caso de errores no esperados.

En la figura 4, se muestra la arquitectura de la aplicación, se observa la interacción entre el Microservicio, la aplicación cliente, el Identity Provider, y el Monolito.

Figura 4: Arquitectura del Microservicio de Autenticación.



Fuente: Elaboración propia

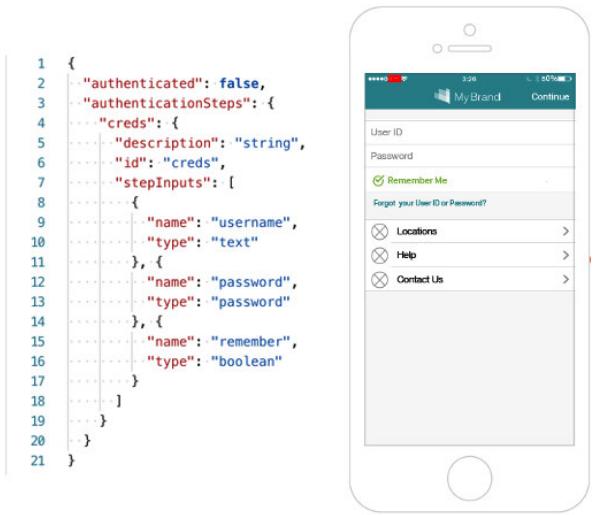
- **Implementar el endpoint para obtener iniciar el flujo de autenticación y obtener la configuración de la institución financiera (/start).**

Se implementó un endpoint authentication/start, cuyo propósito era indicar los campos a mostrar en la pantalla de autenticación en el dispositivo móvil, y si conocer la configuración de autenticación de la institución financiera, por ejemplo, la opción de autenticación biométrica o si la funcionalidad de recordar usuario estaba habilitada para mostrar un dichas opciones en la pantalla de login, para obtener esta información se realizaba una llamada al Identity Server.

Este endpoint, no recibía parámetros y retornaba los campos necesarios para el siguiente paso de autenticación, como se muestra acontinuación, donde la funcionalidad de recordar usuario está habilitada, y se solicita que se envíe el usuario, la contraseña, y si el usuario decidió recordar su username.

En la figura 5, se muestra el response del endpoint authentication/start, así como la pantalla correspondiente en la aplicación móvil, donde se visualiza los campos de entrada, username, password y la opción de recordar usuario como un toggle.

Figura 5: Response del Endpoint de Inicio de Autenticación



Fuente: Elaboración propia

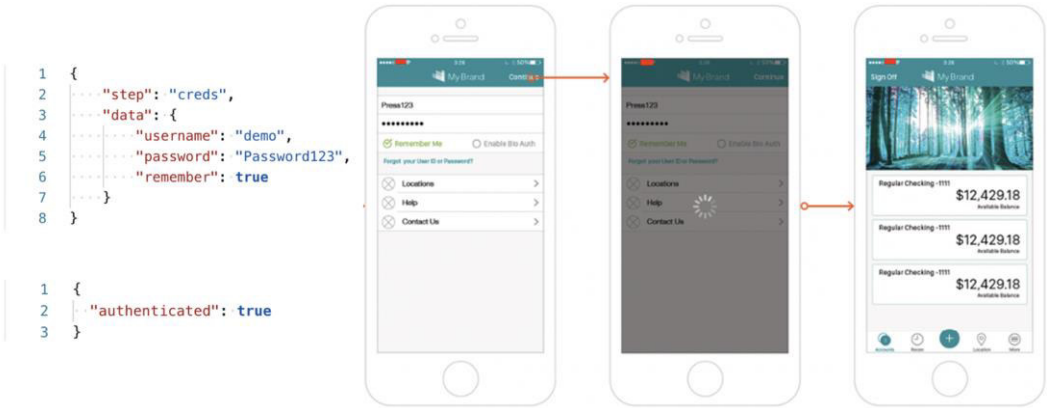
- **Implementar el endpoint de autenticación con credenciales completas, usuario y password (/creds).**

Se implementó un endpoint authentication/creds, donde se recibían las credenciales del usuario, username y password, así como si el usuario había decidido recordar el username. Dado que la seguridad es un factor importante y más teniendo en cuenta que era una aplicación bancaria, se implementó un servicio de encriptación y desencriptación

donde el usuario y password viajan encriptados de la aplicación móvil al microservicio.

En la figura 6 se observa el request del endpoint authentication/creds, así como la pantalla correspondiente en la aplicación móvil, donde se visualiza el envío de los datos solicitados como, username, password y la opción de recordar usuario, así como el response, el cual indica que los valores han sido correctos y la aplicación dirige al usuario hacia la pantalla del dashboard, donde observa sus cuentas bancarias.

Figura 6: Request y Response del Endpoint de Autenticación con Credenciales



Fuente: Elaboración propia

Estos valores eran enviados al Identity Server, para su respectiva validación, en el caso que la configuración del Identity Server solo requería credenciales y que los mismos eran correctos, se retornaba una propiedad authenticated, que indicaba que la autenticación había sido exitosa.

Hasta este sprint no se había implementado la funcionalidad de sincronización de la sesión con el monolito, ya que esa tarea fue trabajada en el siguiente sprint.

Para mayor detalle del flujo de autenticación y el flujo de pantallas, revisar el Anexo 1, Flujo 1: Autenticación con Credenciales**.**

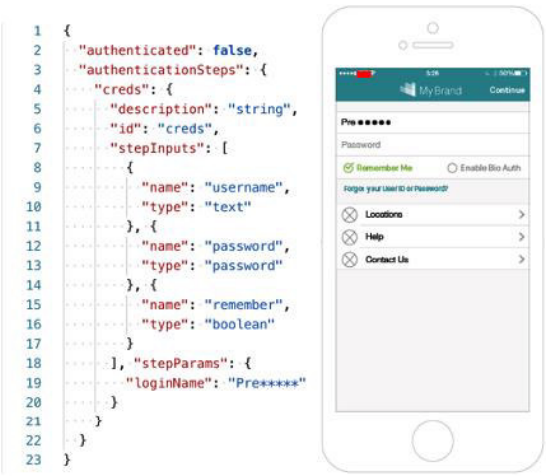
3.2.5.6.2. Sprint 2

- **Implementar la funcionalidad de recordar usuario.**

El Identity Server posee una configuración para permitir al usuario recordar su username, para lo cual en la pantalla de autenticación existe un checkbox, este valor es enviado junto con las credenciales al Identity Server, después de una autenticación exitosa, este valor queda recordado en el Identity Server, y en el siguiente intento de autenticación, el /start retornará un parámetro del paso con el valor del username parcialmente oculto como se muestra a continuación.

En la figura 7, se muestra el response del endpoint authentication/start, con la funcionalidad del usuario recordado habilitado, así como la pantalla correspondiente en la aplicación móvil, donde se visualiza los campos de entrada, username precompletado, password y la opción de recordar usuario marcado.

Figura 7: Response del Endpoint start con username recordado



Fuente: Elaboración propia

- **Implementar cambios en el monolito para la sincronización de la sesión, luego de una autenticación con credenciales exitosa.**

  Luego de una autenticación exitosa por medio del microservicio y del Identity Server, se realizó una llamada al Identity Server para obtener un JWT, dentro del cual, se encontraban algunos campos con información del usuario como nombre, correo electrónico, un access token del Identity Provider que era necesario enviar en las siguientes interacciones entre el Microservicio y el Identity Provider y un campo externalUserId, el cual existía en el HUB, intermediario entre el monolito y los sitemas de los bancos.

  En el monolito se implementó un nuevo servicio rest /syncSession mediante el cual se recibía el JWT, para que fuera enviado al HUB, donde se extría la información del JWT y se compara el externalUserId con un campo interno del HUB, indentificando al usuario que se había autenticado, posteriormente se creaba una sesión, retornando un Access Token del Monolito que era enviado finalmente a las aplicaciones cliente, android y iOS, el cual era usado para las siguientes interacciones entre las aplicaciones móviles y el Monolito para el uso de las diferentes *features*.

- **Implementar la funcionalidad de cierre de sesión para el Identity Server y el monolito (/logout).**

  Se implementó un endpoint authentication/logout en el microservicio, donde se recibía cookies como headers mediante el cual se podía identificar desde que dispositivo y que usuario deseaba cerrar sesión.

  Para el cierre de sesión, se llamó a un servicio rest del Identity Server, y se llamó a otro servicio rest del monolito, de manera que se tenga sincronizada el cierre de sesión.

3.2.5.6.3. Sprint 3

- **Implementar el endpoint para aceptar términos y condiciones (/terms).**

El Identity Server poseía la funcionalidad de términos y condiciones, por lo cual implementó una lógica para manejar este posible paso de autenticación por parte del Identity Server. Estos términos y condiciones como parametros del paso, y dicha información era mostrada en el dispositivo.

Adicionalmente, se implementó un endpoint authentication/terms, mediante el cual el usuario debía aceptar los terminos y condiciones que se habían configurado en el Identity Server, la aceptación era enviada al Identity Server a través de un servicio rest.

En la figura 8, se muestra el response del endpoint authentication/creds, en caso existan términos por aceptar.

Figura 8: Response con T&C como siguiente paso de autenticación

Fuente: Elaboración propia

- **Implementar el endpoint para seleccionar el dispositivo al cual se enviará el codigo OTP (otpchoice).**

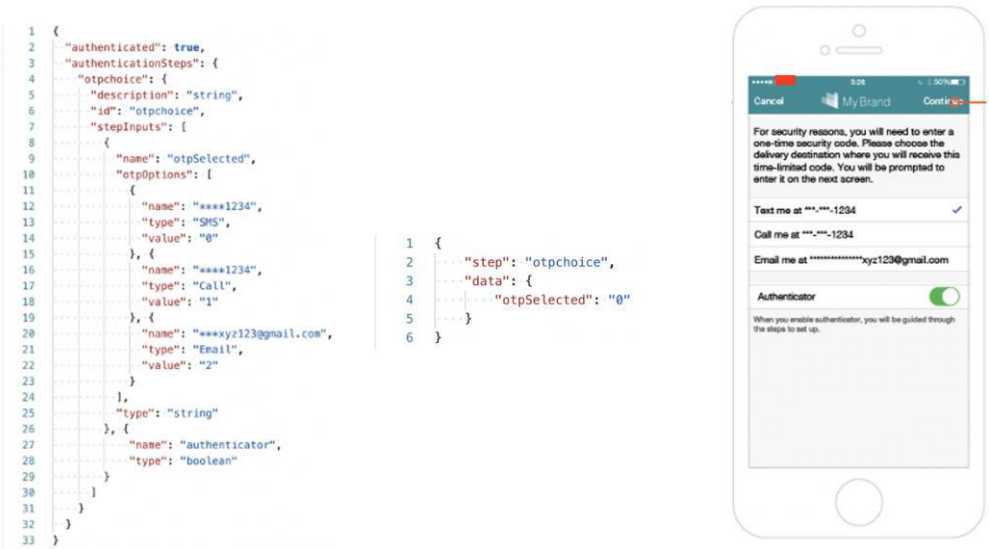
El Identity Server tenía una funcionalidad de autenticación en dos pasos, con OTP, para brindar mayor seguridad. Cuando esta

funcionalidad estaba habilitada, luego de enviar el username y password del usuario, el Identity Server devolvía un listado de opciones, canales, para enviar el código OTP, entre estas se encontraban SMS, llamada y correo electrónico, como se observa en la figura.

Se implementó un endpoint authentication/otpchoice en el microservicio para seleccionar una de estas opciones mediante el cual se enviaría el código OTP. Luego de que el microservicio recibía la opción seleccionada, este valor era enviada al Identity Server y este enviaba el OTP code a través del canal seleccionado.

En la figura 9, se muestra el response del endpoint authentication/creds, en el caso que la funcionalidad de OTP esté habilitada, retornando el listado de dispositivos, y la pantalla de la aplicación correspondiente, además del request de authentication/otpchoice, donde se envía el dispositivo seleccionado al cual se enviará el código OTP.

Figura 9: Request y Response del Endpoint otpchoice



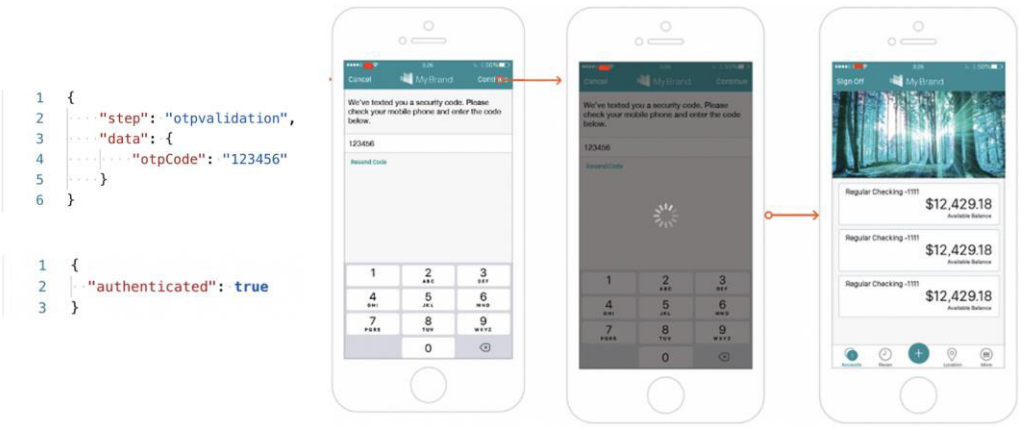
Fuente: Elaboración propia

- **Implementar el endpoint para validar el código OTP (otpvalidation).**

  Se implementó un endpoint authentication/otpvalidation para validar el código OTP ingresado por el usuario, este valor era enviando al Identity Server, para su respectiva validación, en el caso de que el código OTP ingresado por el usuario era correcto, el flujo de autenticación concluía satisfactoriamente.

  En la figura 10, se muestra el request y response del endpoint authentication/otpvalidation, donde se envía el código OTP recbido y en el caso de que sea el correcto, se redirigirá al usuario al dashboard con las cuentas bancarias del usuario.

Figura 10: Request y Response del Endpoint de Validación del OTP



Fuente: Elaboración propia

Para mayor detalle del flujo de autenticación con código OTP y el flujo de pantallas, revisar el Anexo 1, Flujo 1: Autenticación de dos pasos con OTP.

3.2.5.6.4. Sprint 4

- **Implementar la funcionalidad de configuración del key del authenticator.**

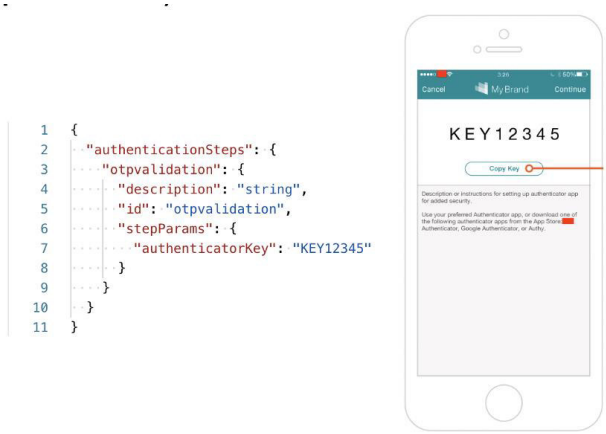
  El Identity Server contaba con la funcionalidad de autenticación de dos pasos, usando un authenticator, en el caso de que el Identity

Server tenía habilitada esta funcionalidad, y luego de la autenticación con username y password, el Identity Server le devolvía al microservicio un key, este valor era retornado a los dispositivos android y iOS.

Esta key era necesaria para configurar el authenticator en alguna aplicación de authenticator como Google Authenticator o Authy.

En la figura 11, se muestra el response del endpoint authentication/otpvalidation con el key del authenticator, en el caso el usuario haya habilitado la configuración del authenticator, en la pantalla de selección de dispositivos y la pantalla correspondiente en la aplicación donde se muestra la key.

Figura 11: Response para la Configuración del Authenticator



Fuente: Elaboración propia

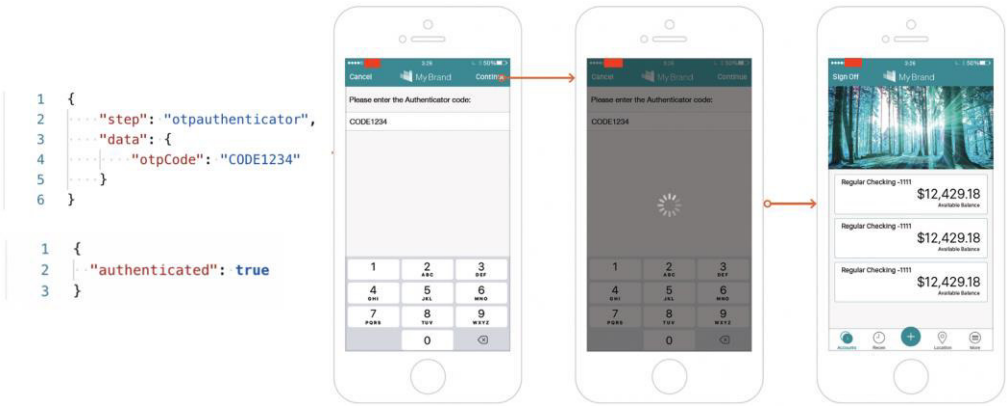
- **Realizar cambios en el endpoint de validación del OTP, para validar el código del authenticator.**

Para la asegurar, que la configuración del authenticator fue realizada correctamente por el usuario, era necesario validar un código authenticator ingresado por el usuario, por lo cual, se realizó una modificación en el endpoint authentication/otpvalidation para soportar la validación del código authenticator, una vez que el microservicio recibía

este código, era enviado al Identity Server, y en el caso que la validación era correcta, el flujo de autenticación concluía exitosamente.

En la figura 12, se muestra el request y el response del endpoint authentication/otpvalidation, al enviar el código authenticator para su respectiva validación, y el resultado en el caso el código fue el correcto, además de la respectiva pantalla en la aplicación.

Figura 12: Request y Response para la Validación del código Authenticator



Fuente: Elaboración propia

Para mayor detalle del flujo de autenticación con código authenticator y el flujo de pantallas, revisar el Anexo 1, Flujo 2: Autenticación de dos pasos con OTP, configurando el Authenticator y Flujo 3: Autenticación de dos pasos con Authenticator.

3.2.5.6.5. Sprint 5

- **Implementar el endpoint de habilitar/ deshabilitar la autenticación biométrica (/setupbiometric).**

  Se implementó un endpoint authentication/setupbiometric para habilitar/deshabilitar la autenticación biométrica desde el dispositivo

móvil, se llevó a cabo a través de la generación de un key pair, una llave pública y privada, este key pair era creado en la aplicación cliente (Android y iOS), la llave pública era enviada al microservicio a través del endpoint mencionado anteriormente, y guardado en Datagrid, mientras que la llave privada era guardada en el storage de la aplicación móvil.

En la figura 13, se muestra el request del endpoint authentication/setupbiometric, mediante el cuál se envía la llave pública al microservicio.

Figura 13: Request para habilitar la Autenticación Biométrica


```
{
  "publicKey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDfqL1c3MIAlfOvbNwG41eCcGP9Wi2fJLEi5PEkTvNVn+Id\/Pyz6P5nQb+LLgXzXq6vZAAJPnFn4ayUzWX3oe0lCvBP6Fn4bQIXua\/
  jJ2KPsO0uIhwPDZyeoRpGreeDHPXBb3IDHQnWvvKqJ3MIz5VAclQRZOFEQqnGUcQNl1x+PQIDAQAB"
}
```


Fuente: Elaboración propia

- **Implementar el endpoint de inicio de la autenticación biométrica (/startdevice).**

Cuando el usuario intentaba autenticarse con autenticación biométrica ya sea Touch Id, Face Id o Fingerprint, el dispositivo móvil se encargaba de verificar al usuario, sin embargo, había un paso más de autenticación. Una vez el usuario era reconocido biométricamente para la aplicación móvil, se realizaba una llamada al microservicio, para iniciar la autenticación biométrica en él, este paso se llevó a acabo a través del endpoint authentication/startdevice.

Para que el microservicio asegurara la autenticación del usuario, se generó un valor random, y se encriptó con la llave pública previamente enviada por el cliente al habilitar la autenticación biométrica, este valor random encriptado o nonce, era retornado a cliente para que se realice el desencriptado en la aplicación cliente usando la llave privada.

En la figura 14, se muestra el request del endpoint authentication/startdevice, donde se envía el tipo de autenticación

biométrica y el response, donde se retorna el valor nonce, un valor aleatorio encriptado en el microservicio con la llave pública.

Figura 14: Request y Response del inicio de la Autenticación Biométrica

```
{
    "step": "startdevice",
    "data": {
        "biometricType": "Touchid"
    }
}
```

```
{
  "authenticationSteps": {
    "enddevice": {
      "id": "enddevice",
      "stepInputs": [
        {
          "name": "deviceNonce",
          "type": "text"
        }
      ],
      "stepParams": {
        "nonce": "EkTvNVn+Id\/Pyz6P5nQb+LLgXzXq6vZAAJPnFn4ayUzWX3oe"
      }
    }
  }
}
```

Fuente: Elaboración propia

- **Implementar el endpoint de fin y validación de la autenticación biométrica (/enddevice).**

Se implementó un endpoint authentication/enddevice en el microservicio para recibir el valor random desencriptado en la aplicación móvil usando la llave privada almacenada en el storage, una vez este valor era enviado al microservicio, se realizaba la comparación de ambos valores, el generado por el microservicio y el descriptado por la aplicación móvil, en caso ambos valores eran iguales, la autenticación biométrica se concluía exitosamente.

En la figura 15, se muestra el request del endpoint authentication/enddevice, donde se envía el valor desencriptado, para su respectiva validación en el microservicio, y el response en el caso de que la comparación sea exitosa.

Figura 15: Request y Response del fin de la Autenticación Biométrica

```
{
    "step": "enddevice",
    "data": {
        "deviceNonce": "1234"
    }
}
```

```
{
    "authenticated": true
}
```

Fuente: Elaboración propia

Para mayor detalle del flujo de autenticación biométrica y el flujo de pantallas, revisar el Anexo 1, **Flujo 2:** Autenticación configurando la Autenticación Biométrica.

3.2.5.6.6. Sprint 6

- **Refrescar el token de autenticación del Identity Server, luego de una autenticación biométrica exitosa.**

  Se implementó una llamada desde el microservicio al Identity Server para refrescar la sesión usando el access token que se encontraba en el JWT, obtenido anteriormente durante la autenticación con credenciales username y passowrd.

- **Implementar cambios en el monolito para la sincronización de la sesión, luego de una autenticación biométrica exitosa.**

  Se realizaron cambios en el monolito, en el endpoint /syncSession para recibir un parámetro biometricType, donde se indicaría si la autenticación biométrica se realizó con Touch Id, Face Id o fingerprint. Así mismo se implementaron otros cambios para enviar esta información al HUB, y asignar valores al monolito como los auth levels, niveles de autenticación, ya que ciertas funcionalidades de la aplicación bancaria

estaban habilitadas o deshabilitadas según el nivel de autenticación del usuario, con credenciales o biométrico.

3.3. Evaluación

3.3.1. Evaluación Económica / Evaluación Costo – Beneficio

Dado que este proyecto, fue realizado por una consultora de software y al ser un proyecto bancario, el costo del proyecto es una información confidencial, sin embargo en la Tabla 10, se muestran los costos aproximados del capital humano, tomando en cuenta la cantidad de personas por cada rol.

Tabla 10: Costo de Capital Humano

Rol	Nº de Personas	Tarifa HH	Monto
Product Owner	1	$ 60.00	$ 30,240.00
Scrum Master	1	$ 40.00	$ 20,160.00
Tech Leader	1	$ 40.00	$ 20,160.00
QA Leader	1	$ 40.00	$ 20,160.00
Java Devs	2	$ 38.00	$ 38,304.00
QAs	3	$ 32.00	$ 48,384.00
iOS Devs	3	$ 38.00	$ 57,456.00
Android Devs	3	$ 35.00	$ 52,920.00
		Total	$ 287,784.00

Fuente: Elaboración propia

Como se puede apreciar, la inversión total aproximada en capital humano, fue de $ 287,784.00; sin embargo el Product Owner y un desarrollador Java eran recursos del cliente, lo cual conlleva a un ingreso de $ 238,392.00 para Globant.

El beneficio obtenido por este proyecto fue dar el primer paso para la migración de esta arquitectura monolítica a una arquitectura orientada a microservicios, y permitio a la fintech tener más crontrol sobre sus funcionalidades, de manera que, podían ser escaladas independientemente, acorde a las necesidades del negocio.

Posteriormente y debido al éxito del proyecto, se encargó a Globant la implementación de un nuevo microservicio, donde se integraba un producto de Mastercard, Nudetect, para la detección de actitudes sospechosas en la pantalla de login de la aplicación por parte del usuario.

Capítulo IV
Reflexión crítica

4.1. Aportes/lecciones aprendidas/ en el qué se puede mejorar

4.1.1. Aportes

En cuanto al aporte del autor en el desarrollo de este proyecto, se puede mencionar lo siguiente:

- Creación de servicios Rest usados por las aplicaciones clientes iOS y Android para el flujo de autenticación.
- Sincronización de la autenticación del usuario entre microservicios y el servidor monolito.
- Integración del Identity Server a través de llamadas Rest, desde el microservicio.
- Verificación de la autenticación biométrica a través de un key pair.
- Despliegue del microservicio como un servicio en Openshift, permitiendo agregar instancias del servicio de acuerdo a las necesidades de negocio.

4.1.2. Lecciones aprendidas

Luego de la implementación del proyecto en mención, el autor de este documento puede mencionar las siguientes:

- Logró conocer más sobre arquitecturas orientadas a microservicios.
- Logró aumentar su conocimiento sobre la autenticación biométrica de aplicaciones móviles, así como la generación de un key pair para mayor seguridad.
- Pudo experimentar el uso de los JWT, como token de autenticación.

Capítulo V
Conclusiones y recomendaciones

5.1. Conclusiones

- La creación del microservicio permitió desacoplar las funcionalidades de autenticación del servidor monolito, dando el primer paso para la futura migración de las diferentes funcionalidades de la aplicación bancaria, como se observa en la **Figura 4**, donde el microservicio de autenticación y el monolito son módulos separados.
- La integración del Identity Server permitió añadir nuevas funcionalidades de autenticación como OTP y authenticator a la aplicación móvil, como se muestra en el Anexo 1, **Flujo 1: Autenticación de dos pasos con OTP** y **Flujo 3: Autenticación de dos pasos con Authenticator**, añadiendo más capas de autenticación, así como brindar una experiencia de autenticación consistente al usuario a través de las diversas plataformas que posee la fintech.
- Se logró migrar las funcionalidades de autenticación biométrica y credenciales de usuario y password, como se muestra, como se muestra en el Anexo 1, **Flujo 1: Autenticación con Credenciales** y **Flujo 3: Pantalla de Login con el botón para lanzar la Autenticación Biométrica**.
- La implementación de los cambios en el monolito descritos en el **Sprint 3**, permitieron la sincronización de la sesión, y se logró tener una sesión compartida entre el monolito, el microservicio y el Identity Server.

5.2. Recomendaciones

- Se recomienda que las siguientes funcionalidades implementadas en la aplicación bancaria sean orientada a microservicios debido a la independencia de los módulos, así como el rápido mantenimiento y la fácil escalabilidad horizontal, de acuerdo a las necesidades del negocio.
- Dado que la aplicación es del rubro bancario y que el flujo de autenticación es un proceso críticon del mismo, se recomienda cambiar frecuentemente los algoritmos de encriptación, así como los algoritmos usados para la firma del JWT.

5.3. Bibliografía

Globant. (2021). *Nuestro trabajo*. Obtenido de https://www.globant.com/: https://www.globant.com/es/our-work

Globant. (2021). *Documents.* Obtenido de https://communications.globant.com/: https://communications.globant.com/Comm/Corporate/Pitch/

Punto a Punto. (16 de Octubre de 2020). *Punto a Punto*. Obtenido de https://puntoapunto.com.ar/: https://puntoapunto.com.ar/migoya-globant-nuestra-mision-es-transformar-digitalmente-al-mundo/

JWT. (2021). *JWT*. Obtenido de https://jwt.io: https://jwt.io/introduction

Microsoft. (2021). *Microsoft*. Obtenido de https://docs.microsoft.com: https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/identity-server

Red Hat. (2021). *Red Hat*. Obtenido de https://www.redhat.com: https://www.redhat.com/es/topics/microservices/what-are-microservices

Comparably. (2021). *Globant LLC*. Obtenido de https://www.comparably.com: https://www.comparably.com/companies/367044/mission

Oauth. (2021). Obtenido de https://oauth.net/: https://oauth.net/

Scrum. (2021). Obtenido de https://www.scrum.org/: https://www.scrum.org/resources/what-is-scrum

CloudCheckr. (2021). Obtenido de https://cloudcheckr.com/: https://cloudcheckr.com/cloud-automation/horizontal-vertical-cloud-scaling/

ScienceDirect. (2021). *Science Direct*. Obtenido de https://www.sciencedirect.com: https://www.sciencedirect.com/topics/computer-science/private-key-pair

Geeks for Geeks. (2021). *Geeks for Geeks*. Obtenido de https://www.geeksforgeeks.org: https://www.geeksforgeeks.org/monolithic-vs-microservices-architecture/

5.4. Glosario de términos

- **Fintech:** Se refiere a cualquier empresa que utilice tecnología para mejorar o automatizar los servicios y procesos financieros.
- **OTP:** El código OTP es una contraseña de un solo uso (un acrónimo de: One-Time Password), también conocida como contraseña o contraseña dinámica. Se utiliza como un segundo factor de autenticación además del nombre de usuario y la contraseña de uso común.
- **Framework:** Un marco de programación es un conjunto de soluciones empaquetadas que resuelve problemas de desarrollo comunes.
- **Open Api 3:** Es interfaz estándar, para las apis REST, que permite que tanto los descubrir y comprender las funcionalidades de un del servicio sin acceso al código fuente.
- **Jira:** Es una aplicación de software que se utiliza para el seguimiento de problemas y la gestión de proyectos.
- **Git:** Es un sistema de control de versiones distribuido gratuito y de código abierto diseñado para manejar proyectos, con velocidad y eficiencia.
- **Bitbucket:** Es una herramienta de colaboración y alojamiento de código basada en Git, creada para equipos de software.
- **Jenkins:** Es un servidor de automatización que controla los procesos de entrega de software a lo largo de todo el ciclo de vida, incluida la construcción, documentación, prueba, paquete, etapa, implementación, análisis de código estático y mucho más.
- **Sonar Qube:** SonarQube es una herramienta de aseguramiento de la calidad del código que recopila y analiza el código fuente y proporciona informes sobre la calidad.
- **Artifactory:** Es un administrador de repositorio que funciona como un único punto de acceso que organiza todos sus recursos binarios, incluidas las bibliotecas propietarias, los artefactos remotos y otros recursos de terceros.

- **Red Hat Openshift:** Es la plataforma empresarial líder de Kubernetes que permite una experiencia similar a la de la nube en todos los lugares donde se implementa.

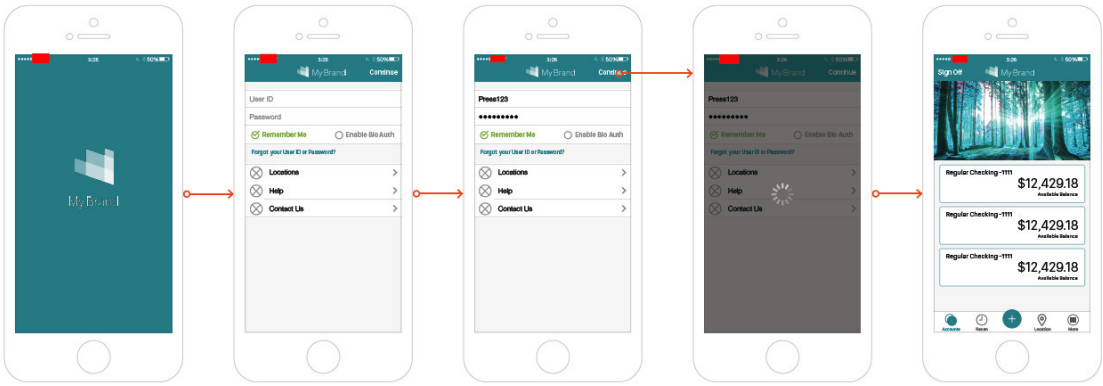
Anexos

Anexo 1: Requerimientos y flujo de pantallas

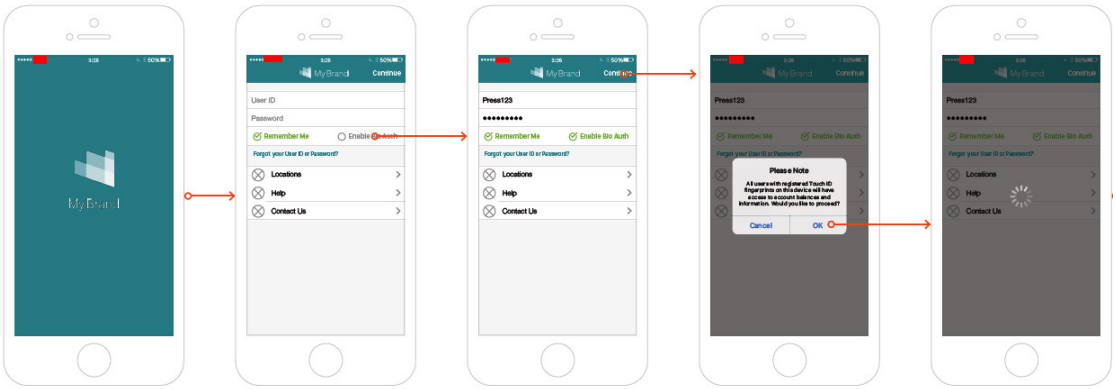
Requerimiento 1: Autenticación contra el Identity Provider

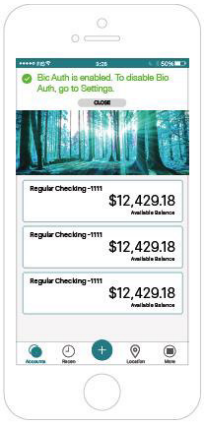
En el Flujo 1, se muestra las pantallas del flujo de autenticación con credenciales, donde son necesarios los valores de username y password, además de la opción de, recordar usuario. En el Flujo 2, se muestra la autenticación con credenciales habilitando la opción de autenticación biométrica, en este caso, usando TouchId. En el Flujo 3 se observa que la opción de autenticación biométrica, esta habilitada y previamente configurada.

Flujo 1: Autenticación con Credenciales

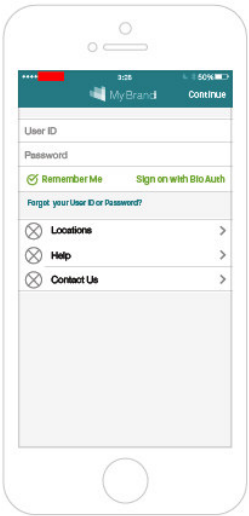


Flujo 2: Autenticación configurando la Autenticación Biométrica





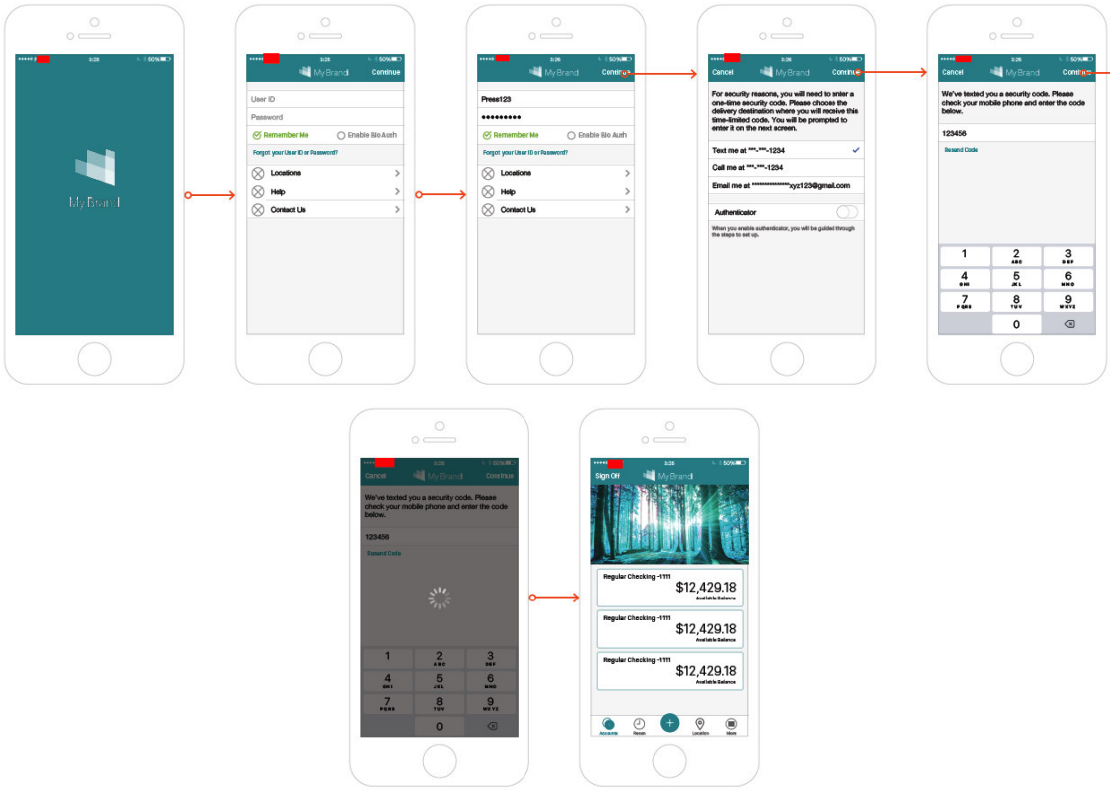
Flujo 3: Pantalla de Login con el botón para lanzar la Autenticación Biométrica



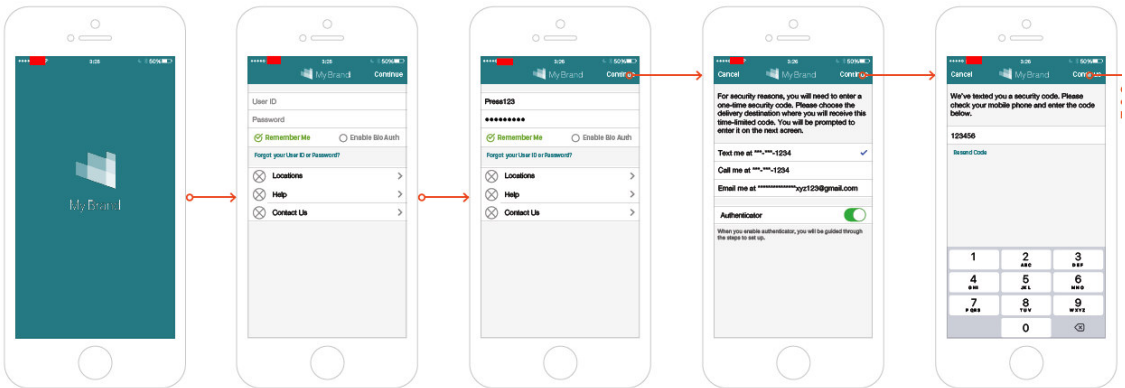
Requerimiento 2: Autenticación de dos pasos

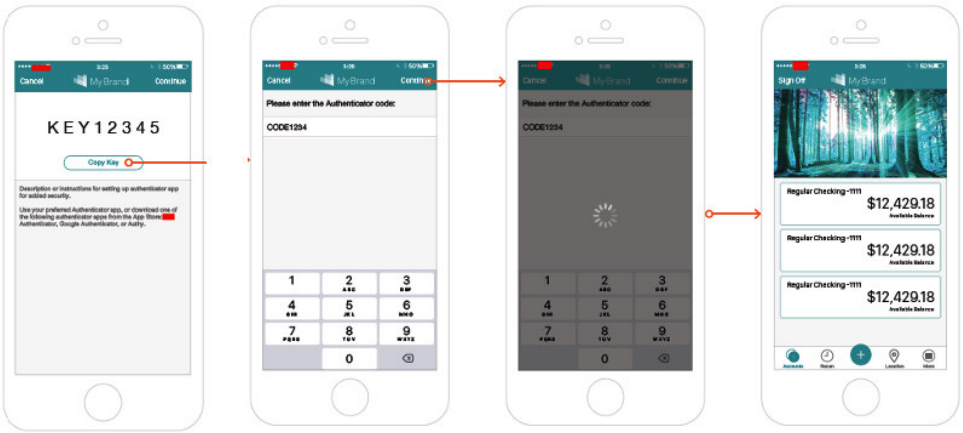
En el Flujo 1, se muestra las pantallas del flujo de autenticación con credenciales, usando username y password, configurando la autenticación con OTP, a través de un dispositivo seleccionado. En el Flujo 2, se muestra el flujo de autenticación con credenciales configurando el authenticator, a través de una autenticación con OTP. En el Flujo 3, se muestra el flujo de autenticación usando el authenticator.

Flujo 1: Autenticación de dos pasos con OTP

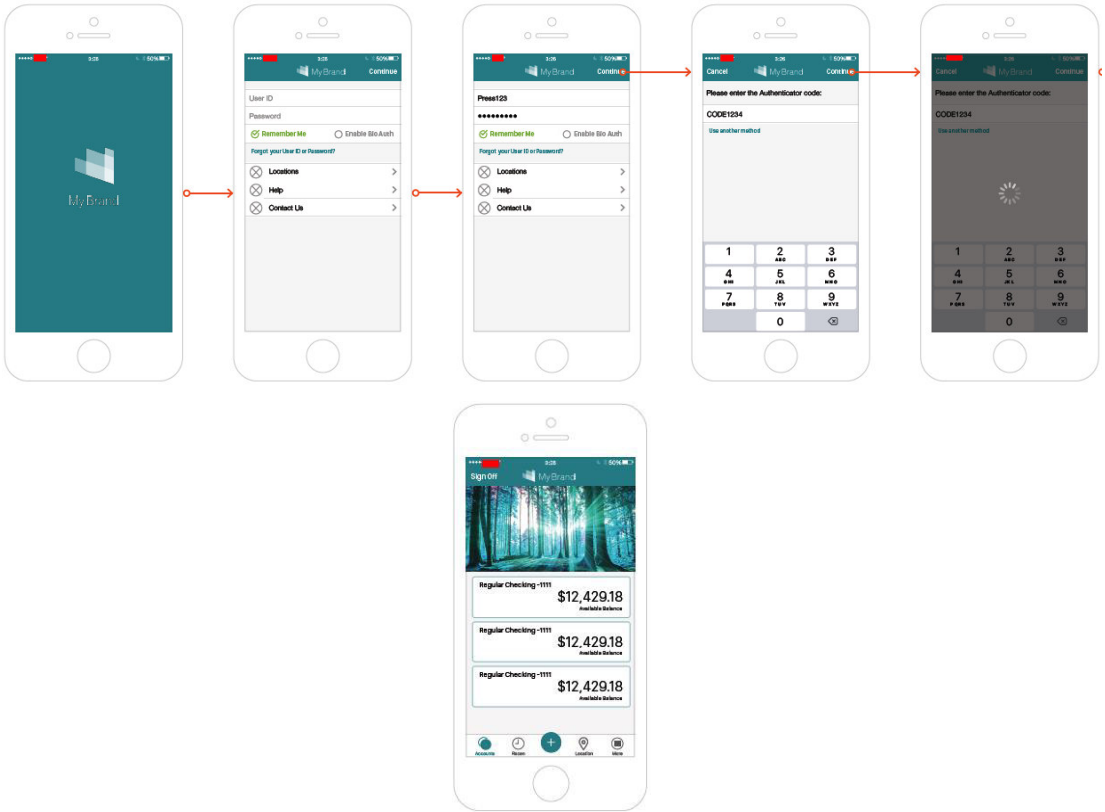


Flujo 2: Autenticación de dos pasos con OTP, configurando el Authenticator



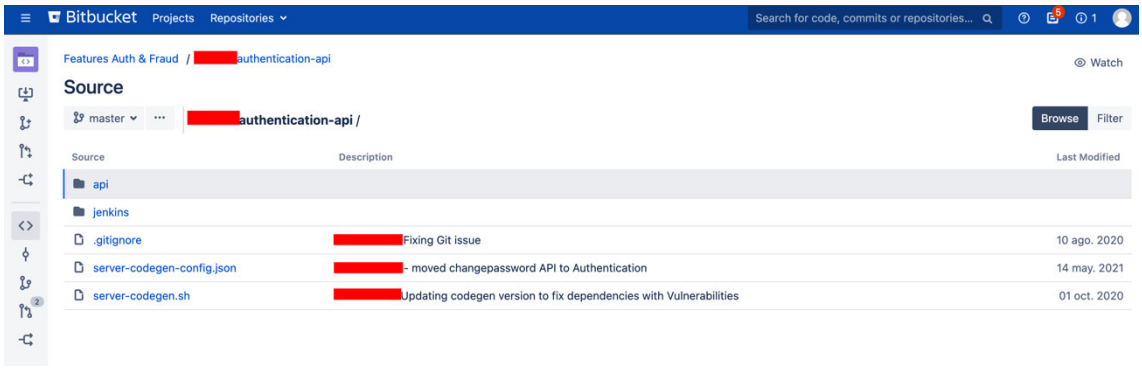


Flujo 3: Autenticación de dos pasos con Authenticator

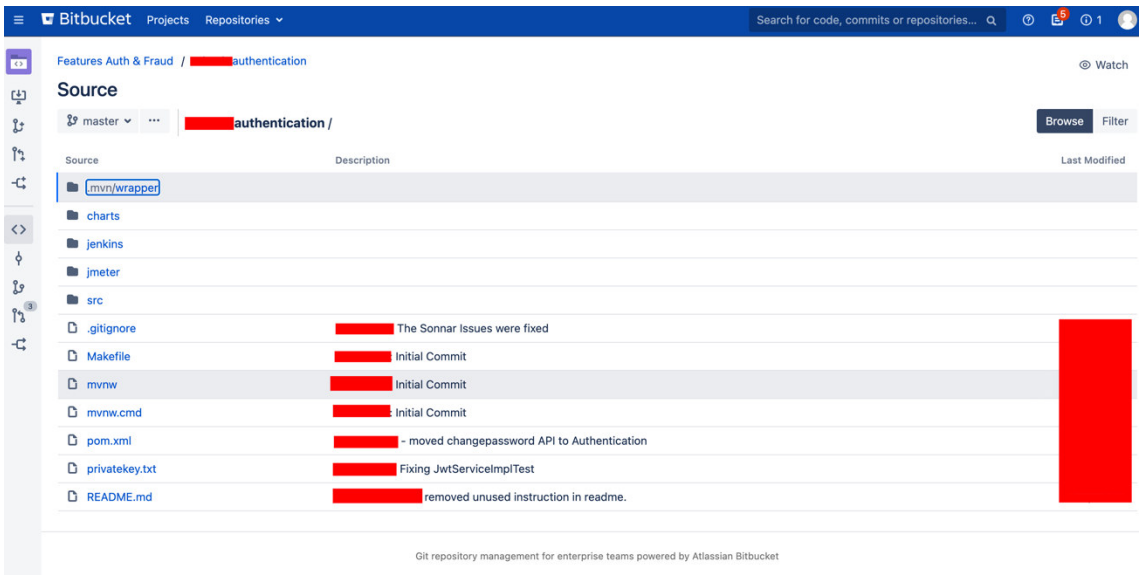


Anexo 2: Repositorio de los proyectos del microservicio

Repositorio en Bitbucket del contrato de apis del microservicio.

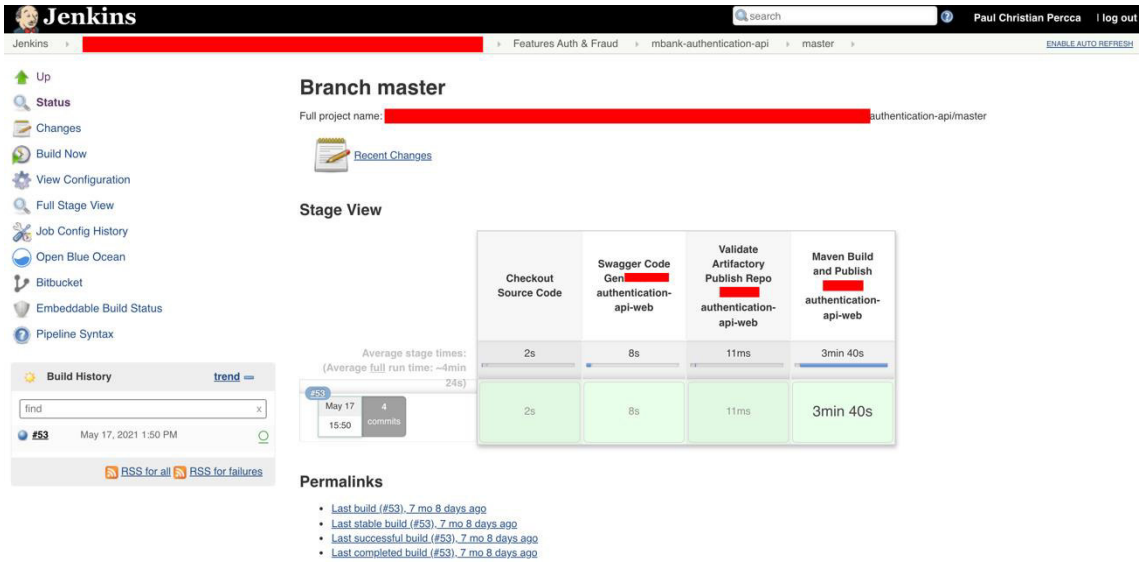


Repositorio en Bitbucket del microservicio, el cual contiene la implementación de las apis



Anexo 3: Pipelines para el despliegue del microservicio

Pipeline de Jenkins para el despliege el contrato de apis del microservicio.



Pipeline de Jenkins para el despliegue del micro servicio de autenticación en Openshift.