ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

CREACIÓN DE UN PROTOTIPO DE SISTEMA DE ONE-TIME PASSWORD (OTP) PARA UN SISTEMA DE AUTENTICACIÓN DE DOS FACTORES.

Implementación de un prototipo de sistema generador de OTP basado en TRNG y su envío a través de Telegram

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

LUIS ERNESTO ALMEIDA ZAMBRANO

luis.almeida01@epn.edu.ec

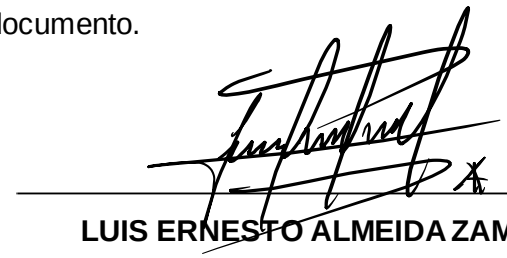
DIRECTOR: SANG GUUN YOO, PhD.

sang.yoo@epn.edu.ec

DMQ, enero de 2023

CERTIFICACIONES

Yo, LUIS ERNESTO ALMEIDA ZAMBRANO declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.



LUIS ERNESTO ALMEIDA ZAMBRANO

Certifico que el presente trabajo de integración curricular fue desarrollado por LUIS ERNESTO ALMEIDA ZAMBRANO, bajo mi supervisión.

SANG GUUN YOO, PhD.
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

LUIS ERNESTO ALMEIDA ZAMBRANO

SANG GUUN YOO

ANTHONY ISRAEL ALMACHI CHASI

BRAYAN ALEXIS FERNANDEZ GONZA

HILTON BLADIMIR PILLAJO ANAGUANO

DALIANA ZAMBRANO PEREDA

DEDICATORIA

Con mucho amor y cariño, dedico este trabajo a mis padres, Ernesto y Margarita. A mi madre, agradezco su amor infinito, apoyo y palabras de aliento incondicionales. A mi padre, le agradezco su guía, consejos y por nunca dejarme rendirme. A ambos, les agradezco por enseñarme que la base para alcanzar mis metas es el trabajo duro y la constancia. Infinitas gracias por ser mi apoyo y motivación para concluir mi carrera académica.

Quiero dedicar unas palabras a mis hermanas, Nora, Diana y Gaby, quienes han sido una parte fundamental en mi vida. Gracias a ellas he aprendido el valor de luchar por mis sueños y perseverar a pesar de las dificultades que se presenten en el camino.

Siempre han estado ahí para mí, escuchándome y brindándome su apoyo incondicional en cada momento que lo he necesitado. Sus consejos sabios y su amor incondicional han sido mi guía en momentos difíciles.

Finalmente, quiero dedicar este trabajo a mis sobrinos Julián, Luana, Omar, Leandro y Sebastián, para transmitirles la importancia de perseguir sus sueños a pesar de las adversidades. Les pido que nunca olviden ser buenas personas, ayudar a los demás y amar a sus padres. El amor y el respeto por la familia son fundamentales en la vida y los ayudarán a alcanzar la felicidad y el éxito en cualquier proyecto que emprendan.

Espero ser un ejemplo para ellos y que este trabajo los inspire a perseguir sus sueños con dedicación, esfuerzo y pasión. Siempre estaré aquí para apoyarlos y guiarlos en su camino.

AGRADECIMIENTO

Un agradecimiento muy especial a Zaida, mi compañera, por ser una fuente constante de palabras de aliento desde el inicio de mi carrera. Gracias por acompañarme en esas noches de desvelo y estudio, y por estar a mi lado tanto en los buenos como en los malos momentos. Tu amor y apoyo significan mucho para mí y siempre estaré agradecido.

A mis amigos Kevin, Xavier, David, Ángel y Andy por demostrarme lo que significa tener una verdadera amistad. Gracias por motivarme a luchar por mis sueños y por estar a mi lado en todo momento.

A mis colegas Edison, David e Ismael, siempre me han motivado para ser una mejor persona y he disfrutado de muchas anécdotas que nos han unido aún más como amigos.

A Brayan, Daliana, Cristhian y Madelyn por todas las largas noches que pasamos juntos haciendo tareas o proyectos. Gracias el apoyo, por todos los consejos y risas que compartimos. Su amistad y ayuda han sido fundamentales en mi éxito y crecimiento personal. Estoy agradecido por tenerlos en mi vida.

A mi tutor, El Profe Sang, quien ha sido una influencia clave en mi pasión por el mundo de la tecnología. Agradezco su motivación para investigar y por permitirme ser parte del Smart Lab. Su guía y apoyo han sido clave en mi formación como profesional. Gracias por enseñarnos a enfrentar nuevos desafíos en el futuro.

A Roberto Andrade por sus valiosos consejos y sugerencias. Su actitud positiva y su capacidad de motivar a los demás es un ejemplo que seguir.

Por último, me gustaría expresar mi agradecimiento a mis amigos del colegio, quienes, a pesar de los años, nos hemos mantenido el contacto. Quiero destacar en especial a David, su esposa Katy y su hijo Benjamín, quienes confiaron en mí desde el inicio y me dieron la oportunidad de demostrar todo lo que soy capaz de hacer.

A todos y cada uno, Gracias.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE CÓDIGOS

ÍNDICE DE TABLAS

RESUMEN

Actualmente, la autenticación de un usuario mediante las contraseñas tradicionales puede resultar inadecuado debido a las vulnerabilidades y los distintos tipos de ataques existentes. Se presenta una solución de autenticación de dos factores (2FA) que se enfoca en la generación, envío y verificación de la One-Time Password (OTP). Se utiliza el algoritmo True Random Number Generator (TRNG) para generar números aleatorios en diferentes lenguajes de programación desplegados en contenedores de Docker seleccionados al azar para aumentar la aleatoriedad. El valor OTP se envía al usuario a través de la aplicación de mensajería Telegram y se envía al servicio de validación para verificar si coincide con el valor OTP generado previamente. Se emplea una técnica de combinación de dos números de tres cifras generados aleatoriamente para reducir el riesgo de ataques de adivinación y proteger la integridad de los valores OTP generados. El índice de repetición de valores OTP se mide haciendo uso de la prueba Chi cuadrado para determinar la aleatoriedad de los números. Con esta prueba se concluye que C++ es el contenedor que genera mayor aleatoriedad. La implementación de esta solución proporciona una forma segura y confiable de obtener valores OTP de seis cifras, lo que es esencial en sistemas que requieran un alto nivel de seguridad y protección de datos.

PALABRAS CLAVE: Autenticación, One-Time Password, OTP, TRNG, seguridad, microservicio.

ABSTRACT

Currently, user authentication using traditional passwords may be inadequate due to vulnerabilities and various types of attacks. A two-factor authentication (2FA) solution is presented that focuses on the generation, transmission, and verification of One-Time Passwords (OTPs). The True Random Number Generator (TRNG) algorithm is used to generate random numbers in different programming languages deployed in randomly selected Docker containers to increase randomness. The OTP value is sent to the user through the Telegram messaging application and sent to the validation service to verify if it matches the previously generated OTP value. A technique of combining two randomly generated three-digit numbers is employed to reduce the risk of guessing attacks and protect the integrity of the generated OTP values. The OTP value repetition rate is measured using the chi-squared test to determine the randomness of the numbers. This test concludes that C++ is the container that generates the highest randomness. The implementation of this solution provides a secure and reliable way to obtain six-digit OTP values, which is essential in systems that require a high level of security and data protection.

KEYWORDS: Authentication, One-Time Password, OTP, TRNG, security, microservices.

1 INTRODUCCIÓN

1.1 Descripción del proyecto

Debido al constante aumento en la implementación y uso de la tecnología, así como a la creciente cantidad de dispositivos conectados a Internet [1], la seguridad informática se ha convertido en un campo de gran importancia. Una de las áreas en las que se enfocan los esfuerzos es la autenticación, la cual se refiere a la verificación de la identidad de un usuario mediante la presentación de credenciales [2], como puede ser el nombre de usuario y la contraseña [3].

Sin embargo, debido a las malas prácticas por parte de los usuarios [2], a la hora de generar contraseñas seguras, se han adoptado otros factores de autenticación, como la autenticación biométrica, el uso de tokens, la presentación y validación de certificados, y la autenticación basada en códigos generados por algoritmos conocidos como One-Time Passwords (OTP) [1].

Según la Association for Computing Machinery (ACM), las One-Time Passwords (OTP) son un método de autenticación de dos factores comúnmente utilizado para verificar la identidad de un usuario antes de permitir el acceso a un sistema o servicio específico [4]. Una OTP se define como una contraseña que solo puede ser utilizada una vez y se genera dinámicamente para cada evento de inicio de sesión o transacción [5]. Actualmente, las OTP son uno de los métodos más populares y ampliamente utilizados para implementar un segundo factor de autenticación (2FA) [6].

Debido a la importancia que se ha generado las OTP en los sistemas de autenticación de los sistemas seguros [1], en este trabajo se ha realizado una investigación de los métodos de generación de OTP y se ha examinado las mejoras potenciales para su implementación [7], para luego realizar el desarrollo de un prototipo de sistema de One-Time Password (OTP) para un sistema de autenticación de dos factores.

El prototipo general propuesto se compone de dos elementos esenciales: un servidor y una aplicación móvil. El servidor cumple con la función de generar, enviar y verificar las OTP. Mientras que la aplicación móvil posibilita al usuario introducir los valores de OTP suministrados por el servidor con la finalidad de efectuar el proceso de autenticación.

La arquitectura del prototipo se muestra en la Figura 1. En la parte izquierda, se muestran dos prototipos de aplicaciones: una aplicación web y una aplicación móvil, que se utilizarán como interfaces de usuario. Ambas interfaces se conectarán al servidor a través de una

API, la cual se basa en una arquitectura de microservicios. El servidor se ha diseñado para contar con cuatro microservicios, cada uno con una función específica:

1. **El servicio de autenticación de usuario**, que permite la verificación de la identidad del usuario a través de un proceso de inicio de sesión.
2. **El servicio de generación de contraseñas de un solo uso (OTP)**, que se encarga de crear códigos aleatorios que se utilizarán para el segundo factor de autenticación.
3. **El servicio de validación de OTP**, que comprueba si el OTP proporcionado por el usuario es correcto o no.
4. **El servicio de envío de OTP**, que se encarga de entregar el código OTP al medio seleccionado por el usuario, ya sea mediante un mensaje de Telegram, correo electrónico o despliegue en pantalla.

Cada microservicio se ejecutará en su propio contenedor, lo que permite una mayor escalabilidad y disponibilidad del sistema en su conjunto.

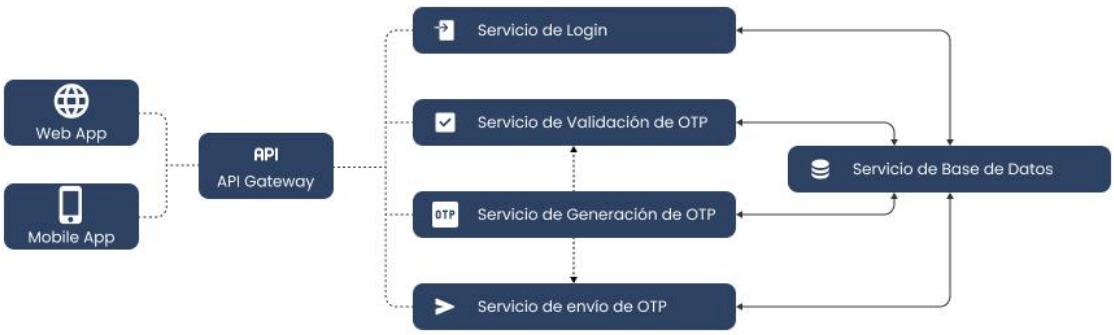


Figura 1. Arquitectura general del prototipo de sistema de One-Time Password (OTP) para un sistema de autenticación de dos factores orientado a microservicios.

El trabajo de integración curricular tiene como objetivo desarrollar tres microservicios necesarios para el proceso de autenticación de usuarios: el servicio de generación, envío y validación de valores OTP.

Una vez desarrollados, estos microservicios se integrarán con un servicio de autenticación de usuario, el cual se ejecutará en un prototipo de aplicación web. El servicio de autenticación permitirá la verificación de la identidad del usuario a través de un proceso de inicio de sesión y la solicitud de un valor OTP generado por los microservicios antes mencionados.

1.2 Descripción del componente desarrollado

El componente desarrollado se enfoca en tres actividades fundamentales para el proceso de autenticación: la generación, el envío y la verificación de la One-Time Password (OTP). Cada una de estas tareas es esencial para garantizar la seguridad y la eficacia de la autenticación de usuarios.

Se ha seleccionado el algoritmo de generación True Random Number Generator (TRNG) para generar números verdaderamente aleatorios. Este algoritmo se basa en la entropía del sistema o de la red, la variación del reloj interno del sistema, el movimiento del ratón, entre otros métodos para proporcionar una aproximación a los números verdaderamente aleatorios [8]. El algoritmo TRNG se implementará en diferentes lenguajes de programación y se desplegará en contenedores de Docker seleccionados de manera aleatoria por el prototipo de autenticación [9].

La implementación del algoritmo TRNG en varios lenguajes de programación y su posterior despliegue en contenedores de Docker seleccionados al azar por el prototipo de autenticación, tiene como objetivo mejorar la seguridad del proceso de autenticación.

La decisión de utilizar múltiples lenguajes de programación se ha tomado para aumentar aún más la aleatoriedad en la generación de valores de contraseña de un solo uso. Cada contenedor generará un valor aleatorio entre 0 y 999, que se concatenará con el valor generado por otro contenedor en el mismo rango. De esta forma, se proporciona una mayor diversidad y se disminuyen las posibilidades de que un atacante pueda predecir los valores generados.

En la Figura 2 se detallan los servicios que se desarrollarán para el proceso de autenticación. Cada vez que un usuario intente autenticarse, la API se comunicará con un servicio de generación de valores OTP. Este valor será generado por dos programas desarrollados en diferentes lenguajes de programación, los cuales se encontrarán desplegados en contenedores de Docker. Cada programa proporcionará un valor numérico de tres cifras, que se concatenarán para producir un valor de OTP. Una vez generado el valor OTP, se enviará al usuario a través de un servicio de envío seleccionado. En este caso, se ha seleccionado Telegram, que ha sido catalogado como una aplicación segura por diversas fuentes, como por ejemplo [10]–[13].

Para proteger el valor OTP generado, se utilizará una función propia de Telegram denominada "Spoiler". Esta función ocultará el valor OTP y permitirá al usuario saber si el valor ha sido leído o no.

Por último, el valor OTP se enviará al servicio de validación, que será consumido por la aplicación móvil. La aplicación móvil solicitará al usuario que ingrese un valor y comprobará si coincide con el valor OTP generado anteriormente.

Figura 2. Arquitectura del componente del prototipo de sistema de One-Time Password (OTP) para un sistema de autenticación de dos factores, usando generadores de OTP basados en TRNG y enviados mediante Telegram.

Con el propósito de simular el proceso de inicio de sesión de un usuario en un portal web, se ha desarrollado un prototipo de sistema de contraseña de un solo uso (OTP) para un sistema de autenticación de dos factores, con los aspectos previamente mencionados. Para validar la OTP, se ha creado una API que permite a la aplicación móvil enviar el valor de la OTP ingresado por el usuario de manera encriptada y segura, con el fin de utilizar un segundo factor de autenticación.

1.3 Objetivo General

Crear un prototipo de Sistema de One-Time Password (OTP) para un sistema de autenticación de dos factores, implementación de un prototipo de sistema generador de OTP basado en TRNG y su envío a través de Telegram.

1.4 Objetivos Específicos

- Comprender el estado actual de las técnicas empleadas y tecnologías relacionadas con las OTP, así como analizar las consideraciones de seguridad necesarias para garantizar la generación y validación mediante la implementación del algoritmo TRNG.
- Diseñar la arquitectura del prototipo de autenticación de dos factores con el generador de OTP basado en el algoritmo TRNG.

- Implementar el generador de OTP utilizando el algoritmo TRNG en diferentes lenguajes de programación y enviar el mismo mediante la aplicación de mensajería Telegram.
- Evaluar la calidad de los valores generados por el algoritmo TRNG mediante pruebas estadísticas.

1.5 Alcance

El presente trabajo de integración curricular tiene como alcance la creación de un prototipo de sistema de One-Time Password (OTP) para un sistema de autenticación de dos factores, y no en un sistema real. Este prototipo simula el inicio de sesión en una aplicación web, donde el usuario debe ingresar la One-Time Password (OTP) recibida a través de su cuenta de Telegram en la aplicación móvil de autenticación.

2 MARCO TEÓRICO

En esta sección se presentan los temas de investigación que serán utilizados en el proceso de desarrollo del componente correspondiente al prototipo del sistema de autenticación de doble factor.

2.1 Autenticación de dos factores (2FA)

La autenticación de dos factores (2FA) es un método de seguridad que requiere la presentación de dos formas distintas de identificación antes de conceder el acceso a un sistema [14]. Este enfoque se considera más seguro que el uso de una única contraseña, ya que, aunque un atacante pudiera obtener la contraseña, necesitaría el segundo factor para poder ingresar a la cuenta [14]. Existen diferentes tipos de factores de autenticación que pueden combinarse para implementar la autenticación de dos factores. Estos incluyen la contraseña, el token físico, el token móvil y la biometría [15]. La contraseña es la forma de autenticación más utilizada, pero por sí sola no es suficiente para garantizar la seguridad. Los tokens físicos y móviles generan códigos de un solo uso que deben combinarse con la contraseña para conceder el acceso. Por último, la autenticación mediante biometría utiliza características físicas o de comportamiento únicas del usuario, como las huellas dactilares, el reconocimiento facial o el reconocimiento de voz [15], [16].

Además de aumentar la seguridad, la autenticación de dos factores también puede ayudar a reducir el riesgo de phishing y otros ataques de suplantación de identidad. Sin embargo, también puede presentar desafíos para los usuarios, especialmente si el segundo factor de autenticación es difícil de usar o está ausente en ciertos dispositivos o aplicaciones [17], [18].

Uno de los tipos de autenticación de dos factores mencionados anteriormente es el token físico, que es un dispositivo que genera un código de un solo uso que el usuario debe ingresar junto con su contraseña [19]. Este tipo de autenticación utiliza lo que se conoce como One-Time Password (OTP), que es un código de uso único que es válido solo por un corto período de tiempo, generalmente unos pocos minutos. Los OTP se generan a partir de un algoritmo que utiliza una clave compartida entre el dispositivo y el servidor de autenticación [2].

Los tokens físicos pueden tomar varias formas, como llaveros o tarjetas inteligentes, y se utilizan en una amplia variedad de aplicaciones, desde la banca en línea hasta la autenticación de VPN empresariales [20]. Los OTP generados por estos dispositivos se consideran extremadamente seguros debido a la naturaleza temporal del código y la falta de conexión en línea con el servidor de autenticación [21].

Sin embargo, los tokens físicos pueden ser costosos y pueden requerir mantenimiento o reemplazo periódico, lo que puede ser problemático para las empresas que buscan implementar la autenticación de dos factores a gran escala [19]. Además, los usuarios deben llevar el dispositivo consigo en todo momento para poder autenticarse, lo que puede ser una molestia [16], [18].

En respuesta a estos desafíos, los tokens móviles se han vuelto cada vez más populares en los últimos años [2], [3]. Los tokens móviles son similares a los tokens físicos en que generan OTP, pero en lugar de un dispositivo físico, el usuario utiliza una aplicación móvil en su teléfono inteligente para generar los códigos. Los tokens móviles son más convenientes para los usuarios, ya que generalmente llevan sus teléfonos con ellos en todo momento [22].

La autenticación de dos factores que utiliza OTP, ya sea a través de un token físico o móvil, es una de las formas más seguras de autenticación disponible actualmente [23]. Las OTP son extremadamente difíciles de adivina, lo que significa que incluso si un atacante obtiene la contraseña de un usuario, todavía necesitaría la OTP para acceder a la cuenta.

2.2 2D-2FA

La autenticación de dos factores (2FA) se ha vuelto cada vez más importante en la protección de los sistemas y las cuentas en línea [5], [16]. Sin embargo, los métodos existentes de autenticación de dos factores tienen limitaciones en términos de seguridad y usabilidad. Por lo tanto, en [16] se propone un mecanismo de autenticación de dos factores llamado 2D-2FA que aborda estos problemas.

El mecanismo de 2D-2FA tiene tres características distintivas que lo hacen diferente de otros métodos de autenticación de dos factores. Después de que el usuario ingresa un nombre de usuario y contraseña en una terminal de inicio de sesión, se genera un identificador único. Este identificador se muestra en la pantalla y se ingresa en el dispositivo 2FA registrado del usuario. Esta característica garantiza la participación adecuada del usuario en el proceso de autenticación [1].

El uso de códigos OTP de un solo uso es una característica común en los métodos de autenticación de dos factore [23]s. En el mecanismo 2D-2FA propuesto, el PIN es un código de acceso temporal que se genera en el dispositivo del usuario y se transfiere automáticamente al servidor para completar el proceso de autenticación. Esta característica garantiza una mayor seguridad en la autenticación de dos factores, lo que ayuda a proteger los sistemas y las cuentas en línea de los usuarios [1].

El artículo presenta una implementación de prueba de concepto y evalúa el rendimiento, la precisión y la usabilidad del sistema. Los resultados muestran que el sistema tiene una tasa de error más baja y una mayor eficiencia en comparación con el método PIN-2FA comúnmente utilizado. Además, la alta usabilidad del sistema se muestra mediante un puntaje SUS y una alta percepción de eficiencia, seguridad, precisión y capacidad de adopción [1].

2.3 One-Time Password (OTP)

Las One-Time Password (OTP) son códigos de uso único que se utilizan en la autenticación de dos factores para proporcionar una capa adicional de seguridad [24]. Existen diferentes métodos de generación de OTP, pero todos comparten la característica de que solo son válidos por un corto período de tiempo, lo que los hace extremadamente difíciles de adivinar o piratear [25]–[27].

Uno de los métodos de generación de OTP más comunes es a través de un token físico, que utiliza un algoritmo para generar un código único cada vez que se solicita [21]. Los tokens físicos suelen tomar la forma de llaveros o tarjetas inteligentes, y están diseñados para ser portátiles y duraderos. Cuando se necesita autenticación, el usuario ingresa su contraseña junto con el código generado por el token, lo que proporciona una capa adicional de seguridad [19].

Otro método de generación de OTP es a través de una aplicación móvil, que se ejecuta en un smartphone o tableta [2]. Estas aplicaciones utilizan un algoritmo similar al de los tokens físicos para generar un código de uso único que se ingresa junto con la contraseña del usuario. Las aplicaciones de OTP móvil son muy populares debido a su conveniencia y

facilidad de uso, ya que la mayoría de las personas llevan sus teléfonos con ellos en todo momento [3].

Es importante destacar que los OTP deben ser generados de manera segura, utilizando técnicas criptográficas para evitar la repetición de códigos y la predicción de futuros códigos [23], [25], [26]. Una forma de lograr esto es mediante la utilización de Fuentes de Entropía Verdaderamente Aleatorias (TRNG, por sus siglas en inglés) [8].

Las TRNG son dispositivos que generan números aleatorios a partir de fuentes físicas de entropía, como el ruido térmico o la radioactividad, que son inherentemente impredecibles. Estos números aleatorios se utilizan como semilla para los algoritmos de generación de OTP, lo que garantiza que los códigos generados sean únicos e imposibles de predecir [28].

2.4 Algoritmo True Random Number Generator (TRNG)

Un True Random Number Generator (TRNG) es un dispositivo o programa que genera números aleatorios verdaderos, es decir, números que no pueden ser predecibles ni repetidos [29]. A diferencia de los pseudo-generadores de números aleatorios, que utilizan un algoritmo para producir una secuencia aparentemente aleatoria de números, los TRNG se basan en fuentes físicas de aleatoriedad, como el ruido atmosférico, el ruido térmico, la radioactividad o la turbulencia del aire [30].

En términos de software, los TRNG se implementan mediante algoritmos que utilizan fuentes de entropía, como la actividad del usuario, el movimiento del ratón o el tiempo entre pulsaciones de teclas. Estos algoritmos producen una secuencia de números aleatorios que son realmente impredecibles y no pueden ser generados de nuevo [30].

Los TRNG son importantes en aplicaciones criptográficas, donde se necesitan números aleatorios verdaderos para generar claves criptográficas, para la autenticación y la generación de One-Time Passwords (OTP). También son utilizados en aplicaciones de juegos, simulaciones y pruebas de software, donde se requiere una fuente de aleatoriedad confiable [31].

Una forma de mejorar la seguridad de las OTP es utilizar TRNG para generar los códigos en lugar de utilizar TOTP o HOTP. Los algoritmos de generación de OTP basados en TRNG utilizan un flujo continuo de números aleatorios para generar códigos de uso único que son verdaderamente impredecibles [31].

2.5 Microservicios

Los microservicios representan una estructura de software que busca la construcción de sistemas a partir de una serie de pequeñas unidades de servicio altamente cohesionadas e independientes que colaboran para brindar una funcionalidad completa a una aplicación [32]. Los microservicios permiten una mayor flexibilidad, escalabilidad y agilidad en el desarrollo y despliegue de aplicaciones, en comparación con las arquitecturas monolíticas tradicionales. Cada microservicio es independiente, lo que permite una mayor eficiencia en la gestión de los recursos y una mayor escalabilidad horizontal [32].

Docker, por otro lado, es una plataforma de contenedores que se utiliza para construir, implementar y ejecutar aplicaciones en diferentes entornos. Docker proporciona un entorno aislado para ejecutar aplicaciones y servicios, lo que hace que sea más fácil para los desarrolladores implementar aplicaciones en diferentes entornos, desde el desarrollo hasta la producción [33], [34]. Al usar contenedores, Docker también permite a los desarrolladores empaquetar aplicaciones y servicios, junto con todas sus dependencias, en un solo paquete.

Los microservicios y Docker son una combinación ideal, ya que Docker permite a los desarrolladores empaquetar cada microservicio en un contenedor separado, lo que hace que sea más fácil de implementar, escalar y administrar. Cada microservicio puede tener sus propias dependencias y configuraciones, lo que permite una mayor flexibilidad en el desarrollo y despliegue de aplicaciones [32].

Los algoritmos generadores de TRNG son un ejemplo de una funcionalidad que se puede implementar como un microservicio en un entorno de Docker. Al implementar un algoritmo generador de TRNG como un microservicio en un entorno de Docker, los desarrolladores pueden aislar y proteger el algoritmo de posibles vulnerabilidades y ataques, lo que aumenta la seguridad de los sistemas de autenticación que utilizan OTP generados por TRNG [32], [33].

3 METODOLOGÍA

En la presente sección se describen las fases que se llevaron a cabo durante el desarrollo del componente del prototipo de Sistema de One-Time Password (OTP) para un sistema de autenticación de dos factores. La metodología inició con la investigación y planificación del proyecto. seguida de la ejecución y desarrollo del componente. Para cumplir con los objetivos planteados, se aplicó la metodología de desarrollo iterativo e incremental. Esta técnica permitió alcanzar una organización eficiente y una ejecución ágil de las tareas, lo

que se tradujo en entregas de calidad dentro del plazo establecido. La implementación de este marco de trabajo ofreció diversos beneficios, entre los que destacan los siguientes:

1. Se decidió implementar el componente a través de cuatro iteraciones, cada una con una duración de tres semanas, con el objetivo de lograr un incremento continuo en el valor del componente.
2. Se estableció una práctica periódica de reuniones para informar sobre las actividades completadas, las tareas pendientes y, en caso de ser necesario, para abordar y resolver los obstáculos o bloqueos con la participación de todos los miembros del equipo.

En la Figura 3, se muestran las etapas del proyecto y las metodologías que se utilizaron durante el desarrollo del componente.

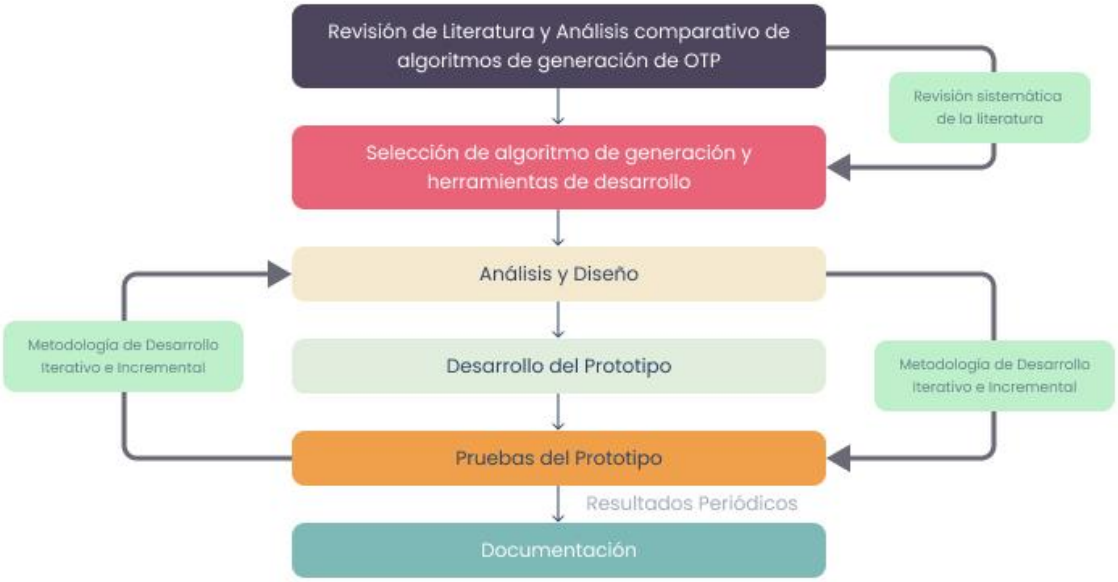


Figura 3. Etapas y metodologías empleadas durante en desarrollo del proyecto.

La metodología empleada para las fases uno y dos consistió en una revisión sistemática de la literatura, técnica ampliamente utilizada en investigaciones en distintos campos debido a su rigor y objetividad [35]. En la primera fase, se formuló una pregunta de investigación que permitió llevar a cabo una búsqueda de literatura relevante en diversas bases de datos electrónicas, entre las que se incluyen ACM Digital Library, IEEE Xplore, Springer, Scopus Digital Library y ScienceDirect.

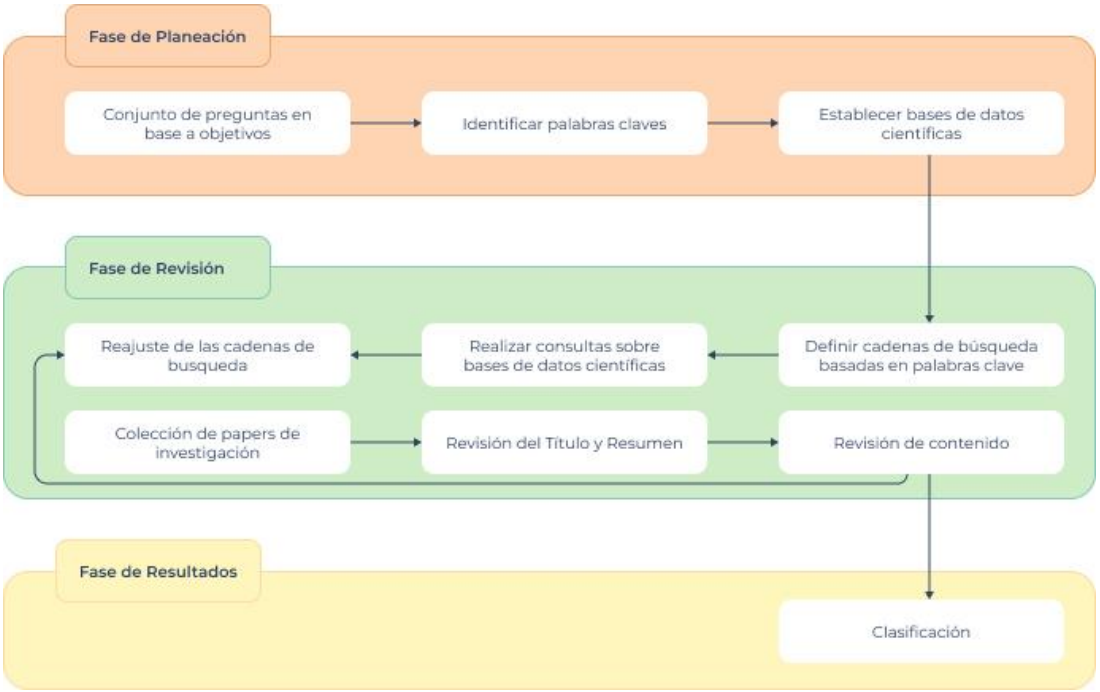


Figura 4. Fases de la metodología Investigación - Acción

En la fase dos, se llevó a cabo una evaluación de herramientas y posibles arquitecturas para la implementación del prototipo. Se seleccionaron las herramientas y arquitecturas más adecuadas para satisfacer los requisitos y objetivos del proyecto. Para la selección de una arquitectura en particular, se revisaron los RFC correspondientes a las One-Time Password (OTP) [25]–[27]., así como los artículos de investigación más relevantes obtenidos en la primera fase [1]–[3], [13].

Conforme se avanzaba en el proceso de revisión de la literatura, se utilizó la plataforma Notion, para poder tener un mejor manejo de las notas que se obtenían de cada uno de los artículos. Notion es una herramienta de productividad y organización que brinda la capacidad de crear y compartir notas, documentos, bases de datos y diferente tipo de contenido, así como también ofrece la opción de colaboración en un mismo proyecto [36].

Se ha seleccionado Flask como base para la creación del prototipo de sistema de One-Time Password (OTP) para un sistema de autenticación de dos factores. Flask es un framework minimalista escrito en Python, que permite la creación y despliegue rápido de aplicaciones web y APIs con un reducido número de líneas de código. Flask hace uso de la especificación WSGI, un estándar que permite la comunicación a través del protocolo HTTP, y del motor de renderizado de plantillas Jinja 2. Flask está distribuido bajo la licencia BSD, una licencia permisiva que permite su uso en software no libre. Con el uso de Flask, es posible desarrollar una arquitectura de microservicios, la cual se caracteriza por su

capacidad de desagregar aplicaciones en servicios más pequeños e independientes, que pueden ser implementados y escalados de manera independiente [37], [38].

Docker es un software de virtualización que automatiza la implementación de aplicaciones en contenedores de software [39]. Un contenedor de software es una unidad de ejecución que encapsula una aplicación y sus dependencias, proporcionando un ambiente aislado del sistema operativo host. Este tipo de tecnología facilita la portabilidad y la escalabilidad de las aplicaciones, ya que se pueden desplegar fácilmente en diferentes entornos sin necesidad de modificar el código [40]. En la implementación de microservicios, es común que cada servicio tenga su propia base de datos encapsulada, la cual es utilizada para almacenar la información necesaria para su correcto funcionamiento. Para cumplir con esta necesidad, es importante contar con un gestor de base de datos confiable y escalable [32].

PostgreSQL es un sistema de gestión de bases de datos relacional que se enfoca en la confiabilidad, escalabilidad y capacidad de manejar grandes cantidades de datos de manera eficiente [32]. Se trata de un software de código abierto y gratuito, desarrollado con el objetivo de cumplir con los estándares de SQL. PostgreSQL destaca por ofrecer una amplia variedad de características avanzadas, entre las que se incluyen la capacidad de soportar múltiples usuarios concurrentes, transacciones ACID, replicación, indexación avanzada, así como funciones y procedimientos almacenados. Todo esto hace que PostgreSQL sea una excelente opción para empresas y organizaciones que necesitan un sistema de gestión de bases de datos robusto, confiable y escalable [41].

A partir de la fase 3, Análisis y Diseño, se optó por integrar la metodología Iterativa Incremental [42]. Cada iteración incluye las siguientes fases en el proceso de desarrollo:

1. **Fase de Planificación**: Durante esta fase, se definen las actividades, objetivos y tareas a cumplir durante la iteración. También se establecen los requisitos funcionales necesarios para el desarrollo del prototipo.
2. **Fase de Diseño**: se utilizaron mockups y convenciones para proponer una solución a los requerimientos obtenidos. El objetivo fue diseñar una interfaz gráfica y una API eficiente que cumpla con los requisitos del prototipo.
3. **Fase de Desarrollo**: Se implementó la solución que se obtuvo en la etapa anterior.
4. **Fase de Pruebas**: En esta fase se ejecutan pruebas a la solución obtenida. En caso de detectar errores, se corrigen en la siguiente iteración para asegurar la calidad del prototipo.

La representación gráfica de cada una de las iteraciones junto con sus respectivas fases se puede visualizar en la Figura 5.

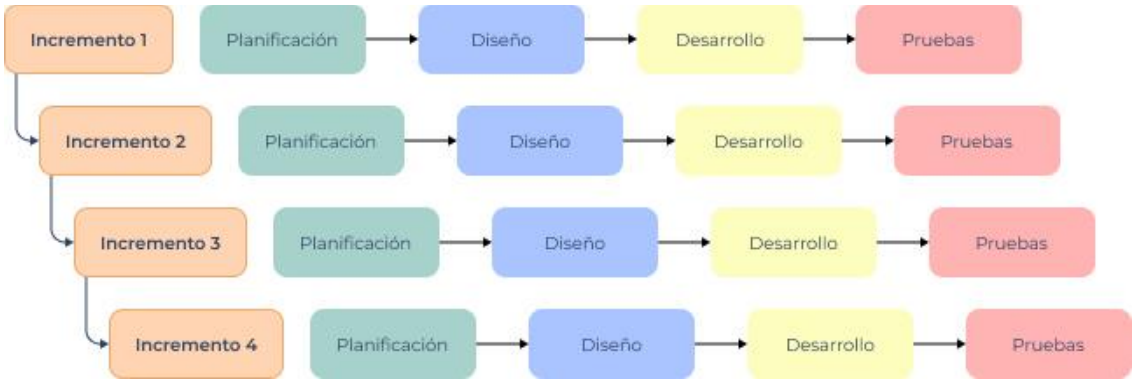


Figura 5. Incrementos y fases de la metodología de desarrollo iterativo incremental.

4 INCREMENTOS DEL PROTOTIPO

4.1 Medidas de seguridad y algoritmos de TRNG.

4.1.1 Análisis de seguridad para los algoritmos de generación de TRNG.

En el primer incremento del proyecto se han considerado las medidas de seguridad y requisitos necesarios para la generación de One-Time Passwords (OTP) definidos en [1]–[3], las cuales se encuentran detalladas en el Anexo I. Para llevar a cabo este proceso, se han implementado diferentes contenedores que alojan generadores de OTP en diversos lenguajes de programación. Estos contenedores permitirán la generación de códigos OTP de 3 cifras, los cuales serán utilizados posteriormente para la autenticación de los usuarios en el sistema.

4.1.2 Diseño de la arquitectura basada en contenedores para los algoritmos de TRNG.

Teniendo en cuenta las consideraciones de seguridad detalladas en el Anexo I, se ha decidido implementar algoritmos que empleen generadores de números verdaderamente aleatorios (TRNG). Estos algoritmos se implementaron en diferentes lenguajes de programación haciendo uso de contenedores. Esta medida se basa en la necesidad de garantizar la seguridad de las One-Time Passwords (OTP) generadas.

Los valores generados mediante el uso de TRNG son utilizados para asegurar la singularidad y no predecibilidad de cada OTP generada. Además, la implementación de contenedores en la aplicación generadora ayuda a aislar esta del sistema base, lo que contribuye a la protección de los valores generados contra posibles ataques.

4.1.3 Desarrollo de los algoritmos en diferentes lenguajes de programación.

Una vez establecida la arquitectura de los contenedores, se inició la fase de desarrollo de los algoritmos utilizando diferentes lenguajes de programación, tomando en cuenta las consideraciones de seguridad mencionadas en [28], [29]. El índice TIOBE se utilizó como factor de selección de los lenguajes. Este índice se basa en la búsqueda mensual realizada en los diferentes motores de búsqueda (Google, Bing, Yahoo, etc.) y se actualiza de forma mensual [43].

Los lenguajes de programación seleccionados para el desarrollo de los algoritmos son los siguientes:

- **Golang**

  Golang o Go, es un lenguaje de programación abierto desarrollado por Google. Go es eficiente en el uso de la memoria y con una sintaxis concisa. Entre sus ventajas se destacan su compilación rápida, su soporte para la concurrencia y su facilidad para trabajar con grandes conjuntos de datos [44].

```go
package main

import (
    "crypto/rand"
    "fmt"
)

func main() {
    // creamos un slice para guardar los bytes generados
    bytes := make([]byte, 2)

    // leemos bytes aleatorios en el slice
    _, err := rand.Read(bytes)
    if err != nil {
        panic(err)
    }

    // convertimos los bytes a un número de 3 cifras
    otp := fmt.Sprintf("%03d",
(int(bytes[0])*256+int(bytes[1]))%1000)
}
```

Código 1. Implementación de TRNG en Golang.

- **Python**

  Python es un lenguaje de programación de alto nivel, interpretado, con una sintaxis clara y sencilla. Entre sus ventajas se destacan su amplia biblioteca estándar, su portabilidad y su facilidad de aprendizaje. Python permite crear diferente tipo de proyectos, como Aplicaciones Web, Machine Learning, Análisis de Datos, etc. [37], [45].

```python
import os

# leemos 2 bytes aleatorios del generador de números aleatorios del sistema
rand_bytes = os.urandom(2)

# convertimos los bytes a un número de 3 cifras
otp = '{:03d}'.format(sum(rand_bytes) % 1000)

print('Tu OTP es:', otp)
```

Código 2. Implementación de TRNG en Python.

- **Java**

Java es un lenguaje de programación de alto nivel y orientado a objetos. Entre sus ventajas se destacan su portabilidad, seguridad y escalabilidad, lo que lo convierte en una buena opción para aplicaciones empresariales y sistemas embebidos [46].

```java
import java.security.SecureRandom;

public class OTPGenerator {
    public static void main(String[] args) {
        // creamos una instancia de SecureRandom
        SecureRandom secureRandom = new SecureRandom();

        // generamos un número aleatorio de 3 cifras
        int otp = secureRandom.nextInt(1000);

        // aseguramos que el número tenga 3 cifras
        String otpString = String.format("%03d", otp);
        System.out.println("Tu OTP es: " + otpString);
    }
}use rand::Rng;

fn main() {
    // creamos una instancia de Rng
    let mut rng = rand::thread_rng();

    // generamos un número aleatorio de 3 cifras
    let otp = rng.gen_range(0, 1000);

    // aseguramos que el número tenga 3 cifras
    let otp_string = format!("{:03}", otp);
    println!("Tu OTP es: {}", otp_string);
}
```

Código 3. Implementación de TRNG en Java.

- **Rust**

Rust es un lenguaje de programación de sistemas de bajo nivel, con una sintaxis similar a la de C++. Entre sus ventajas se destacan su seguridad, rendimiento y concurrencia, lo que lo convierte en una buena opción para el desarrollo de sistemas operativos, controladores de dispositivos y otros proyectos de bajo nivel [47].

```rust
use rand::Rng;

fn main() {
    // creamos una instancia de Rng
    let mut rng = rand::thread_rng();

    // generamos un número aleatorio de 3 cifras
    let otp = rng.gen_range(0, 1000);

    // aseguramos que el número tenga 3 cifras
    let otp_string = format!("{:03}", otp);
    println!("Tu OTP es: {}", otp_string);
}
```

Código 4. Implementación de TRNG en Rust.

- **C++**

C++ es un lenguaje de programación de alto nivel que se basa en C. Se caracteriza por su eficiencia, portabilidad y flexibilidad, lo que lo convierte en una excelente opción para el desarrollo de sistemas operativos, compiladores y otros proyectos que requieren acceso directo al hardware. [48].

```cpp
#include <random>
#include <iostream>

int main() {
  // Crear un generador de números aleatorios verdaderamente aleatorios (TRNG)
  std::random_device rd;

  // Crear un motor de números aleatorios que utilice el TRNG como fuente de entropía
  std::mt19937 gen(rd());

  // Crear una distribución uniforme que genere valores en el rango [0, 999]
  std::uniform_int_distribution<int> dis(0, 999);

  // Generar un número aleatorio utilizando el motor y la distribución
  int random_num = dis(gen);

  // Asegurarse de que el número generado tenga tres dígitos
  std::string random_str = std::to_string(random_num);
  while (random_str.length() < 3) {
    random_str = "0" + random_str;
  }

  // Imprimir el número generado
  std::cout << random_str << std::endl;

  return 0;
}
```

Código 5. Implementación de TRNG en C.

4.1.4 Prueba de cada uno de los algoritmos generadores de TRNG

Durante el presente incremento, se llevaron a cabo pruebas con el fin de determinar el índice de repetición de valores OTP generados cada 30 segundos durante un periodo de 24 horas. Es importante mencionar que los valores OTP fueron generados a través de códigos que producen valores de tres cifras en cada programa.

El objetivo principal de estas pruebas es permitir la concatenación de valores de dos contenedores en el siguiente incremento, para obtener valores OTP de seis cifras.

4.2 Configuración de arquitectura y servicios de API para comunicar los valores generados

4.2.1 Análisis de la configuración de la arquitectura de comunicación basada en una API

En el segundo incremento del proyecto, se ha llevado a cabo la configuración de la arquitectura necesaria para establecer la comunicación entre los diferentes contenedores del sistema. Con este objetivo, se ha creado un contenedor que alojará un servicio de API implementado en Flask, el cual permitirá la comunicación y transmisión de información entre los distintos módulos del prototipo.

4.2.2 Diseño de la configuración de la arquitectura de comunicación basada en una API

El diseño de la arquitectura se presenta en la Figura 6. En dicha figura, se observa que la API enviará una solicitud a dos contenedores para la generación de números aleatorios. Es importante destacar que estos números aleatorios presentan una longitud de tres cifras. Por consiguiente, para producir un OTP de seis cifras, se deberán ejecutar ambos contenedores.

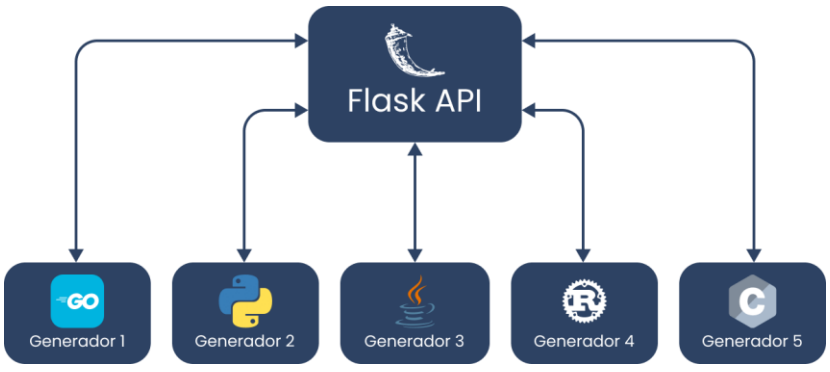


Figura 6. Arquitectura de la API para obtener los valores OTP generados.

Para garantizar la confidencialidad de la información que será transmitida a través de la API, se utilizará el algoritmo de cifrado RSA. Con este propósito, se generará un par de claves criptográficas, consistente en una clave pública y una clave privada, en cada contenedor. Estas claves se utilizarán para encriptar los valores generados antes de ser enviados a la API. De esta forma, se asegura que los datos sensibles no puedan ser accedidos por terceros.

4.2.3 Desarrollo de la configuración de la arquitectura de comunicación basada en una API

El proceso de desarrollo de este incremento comienza con la creación de un par de llaves en cada contenedor, las cuales serán empleadas para cifrar el contenido generado y enviado a través de la API (en este caso, los valores OTP). El Código 6 detalla los pasos para la generación de dichas claves en cada contenedor, mientras que la implementación completa se presenta en el Anexo II.

```python
import os
from cryptography.hazmat.primitives.asymmetric import rsa,
padding
from cryptography.hazmat.primitives import serialization

# Generar un par de llaves RSA
private_key = rsa.generate_private_key(
    public_exponent=65537,
    key_size=2048,
)
public_key = private_key.public_key()

# Serializar las llaves en formato PEM
private_key_pem = private_key.private_bytes(
    encoding=serialization.Encoding.PEM,
    format=serialization.PrivateFormat.PKCS8,
    encryption_algorithm=serialization.NoEncryption(),
)
public_key_pem = public_key.public_bytes(
    encoding=serialization.Encoding.PEM,
    format=serialization.PublicFormat.SubjectPublicKeyInfo,
)
```

Código 6. Generación de par de llaves RSA en Python.

Una vez se haya creado el par de llaves, estas serán almacenadas como variables de entorno. Para realizar esta operación, se ejecutará el Código 7.

```python
# Guardar las llaves como variables de entorno
os.environ["RSA_PRIVATE_KEY"] = private_key_pem.decode('utf-
8')
os.environ["RSA_PUBLIC_KEY"] = public_key_pem.decode('utf-8')
```

Código 7. Almacenamiento de par de llaves como variables de entorno.

Finalmente, mediante un Endpoint, Código 8, se comunicará la llave publica en el caso de que se necesite realizar una nueva generación de este par de llaves.

```python
@app.route('/get-public-key')
def get_public_key():
    return os.environ.get('RSA_PUBLIC_KEY')
```

Código 8. Endpoint para obtener el par de llaves.

4.2.4 Pruebas de la configuración de la arquitectura de comunicación basada en una API

Para el presente incremento, se llevarán a cabo pruebas destinadas a verificar la correcta transmisión de valores encriptados de OTP generados para cada contenedor. Estas pruebas consistirán en la obtención de dos valores encriptados, los cuales posteriormente serán descifrados a través del uso de la llave privada del contenedor de la API.

4.3 Implementación de servicio de validación de One-Time Password (OTP)

4.3.1 Análisis de la implementación de un servicio de validación de One-Time Password (OTP)

Para este tercer incremento del proyecto, se ha llevado a cabo la implementación de un servicio de validación de valores de One-Time Password (OTP), el cual se ha desarrollado mediante la implementación de una API. que será consumida por la aplicación móvil del cliente y por la API del prototipo del sistema.

La API implementada para la validación de valores de One-Time Password (OTP), permite a la aplicación móvil del cliente y al prototipo del sistema validar la autenticidad de los OTP generados y recibidos, mediante la comparación de ambos valores para comprobar su coincidencia.

4.3.2 Diseño de la implementación de un servicio de validación de One-Time Password (OTP)

En la Figura 7 se muestra el diseño de la arquitectura para este incremento. En ella se puede observar que, al recibir el valor de OTP generado, la API del incremento anterior se encarga de enviar dicho valor al servicio de validación. Este último, espera un tiempo específico antes de abortar el proceso de inicio de sesión, tal y como se especifica en [2], [3], [28].

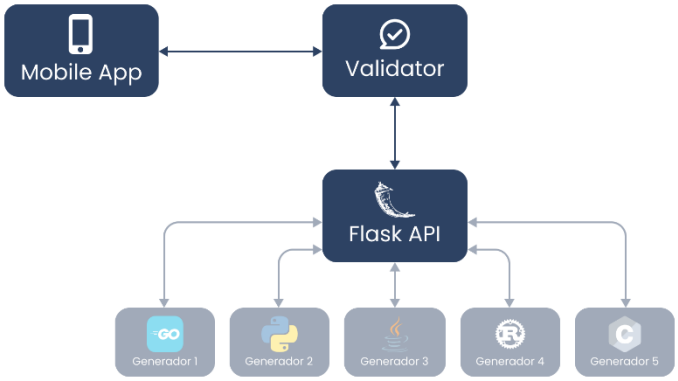


Figura 7. Arquitectura de la API para el servicio de validación.

La comunicación de estas llaves entre los contenedores se emplea siguiendo la lógica implementada en la API anterior. Mientras que la transferencia de la llave pública del servidor y la llave pública del teléfono celular, se hace uso de una configuración previa, en el proceso de registro. Este proceso de implementará en el siguiente incremento. Mientras, se hará uso de un servicio que consuma la API enviando valores de OTP generados con la finalidad de simular que es una aplicación móvil la que está realizando este consumo.

4.3.3 Desarrollo de la implementación de un servicio de validación de One-Time Password (OTP)

La fase de desarrollo de este incremento comienza con la implementación de un contenedor y la creación de Endpoints, Código 9, correspondientes en la API y la aplicación móvil para permitir el intercambio de valores de OTP generados.

```python
@app.route('/api/external', methods=['POST'])
def get_external_value():
    external_value = request.json.get('value')
    if external_value is None:
        return jsonify({'error': 'No se recibió ningún valor externo'}),
400 else:
        return jsonify({'value': external_value}), 200

@app.route('/api/mobile', methods=['POST'])
def check_values():
    mobile_value = request.json.get('value')
    if mobile_value is None:
        return jsonify({'error': 'No se recibió ningún valor móvil'}), 400
    else:
        external_value = request.json.get('value')
        if mobile_value == external_value:
            return jsonify({'message': 'Los valores son iguales'}), 200
        else:
            return jsonify({'error': 'Los valores no coinciden'}), 400

if __name__ == '__main__':
    app.run()
```

Código 9. Endpoints del servicio de validación.

El código cuenta con dos endpoints en rutas específicas, /api/external y /api/mobile, que usan el método POST para recibir valores. En el primer endpoint, se realiza una verificación en el cuerpo de la solicitud para determinar si se ha recibido algún valor desde la otra API. Si el valor existe, se devuelve el valor correspondiente. En caso contrario, se retorna un mensaje de error con un código de estado 400.

En el segundo endpoint, se realiza una verificación similar en el cuerpo de la solicitud para determinar si se ha recibido algún valor desde la aplicación móvil. Si no se recibe ningún valor, se retorna un mensaje de error con un código de estado 400. En caso de que se reciba un valor, se compara con el valor externo recibido en el primer endpoint. Si los valores son iguales, se devuelve un mensaje indicando que los valores coinciden, junto con un código de estado 200. En cambio, si los valores no coinciden, se retorna un mensaje de error con un código de estado 400.

4.3.4 Pruebas de la implementación de un servicio de validación de One-Time Password (OTP)

Para llevar a cabo las pruebas correspondientes a este incremento, se generaron valores de OTP que fueron enviados a los endpoints implementados. Con el fin de garantizar el éxito de las pruebas de verificación, se realizaron varias modificaciones en los valores de OTP, las cadenas de RSA que se enviaban y las claves públicas que simulaban ser la aplicación móvil. Además, se enviaron datos completos e incompletos, valores en texto plano y valores erróneos. Todo lo anterior se hizo con la finalidad de validar que los endpoints funcionaran correctamente ante diversos escenarios y casos de uso.

4.4 Creación de un Bot para Telegram y Prototipo de Sistema Web

4.4.1 Análisis de la creación de un Bot para Telegram y prototipo de sistema web

Con respecto al cuarto requerimiento, se ha tomado en cuenta la creación de un Bot para la aplicación de mensajería Telegram con el fin de proporcionar al usuario el valor del OTP generado. Telegram es una aplicación de mensajería altamente segura que ofrece diversas funcionalidades para garantizar la seguridad de las comunicaciones [11], [13].

En este caso en particular, se ha implementado la función de Spoiler en la aplicación de Telegram para agregar una capa extra de seguridad al momento de entregar el valor del OTP. El Spoiler es una característica que permite ocultar cierto contenido dentro de un

mensaje, lo que brinda una mayor protección contra posibles ataques de terceros que pudieran interceptar la información.

Adicionalmente, se ha creado el prototipo de un sistema web que simula el proceso de inicio de sesión. Para llevar a cabo esta tarea, se ha diseñado una página de Login que permite al usuario ingresar sus credenciales de autenticación y se ha desarrollado un sistema que posibilita la configuración y selección de una forma de generación de OTP.

El prototipo de sistema web se ha creado con el propósito de emular el proceso de inicio de sesión en un entorno controlado y seguro, permitiendo la realización de pruebas y evaluaciones para verificar el correcto funcionamiento de los mecanismos de autenticación implementados.

4.4.2 Diseño de la creación de un Bot para Telegram y prototipo de sistema web

El diseño de la arquitectura para el incremento mencionado se encuentra representado en la **Figura *8***. En esta solución, se despliega un contenedor que aloja el Bot de Telegram. Para efectuar la comunicación del valor OTP generado y su posterior envío al Bot, el contenedor debe contar con una API que conecte tanto con la API que se encarga de obtener los valores como con el servicio de validación.

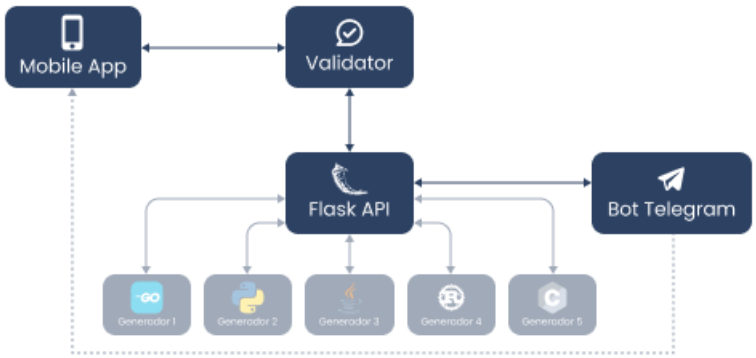


Figura 8. Implementación de la arquitectura del servicio de envío de valores OTP.

Además, el servicio de Bot tiene la capacidad de enviar el valor OTP generado al ChatID asociado con el usuario. La obtención y configuración del ChatID se realizará una vez se implemente el prototipo de sistema web, ya que allí se le permitirá al usuario configurar su cuenta en su dispositivo móvil.

Una vez el usuario configure su cuenta en el dispositivo móvil y el sistema web registre el ChatID asociado, el servicio de Bot podrá enviar el valor OTP generado a dicho ChatID para que el usuario pueda proceder con el proceso de autenticación.

Es importante destacar que el ChatID es un identificador único que se asigna a cada conversación en la aplicación de mensajería Telegram y es necesario para que el servicio de Bot pueda enviar mensajes al usuario.

Por último, se ha desarrollado un servicio que permite el acceso a través del navegador y la simulación del proceso de Login en la aplicación. La arquitectura utilizada para este servicio se encuentra representada en la Figura 9.

Este servicio se ha diseñado con el objetivo de brindar una interfaz de usuario amigable y de fácil uso que permita al usuario final interactuar con la aplicación y realizar pruebas de autenticación en un entorno seguro y controlado.

La arquitectura empleada para este servicio se basa en la utilización de tecnologías web y se compone de diversas capas que se encargan de gestionar el flujo de datos y de interacción entre el usuario y la aplicación.

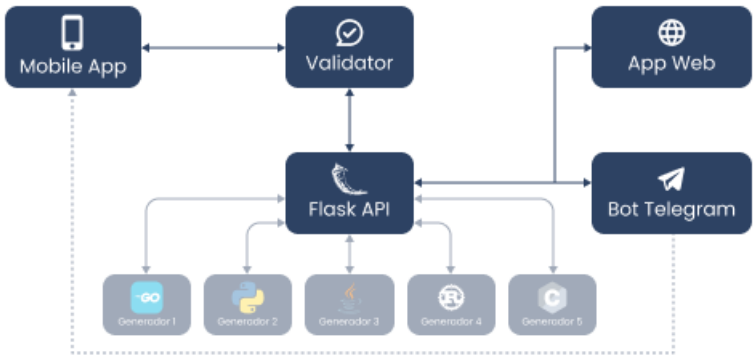


Figura 9. Implementación del servicio web para realizar el proceso de login.

Para la estructura web del prototipo, se optó por implementar el mismo con Bootstrap. Bootstrap es un framework de diseño web de código abierto que proporciona herramientas y recursos para crear sitios web responsivos y modernos. Uno de los principales beneficios de Bootstrap es que proporciona una base sólida y coherente para el diseño de un sitio web, lo que puede ahorrar mucho tiempo y esfuerzo en la creación de la interfaz de usuario. Además, Bootstrap también incluye una amplia gama de componentes y widgets, como botones, formularios, tablas y menús, que se pueden integrar fácilmente en un sitio web [49].

En la Figura 10 podemos ver el Flowchart del prototipo de aplicación web. Un flowchart de un sitio web es un diagrama visual que muestra la estructura y organización de las páginas web de un sitio. Es una herramienta que ayuda a planificar y diseñar la navegación de un sitio web.

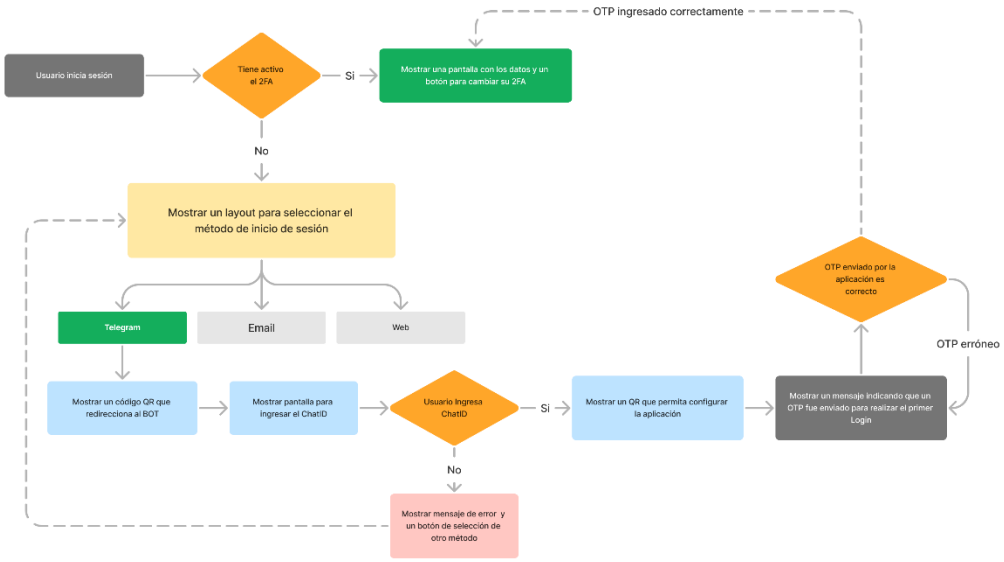


Figura 10. Flowchart del prototipo del sistema web.

4.4.3 Desarrollo de la creación de un Bot para Telegram y prototipo de sistema web

Para la implementación de un bot en Telegram, se requiere realizar una interacción previa con Botfather, el cual es un bot de Telegram diseñado específicamente para guiar al usuario en el proceso de creación del bot que se desea implementar.

El proceso de configuración del bot comienza iniciando una conversación con Botfather en Telegram y seleccionando la opción "menu", ubicada en la parte inferior del chat. A continuación, se debe seleccionar la opción /newbot, la cual proporcionará las instrucciones necesarias para crear el bot deseado. Este proceso de creación del bot involucra la asignación de un nombre y un nombre de usuario al bot, lo que permitirá a los usuarios identificarlo y comunicarse con él en Telegram.

La Figura 11 muestra la configuración específica utilizada para la creación del bot que se utilizará en el presente trabajo de integración curricular. Es fundamental que se sigan las instrucciones proporcionadas por Botfather con atención para garantizar que el bot sea creado de manera efectiva y para poder obtener el Token correspondiente.

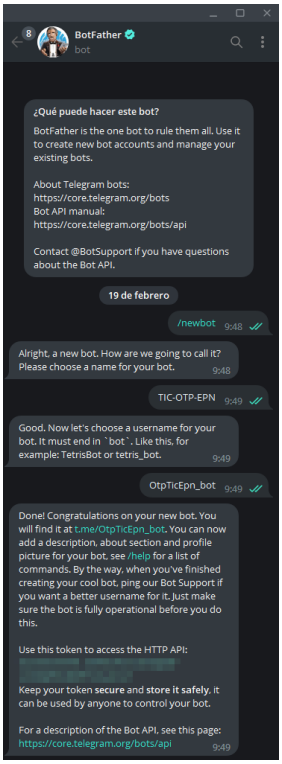


Figura 11. Configuración realizada en Botfather para el Bot.

La respuesta proporcionada por Botfather incluye el Token de acceso correspondiente a la API HTTP que se utilizará para interactuar con el bot. Este Token es un valor único y confidencial que deberá ser utilizado como variable de entorno en el servicio encargado de la entrega de OTP (One-Time Password).

Una vez que se ha obtenido la API, se procederá a programar el servicio de envío de mensajes. El Código 10 muestra la configuración del endpoint utilizado para enviar un mensaje específico a un ChatID determinado. La implementación completa del Bot se encuentra disponible en el Anexo II.

```python
@messages.route('/send-message', methods=['POST'])
def send_message():
    message = request.json.get('message')

        # Especificamos el ChatID
    chat_id = '<chat_id>'

    TelegramConfig.bot.send_message(chat_id=chat_id, text=message)

    return {'message': 'Message sent'}
```

Código 10. Endpoint para enviar un mensaje de Telegram a un ChatID específico.

El proceso de creación de los mockups de alto nivel se llevó a cabo mediante el uso de Figma, una herramienta que permite diseñar interfaces de usuario y prototipos interactivos [50]. Para ello, se utilizó el Flowchart de la Figura 10 como base y se desarrolló un diseño

tentativo en Figma. A fin de implementar eficientemente el frontend de la aplicación web, se optó por emplear el framework de desarrollo Bootstrap. Bootstrap es un conjunto de herramientas y componentes de CSS que permite desarrollar sitios web responsivos y atractivos en un tiempo reducido. Al combinar Bootstrap con Flask, se logra una integración fluida entre el Frontend y el Backend de la aplicación, lo que contribuye a una mayor eficiencia en el proceso de desarrollo.

El Código 11 incluye un endpoint en una aplicación web desarrollada con Flask, que utiliza Bootstrap para renderizar una plantilla de inicio de sesión. La función asociada al endpoint verifica si la solicitud es de tipo GET o POST, y en caso de que sea de tipo GET, se renderiza el formulario de inicio de sesión. Por otro lado, si la solicitud es de tipo POST, la función procede a validar los datos de inicio de sesión ingresados por el usuario, y en caso de que sean correctos, se redirecciona al usuario a la plantilla de Home, donde se encuentran los datos correspondientes a la sesión iniciada.

En caso de que los datos ingresados sean incorrectos, la función renderiza nuevamente el formulario de inicio de sesión, esta vez con un mensaje de error que indica que los datos ingresados no son válidos. El uso de Bootstrap en la construcción de la plantilla de inicio de sesión permite una mayor personalización y flexibilidad en la interfaz de usuario, así como una mejor visualización de la página en diferentes dispositivos y tamaños de pantalla.

```python
from flask import Flask, render_template, request, redirect, url_for
from flask_bootstrap import Bootstrap

app = Flask(__name__)
bootstrap = Bootstrap(app)

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        if username == 'user' and password == 'pass':
            return redirect(url_for('home'))
        else:
            error = 'Nombre de usuario con contraseña invalidos'
            return render_template('login.html', error=error)
    return render_template('login.html')
```

Código 11. Endpoint para renderizar la plantilla de login en el prototipo de sistema web.

4.4.4 Pruebas de la creación de un Bot para Telegram y prototipo de sistema web

Se realizaron pruebas para verificar el correcto funcionamiento de la integración de los servicios. Específicamente, se enviaron valores OTP a un ChatID específico y se transmitieron a la API de validación a través de Postman. Estas pruebas fueron llevadas a cabo antes y después de la integración con la funcionalidad del Backend para verificar la funcionalidad y flujo del proceso establecido en el Flowchart.

Además, se realizaron pruebas para el prototipo del sistema web, centradas en la verificación de su funcionalidad y flujo del proceso establecido en el Flowchart. Los mockups de alto nivel creados previamente en Figma, se encuentran disponibles en el Anexo III del documento. Las pruebas realizadas en Postman para verificar el correcto funcionamiento del sistema web se incluyen en el Anexo IV.

Finalmente, después de la implementación de la interfaz de usuario, se realizó un estudio de usabilidad utilizando el Sistema de Escalas de Usabilidad (SUS), ampliamente utilizado debido a su eficacia en la medición de la satisfacción del usuario y su facilidad de aplicación. El SUS consiste en un conjunto de preguntas con una escala de respuesta de cinco puntos, enfocándose en la evaluación de aspectos como la facilidad de uso y la eficacia del sistema.

5 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

5.1 Resultados

En esta sección se realizó el análisis de los resultados de todas las pruebas del prototipo. Para el primer incremento se realizó una prueba de bondad de ajuste utilizando la técnica estadística conocida como Chi cuadrado, denotado por χ^2, la cual se utiliza comúnmente en el análisis de números aleatorios para verificar si la secuencia generada se ajusta a una distribución uniforme. El propósito de aplicar esta técnica era obtener el estadístico de prueba para determinar qué programa de los desplegados en los contenedores era más aleatorio que los demás. Los detalles específicos sobre el cálculo se encuentran en el Anexo IV del presente trabajo.

La Figura 12 muestra el histograma de los valores generados por un programa implementado en Golang. A partir de la observación del histograma, se puede inferir que los valores generados por el programa están distribuidos de manera uniforme en el rango permitido, lo que significa que cada número tiene la misma probabilidad de ser generado.

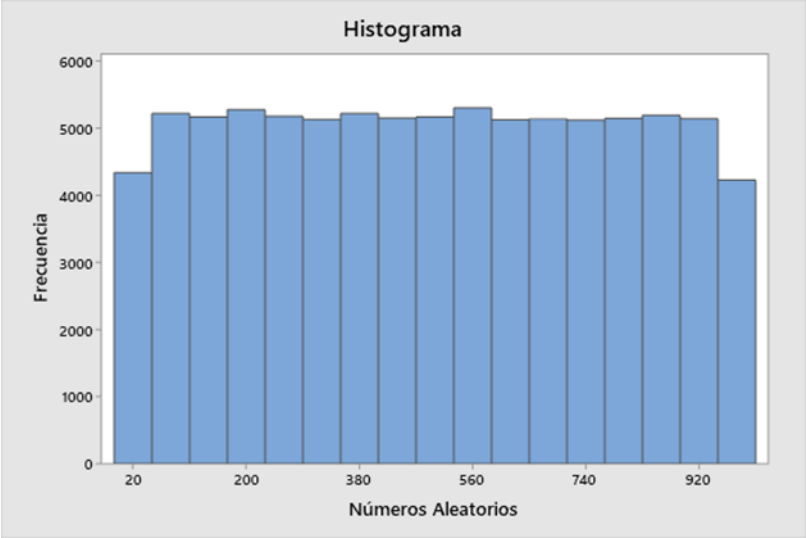


Figura 12. Histograma de los valores generados por Golang.

Al realizar el cálculo del estadístico de prueba, se pueden observar los valores obtenidos por los programas desplegados en los contenedores en la Tabla 1, lo que nos indica cuan aleatorio es cada programa.

Tabla 1. Valor del estadístico de prueba para cada lenguaje.

Contenedor	Valor
Golang	16,5702546
Python	24,3894675
Rust	20,6510416
Java	20,4479638
C++	11,6744907

La API proporciona un medio seguro para establecer una conexión entre diferentes servicios, como el servicio de validación y el servicio de envío de One-Time Password (OTP). Esta conexión asegura la fiabilidad y la protección de los datos transmitidos a través de la implementación de técnicas de cifrado.

Para proteger la privacidad de los valores OTP generados, se utiliza una función de Telegram llamada "Spoiler". Esta opción difumina el contenido del mensaje para asegurar que solo el usuario pueda ver el valor. La Figura 13.a muestra el mensaje antes de desvelar la información confidencial, mientras que la Figura 13.b presenta el mensaje en texto claro una vez que se ha desvelado la información.

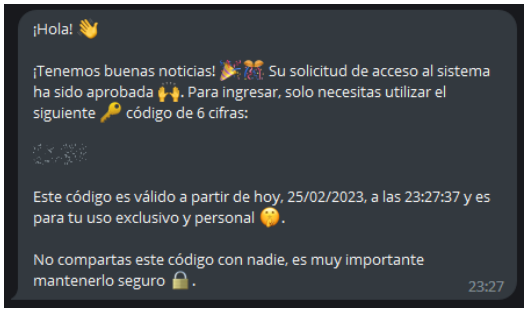


Figura 13.a. Mensaje cifrado con información confidencial

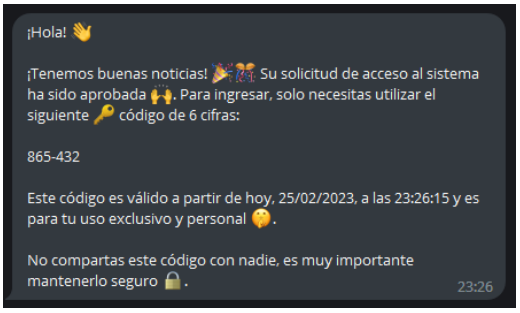


Figura 13.b. Mensaje cifrado con información confidencial

Figura 13. Comparación del mensaje antes y después de desvelar información confidencial.

Una vez que el usuario ha adquirido el código y lo ha ingresado en la aplicación, ésta establecerá contacto con el servicio de validación. En este sentido, se han llevado a cabo pruebas para evaluar la latencia en el envío de dichos valores. Como resultado de estas pruebas, se ha determinado que la latencia promedio es de 50 milisegundos, lo que sugiere un alto rendimiento en el procesamiento y la transmisión de datos en la aplicación.

Para llevar a cabo las pruebas de usabilidad del aplicativo web y de la opción de entrega y visualización del valor OTP, se procedió a realizar una prueba de sistema de usabilidad (SUS), la cual consistió en la aplicación de un conjunto de preguntas que se encuentran detalladas en la Tabla 2. Cabe destacar que, en esta encuesta, las preguntas pares representan puntos negativos del sistema, mientras que las preguntas impares representan puntos positivos, los resultados de cada una de las encuestas se encuentra en el Anexo VI.

La prueba de SUS fue llevada a cabo con la participación de un total de 15 individuos, lo cual supera el número mínimo sugerido para este tipo de pruebas, que se sitúa en 5 participantes [51]. Sin embargo, es importante señalar que, si bien algunos expertos en usabilidad indican que se pueden obtener resultados útiles con solo 3 participantes [52], se recomienda utilizar el número más alto posible de participantes para obtener resultados más precisos y confiables, siempre y cuando los recursos y el tiempo disponible lo permitan.

Tabla 2. Preguntas para la prueba SUS.

Preguntas planteadas al usuario
Creo que el envío del OTP utilizando la nueva propuesta de Telegram fue muy fácil

Creo que el envío del OTP utilizando el método tradicional fue más fácil que el método propuesto
Creo que el proceso de envío de OTP utilizando la nueva propuesta de Telegram fue bastante rápido
Creo que el proceso de envío de OTP utilizando el método propuesto fue bastante lento
Creo que la nueva propuesta de Telegram para enviar el OTP es segura
Creo que el método propuesto para enviar el OTP es inseguro
Creo que prefiero el método propuesto porque me parece más fácil de utilizar
Creo que prefiero utilizar los métodos tradicionales para obtener el OTP
Creo que recomendaría la nueva propuesta de Telegram a un amigo o colega
Creo que el método propuesto tiene algunos problemas que deben ser corregidos por eso no lo recomendaría

El proceso de prueba consistió en proporcionar a cada participante una explicación del concepto de One-Time Password (OTP) y demostrar el proceso de registro en aplicaciones de uso común. Posteriormente, se guio a los participantes en el proceso de registro en nuestro prototipo de aplicación web, brindándoles asistencia en la configuración de la autenticación de doble factor (2FA), con la opción de seleccionar Telegram como el método de entrega preferido. La Figura 14 muestra el prototipo diseñado para este componente y utilizado en las pruebas con los usuarios, mientras que el Anexo II proporciona un el código fuente este.

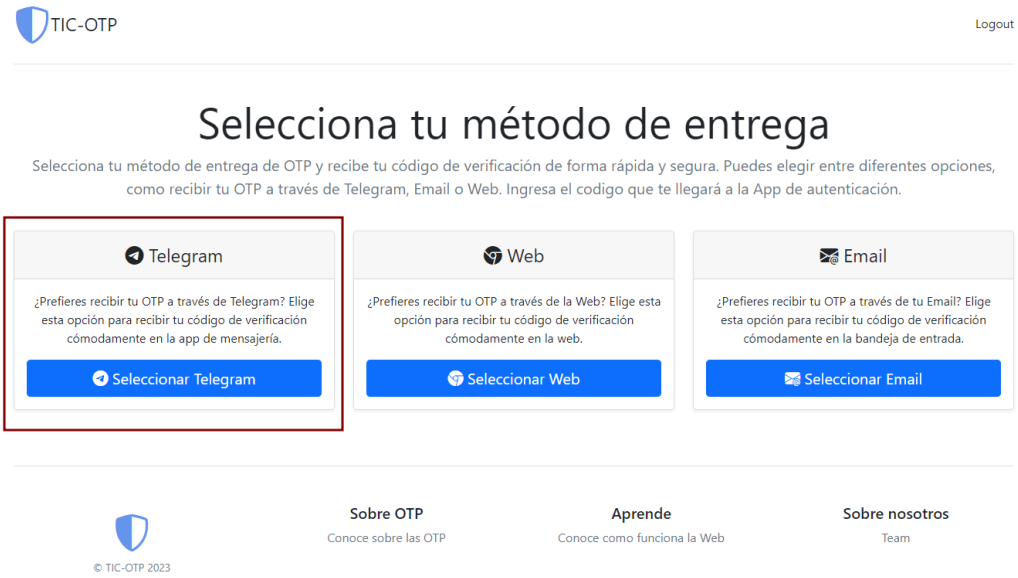


Figura 14. Interfaz de usuario utilizada en la encuesta de usabilidad SUS.

Los resultados obtenidos en las pruebas se presentan en la Tabla 3, en la que se muestra el promedio de las calificaciones obtenidas al encuestar a quince usuarios, así como los valores máximos y mínimos obtenidos en las encuestas realizadas. Es importante destacar que el promedio no es un porcentaje, sino una calificación que proporciona información sobre la facilidad de uso y la satisfacción de los usuarios con el producto desarrollado.

Tabla 3. Resultados obtenidos en la prueba de usabilidad SUS.

Resultado	Total
Promedio	73.33
Máximo	85
Mínimo	60

5.2 Conclusiones

La implementación de esta solución proporciona una forma segura y confiable de obtener valores OTP de seis cifras, lo que es esencial en sistemas que requieran un alto nivel de seguridad y protección de datos.

El índice de repetición de valores OTP se refiere al número de veces que se generan valores idénticos en una secuencia de contraseñas OTP. En la Tabla 1 se muestran los resultados de las pruebas estadísticas realizadas para comparar el nivel de aleatoriedad producido por diferentes contenedores. Se puede concluir que, de acuerdo con los valores obtenidos, C++ es el contenedor que genera mayor aleatoriedad, seguido por Golang, Java, Rust y finalmente Python.

El riesgo de ataques de adivinación o ataques de repetición aumenta proporcionalmente con el índice de repetición utilizado en la generación de valores OTP. Para mitigar este riesgo, se emplea una técnica de combinación de dos números de tres cifras generados aleatoriamente por un contenedor generador seleccionado al azar. Esta técnica de combinación de números es una medida efectiva para reducir el riesgo de ataques de adivinación y proteger la integridad de los valores OTP generados.

La aleatoriedad juega un papel crucial en muchas aplicaciones que requieren un alto nivel de seguridad y fiabilidad. Los resultados obtenidos en este trabajo demuestran que la elección del contenedor adecuado puede afectar significativamente la calidad de la aleatoriedad generada. Por lo tanto, es fundamental evaluar cuidadosamente los contenedores generadores disponibles y seleccionar aquellos que ofrecen el nivel de aleatoriedad requerido.

La seguridad informática es fundamental en la transmisión de información, ya que existen riesgos asociados a la exposición y acceso no autorizado de datos sensibles. En este sentido, se deben implementar medidas de protección y seguridad adecuadas para garantizar la confidencialidad, integridad y disponibilidad de la información transmitida. Es importante utilizar herramientas y tecnologías apropiadas para establecer conexiones seguras, el uso de funciones de privacidad, como "Spoiler" de Telegram, puede ser útil para garantizar la privacidad y confidencialidad de la información transmitida.

La realización de pruebas de latencia en el envío de valores a través de la aplicación es crucial para garantizar un alto rendimiento en el procesamiento y transmisión de datos. Con esta información, se pueden optimizar los tiempos de respuesta de la aplicación y mejorar la experiencia de usuario.

Bootstrap es herramienta altamente efectiva para mejorar la usabilidad de los sitios web. Su conjunto de estilos predefinidos y componentes interactivos, permiten organizar el contenido de manera efectiva en diferentes tamaños de pantalla, facilitando la navegación del usuario y haciendo que el sitio web sea atractivo y funcional. Bootstrap como

herramienta para mejorar la usabilidad, puede significar una gran ventaja competitiva en la creación de sitios web modernos y eficientes.

La nota obtenida en el estudio SUS indica que la facilidad de uso y la satisfacción del usuario en relación con el envío del OTP utilizando la nueva propuesta de Telegram es razonablemente alta. Específicamente, la nota obtenida de 73,33 sugiere que la mayoría de los usuarios evaluados encontraron la nueva propuesta de Telegram para obtener el OTP relativamente fácil de usar, de configurar y segura.

Es importante mencionar que se encontraron ciertos aspectos específicos que requieren mejoras, tales como la configuración inicial del Bot o el bajo uso que los usuarios le dan a la aplicación Telegram. Estos resultados proporcionan información valiosa que puede ser utilizada para mejorar la propuesta de entrega del valor OTP y aumentar aún más su facilidad de uso y satisfacción del usuario.

Es necesario analizar en profundidad estos aspectos identificados y determinar acciones concretas para abordarlos y mejorar la experiencia del usuario en relación a la recepción del OTP utilizando la nueva propuesta de Telegram.

La combinación del algoritmo de cifrado RSA y de contenedores seguros es una solución eficaz para proteger la información confidencial y prevenir amenazas de seguridad en los sistemas informáticos. La encriptación RSA, basada en la teoría de números, es una de las técnicas de cifrado más seguras y robustas, mientras que los contenedores seguros proporcionan una capa adicional de protección separando y protegiendo los datos sensibles del resto del sistema. Para garantizar su efectividad, es importante evaluar cuidadosamente las necesidades de seguridad del sistema y mantener la implementación actualizada frente a las amenazas y vulnerabilidades existentes.

La implementación de un segundo factor de autenticación es una técnica fundamental en la ciberseguridad para asegurar la protección de los sistemas informáticos. Además, es crucial establecer políticas de contraseñas robustas para los usuarios que acceden a los sistemas, a fin de evitar posibles vulnerabilidades en la autenticación. La combinación de ambas prácticas aumenta la seguridad del sistema y minimiza el riesgo de ataques de fuerza bruta que podrían comprometer la integridad de los datos del sistema.

5.3 Recomendaciones

Se sugiere la implementación de un sistema de registro de eventos para mejorar la capacidad de los administradores en la detección temprana de posibles actividades maliciosas en el sistema. La implementación de un sistema de registro de eventos también

puede ser útil en la evaluación del rendimiento del sistema, permitiendo a los administradores optimizar la configuración del sistema para mejorar su eficiencia y rendimiento.

Se recomienda la realización de auditorías periódicas en el caso de llevar el prototipo a producción, esto con el fin de identificar posibles vulnerabilidades de seguridad y prevenir ataques. Las auditorías permiten analizar y detectar posibles fallas y brechas de seguridad, lo que ayuda a los administradores a implementar medidas de seguridad adecuadas para proteger la información. La realización de auditorías periódicas también garantiza que la solución se mantenga actualizada y que las medidas de seguridad estén en línea con las últimas tendencias y prácticas recomendadas.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] M. Shirvanian and S. Agrawal, "2D-2FA: A New Dimension in Two-Factor Authentication," in *Annual Computer Security Applications Conference*, Dec. 2021, pp. 482–496. doi: 10.1145/3485832.3485910.

[2] S. Ma *et al.*, "An empirical study of SMS one-time password authentication in Android apps," in *Proceedings of the 35th Annual Computer Security Applications Conference*, Dec. 2019, pp. 339–354. doi: 10.1145/3359789.3359828.

[3] S. Ma *et al.*, "Fine with '1234'? An Analysis of SMS One-Time Password Randomness in Android Apps," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, May 2021, pp. 1671–1682. doi: 10.1109/ICSE43902.2021.00148.

[4] L. Lamport, "Password authentication with insecure communication," *Commun ACM*, vol. 24, no. 11, pp. 770–772, Nov. 1981, doi: 10.1145/358790.358797.

[5] I. Tzemos, A. P. Fournaris, and N. Sklavos, "Security and Efficiency Analysis of One Time Password Techniques," in *Proceedings of the 20th Pan-Hellenic Conference on Informatics*, Nov. 2016, pp. 1–5. doi: 10.1145/3003733.3003791.

[6] M. J. Zadeh and H. Barati, "Security Improvement in Mobile Banking Using Hybrid Authentication," in *Proceedings of the 2019 3rd International Conference on Advances in Artificial Intelligence*, Oct. 2019, pp. 198–201. doi: 10.1145/3369114.3369151.

[7] S. P. Wankhade and A. N. Bandal, "Security for Automation in Internet of Things Using One Time Password," in *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, Aug. 2017, pp. 1–6. doi: 10.1109/ICCUBEA.2017.8463777.

[8] E. M. M. Manucom, B. D. Gerardo, and R. P. Medina, "Analysis of Key Randomness in Improved One-Time Pad Cryptography," in *2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, Oct. 2019, pp. 11–16. doi: 10.1109/ICASID.2019.8925173.

[9] F. Shah-Mohammadi, W. Cui, K. Bachi, Y. Hurd, and J. Finkelstein, "Using Natural Language Processing of Clinical Notes to Predict Outcomes of Opioid Treatment Program," in *2022*

44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), Jul. 2022, pp. 4415–4420. doi: 10.1109/EMBC48229.2022.9871960.

[10] N. Bunzel, T. Chen, and M. Steinebach, "Using Telegram as a carrier for image steganography: Analysing Telegrams API limits," in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, Aug. 2022, pp. 1–8. doi: 10.1145/3538969.3544440.

[11] T. Sušánka and J. Kokeš, "Security Analysis of the Telegram IM," in *Proceedings of the 1st Reversing and Offensive-oriented Trends Symposium*, Nov. 2017, pp. 1–8. doi: 10.1145/3150376.3150382.

[12] N. Hema and J. Yadav, "Secure Home Entry Using Raspberry Pi with Notification via Telegram," in *2020 6th International Conference on Signal Processing and Communication (ICSC)*, Mar. 2020, pp. 211–215. doi: 10.1109/ICSC48311.2020.9182778.

[13] A. Candra, Y. Kurniawan, and K.-H. Rhee, "Security analysis testing for secure instant messaging in android with study case: Telegram," in *2016 6th International Conference on System Engineering and Technology (ICSET)*, Oct. 2016, pp. 92–96. doi: 10.1109/ICSEngT.2016.7849630.

[14] F. Rochet, K. Efthymiadis, F. Koeune, and O. Pereira, "SWAT: Seamless Web Authentication Technology," in *The World Wide Web Conference*, May 2019, pp. 1579–1589. doi: 10.1145/3308558.3313637.

[15] A. A. S. AlQahtani, Z. El-Awadi, and M. Min, "A Survey on User Authentication Factors," in *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Oct. 2021, pp. 0323–0328. doi: 10.1109/IEMCON53756.2021.9623159.

[16] S. Ibrokhimov, K. L. Hui, A. Abdulhakim Al-Absi, hoon jae lee, and M. Sain, "Multi-Factor Authentication in Cyber Physical System: A State of Art Survey," in *2019 21st International Conference on Advanced Communication Technology (ICACT)*, Feb. 2019, pp. 279–284. doi: 10.23919/ICACT.2019.8701960.

[17] S. Wiefling, M. Dürmuth, and L. lo Iacono, "More Than Just Good Passwords? A Study on Usability and Security Perceptions of Risk-based Authentication," in *Annual Computer Security Applications Conference*, Dec. 2020, pp. 203–218. doi: 10.1145/3427228.3427243.

[18] S. Ruoti, B. Roberts, and K. Seamons, "Authentication Melee," in *Proceedings of the 24th International Conference on World Wide Web*, May 2015, pp. 916–926. doi: 10.1145/2736277.2741683.

[19] M. Theofanos, S. Garfinkel, and Y.-Y. Choong, "Secure and Usable Enterprise Authentication: Lessons from the Field," *IEEE Secur Priv*, vol. 14, no. 5, pp. 14–21, Sep. 2016, doi: 10.1109/MSP.2016.96.

[20] S. A. Lone and A. H. Mir, "A novel OTP based tripartite authentication scheme," *International Journal of Pervasive Computing and Communications*, vol. 18, no. 4, pp. 437–459, Jul. 2022, doi: 10.1108/IJPCC-04-2021-0097.

[21] Fortinet, "FortiToken One-Time Password Token," *https://www.fortinet.com/content/dam/fortinet/assets/data-sheets/fortitoken.pdf*, Nov. 29, 2022.

[22] M. Imanullah and Y. Reswan, "Randomized QR-code scanning for a low-cost secured attendance system," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 4, p. 3762, Aug. 2022, doi: 10.11591/ijece.v12i4.pp3762-3769.

[23] E. Erdem and M. T. Sandikkaya, "OTPaaS—One Time Password as a Service," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 743–756, Mar. 2019, doi: 10.1109/TIFS.2018.2866025.

[24] B. B. Balilo, B. D. Gerardo, R. P. Medina, and Y. Byun, "Design of physical authentication based on OTP KeyPad," in *2017 International Conference on Applied Computer and Communication Technologies (ComCom)*, May 2017, pp. 1–5. doi: 10.1109/COMCOM.2017.8167082.

[25] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen, "HOTP: An HMAC-Based One-Time Password Algorithm," Dec. 2005. doi: 10.17487/rfc4226.

[26] D. M'Raihi, S. Machani, M. Pei, and J. Rydell, "TOTP: Time-Based One-Time Password Algorithm," May 2011. doi: 10.17487/rfc6238.

[27] N. Haller, C. Metz, P. Nesser, and M. Straw, "A One-Time Password System," Feb. 1998. doi: 10.17487/rfc2289.

[28] N. Nassar and L.-C. Chen, "Seed-based authentication," in *2015 International Conference on Collaboration Technologies and Systems (CTS)*, Jun. 2015, pp. 345–350. doi: 10.1109/CTS.2015.7210447.

[29] F. Yu, L. Li, Q. Tang, S. Cai, Y. Song, and Q. Xu, "A Survey on True Random Number Generators Based on Chaos," *Discrete Dyn Nat Soc*, vol. 2019, pp. 1–10, Dec. 2019, doi: 10.1155/2019/2545123.

[30] L. Bassham *et al.*, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications." Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, Feb. 2010. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=906762

[31] S. Tupparwar and M. N., "A Hybrid True Random Number Generator using Ring Oscillator and Digital Clock Manager," in *2021 6th International Conference on Inventive Computation Technologies (ICICT)*, Jan. 2021, pp. 290–294. doi: 10.1109/ICICT50816.2021.9358750.

[32] Sarita and S. Sebastian, "Transform Monolith into Microservices using Docker," in *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, Aug. 2017, pp. 1–5. doi: 10.1109/ICCUBEA.2017.8463820.

[33] D. Jaramillo, D. v Nguyen, and R. Smart, "Leveraging microservices architecture by using Docker technology," in *SoutheastCon 2016*, Mar. 2016, pp. 1–5. doi: 10.1109/SECON.2016.7506647.

[34] A. Naik, J. Choudhari, V. Pawar, and S. Shitole, "Building an EdTech Platform Using Microservices and Docker," in *2021 IEEE Pune Section International Conference (PuneCon)*, Dec. 2021, pp. 1–6. doi: 10.1109/PuneCon52575.2021.9686535.

[35] B. A. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering," Feb. 2007. [Online]. Available: https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf

[36] Notion, "What is notion?," *Notion*. [Online]. Available: https://www.notion.so/help/guides/what-is-notion

[37] M. R. Mufid, A. Basofi, M. U. H. al Rasyid, I. F. Rochimansyah, and A. rokhim, "Design an MVC Model using Python for Flask Framework Development," in *2019 International Electronics Symposium (IES)*, Sep. 2019, pp. 214–219. doi: 10.1109/ELECSYM.2019.8901656.

[38] The Pallets Projects, "Flask," *Pallets*. [Online]. Available: https://palletsprojects.com/p/flask/

[39] Docker, "What is a container?," *Docker*. Feb. 2023. [Online]. Available: https://www.docker.com/resources/what-container/

[40] D. Reis, B. Piedade, F. F. Correia, J. P. Dias, and A. Aguiar, "Developing Docker and Docker-Compose Specifications: A Developers' Survey," *IEEE Access*, vol. 10, pp. 2318–2329, 2022, doi: 10.1109/ACCESS.2021.3137671.

[41] M. G. Ginestà and O. P. Mora, "Bases de datos en PostgreSQL," *Sl]:[sn]*, 2012.

[42] C. Larman and V. R. Basili, "Iterative and incremental developments. a brief history," *Computer (Long Beach Calif)*, vol. 36, no. 6, pp. 47–56, 2003.

[43] N. P. L. Larrea, M. V. R. Valencia, S. A. S. Cazco, and B. A. B. Hermida, "Análisis de los lenguajes de programación más utilizados en el desarrollo de aplicaciones web y móviles," *Domino de las Ciencias*, vol. 8, no. 3, pp. 1601–1625, 2022.

[44] M. McGrath, *GO Programming in easy steps: Discover Google's Go language (golang)*. In Easy Steps Limited, 2020.

[45] G. van Rossum and others, "Python Programming Language.," in *USENIX annual technical conference*, 2007, vol. 41, no. 1, pp. 1–36.

[46] Y. Li, "Computer Software Java Programming Optimization Design," in *Frontier Computing: Proceedings of FC 2021*, Springer, 2022, pp. 1086–1092.

[47] A. N. Evans, B. Campbell, and M. lou Soffa, "Is Rust used safely by software developers?," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 246–257.

[48] M. Barr, *Programming embedded systems in C and C++*. " O'Reilly Media, Inc.," 1999.

[49] Z. A. Rozi and others, *Bootstrap design framework*. Elex Media Komputindo, 2015.

[50] S. BAZOALTO CRUZ, "FIGMA COMO HERRAMIENTA PARA EL DISEÑO DE MUCKUPS," 2022.

[51] J. Nielsen, "Why You Only Need to Test with 5 Users." 2000.

[52] J. Nielsen, “How Many Test Users in a Usability Study?,” 2012, [Online]. Available: https://www.nngroup.com/articles/how-many-test-users/

7 ANEXOS

7.1 Anexo I - Recopilación de consideraciones de seguridad de múltiples papers.

Anexo I - Consideraciones de Seguridad

https://epnecuador-my.sharepoint.com/personal/luis_almeida01_epn_edu_ec/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fluis%5Falmeida01%5Fepn%5Fedu%5Fec%2FDocuments%2FTIC%2DAlmeidaLuis%2FAnexo%20I%20%2D%20Consideraciones%20de%20Seguridad&ga=1

En el presente anexo se incluye un archivo que contiene la recopilación de diversas consideraciones de seguridad, junto con su respectiva categorización. Así mismo, el archivo ofrece una descripción detallada de las consideraciones que se han tenido en cuenta para el proyecto actual.

7.2 Anexo II – Código completo del prototipo.

Anexo II - Repositorio de Código

https://epnecuador-my.sharepoint.com/personal/luis_almeida01_epn_edu_ec/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fluis%5Falmeida01%5Fepn%5Fedu%5Fec%2FDocuments%2FTIC%2DAlmeidaLuis%2FAnexo%20II%20%2D%20Repositorio%20de%20C%C3%B3digo&ga=1

El anexo, contiene un enlace al repositorio de código correspondiente, donde se incluyen las instrucciones detalladas paso a paso para la ejecución y replicación del presente proyecto.

7.3 Anexo III – Mockups de alto nivel desarrollados en Figma

Anexo III - Mockups de alto nivel desarrollados en Figma

https://epnecuador-my.sharepoint.com/:f:/g/personal/luis_almeida01_epn_edu_ec/EvoTsMsuWSJMtipp7g5KR-UBsn-w0IFV0T9tCvSXCsJ6TQ?e=MvLfdx

Se facilita un enlace que permite acceder a los recursos de diseño de la aplicación web, incluyendo el Mockup de alto nivel desarrollado en Figma y los diseños generados en esta herramienta.

7.4 Anexo IV - Pruebas realizadas con la herramienta Postman.

Anexo IV - Pruebas realizadas con la herramienta Postman

https://epnecuador-my.sharepoint.com/:f:/g/personal/luis_almeida01_epn_edu_ec/EmfIBfqEFddBj68DUEaeedkBfBqtp01Ph5cSCQz76fvPzA?e=lLLgPI

Se proporcionan las imágenes y los archivos que contienen cada una de las URL y el contenido utilizado para llevar a cabo las pruebas mediante la herramienta Postman.

7.5 Anexo V – Cálculo del estadístico de prueba para cada lenguaje.

7.6 Anexo VI – Resultados de la encuesta de usabilidad SUS

Anexo VI - Resultados de la encuesta de usabilidad SUS

https://epnecuador-my.sharepoint.com/:f:/g/personal/luis_almeida01_epn_edu_ec/EnbhlTdBO3tCmGHFBHPX2fsBL_cNQIuwWZkHRR4VCalCzQ?e=mNqyIU

En el presente anexo, se incluyen los archivos utilizados y los datos obtenidos para elaborar la Tabla 3 con el fin de determinar los resultados de la encuesta de usabilidad SUS.