



Instituto Tecnológico Superior Ecuatoriano De Productividad

Carrera Desarrollo De Software

Trabajo De Titulación Previo A La Obtención del Título de Tecnólogo/A
Superior en Desarrollo de Software

Trabajo De Titulación

AUTENTICACIÓN MULTIFACTOR (MFA) Y CONTROL DE ACCESO BASADO
EN ROLES (RBAC) PARA LA PROTECCIÓN DE ACCESOS A MÓDULOS DE
UN PROTOTIPO DE SOFTWARE

Autor

BRANDON STALYN GAVILANEZ JAGUACO

Director/Tutor

Ing. Wilmer Valle

Quito 2026

© ITSEP. 2026

Reservados todos los derechos de reproducción

FORMULARIO DE REGISTRO BIBLIOGRÁFICO TRABAJO DE TITULACIÓN

DATOS DE CONTACTO	
CÉDULA DE IDENTIDAD:	1728043421
APELLIDO Y NOMBRES:	Gavilanez Jaguaco Brandon Stalyn
DIRECCIÓN:	Aloag
EMAIL:	brandon.gavilanez@itsep.edu.ec
TELÉFONO FIJO:	S/N
TELÉFONO MOVIL:	0980669021
DATOS DE LA OBRA	
TÍTULO:	AUTENTICACIÓN MULTIFACTOR (MFA) Y CONTROL DE ACCESO BASADO EN ROLES (RBAC) PARA LA PROTECCIÓN DE ACCESOS A MÓDULOS DE UN PROTOTIPO DE SOFTWARE
AUTOR O AUTORES:	Gavilanez Jaguaco Brandon Stalyn
FECHA DE ENTREGA DEL PROYECTO DE TITULACIÓN:	16 de octubre del 2026
DIRECTOR DEL PROYECTO DE TITULACIÓN:	Ing. Wilmer Valle
PROGRAMA	PREGRADO ☒ POSGRADO ☐
TÍTULO POR EL QUE OPTA:	Tecnólogo/A Superior en Desarrollo de Software

RESUMEN: Mínimo 250 palabras

El presente trabajo de titulación contempla el estudio del proceso de autenticación centralizado que integra autenticación multifactor mediante contraseñas temporales basadas en tiempo y control de acceso basado en roles, con el objetivo de mitigar los riesgos de suplantación de identidad y garantizar un nivel adecuado de usabilidad en entornos informáticos. La metodología adoptada siguió un enfoque cuantitativo de tipo aplicado y un diseño preexperimental pretest-posttest, estructurado operacionalmente bajo la metodología ágil Scrum a lo largo de cinco sprints. El sistema fue desarrollado en el lenguaje de programación Python utilizando la librería CustomTkinter para la interfaz gráfica de usuario, las bibliotecas pyotp, qrcode y PIL para el motor criptográfico y la generación dinámica de códigos QR, y el motor embebido SQLite3 para la gestión del modelo de datos relacional de la seguridad. Los resultados de las pruebas funcionales demostraron tiempos de respuesta altamente eficientes, registrando la validación del token dinámico en un promedio inferior a 25 milisegundos y la carga de módulos autorizados en menos de 30 milisegundos. Asimismo, la encuesta de diagnóstico aplicada reflejó que el 88% de los usuarios mostró una disposición favorable hacia la adopción del segundo factor y un 94% evaluó el enrolamiento mediante código QR como altamente intuitivo. Se concluye que la superposición del factor de posesión mediante el estándar RFC 6238 en conjunto con la matriz de permisos sobre perfiles desvincula con éxito la dependencia exclusiva de contraseñas estáticas, neutralizando ataques de reutilización de credenciales y reforzando la postura defensiva del sistema de acceso sin generar fricción en la experiencia operacional del usuario.

PALABRAS CLAVES:	Autenticación de Doble Factor, Seguridad Informática, Gestión de Identidades, Tokens Temporales, Permisos de Usuario

ABSTRACT

This degree project examines a centralized authentication process that integrates multi-factor authentication using Time-based One-Time Passwords (TOTP) and Role-Based Access Control (RBAC), aimed at mitigating identity impersonation risks and ensuring an adequate level of usability in IT environments. The methodology adopted a quantitative, applied approach with a pretest-posttest pre-experimental design, operationally structured under the Scrum agile framework across five sprints. The system was developed using the Python programming language, utilizing the CustomTkinter library for the graphical user interface; the pyotp, qrcode, and PIL libraries for the cryptographic engine and dynamic QR code generation; and the embedded SQLite3 engine for managing the security relational data model. Functional testing results demonstrated highly efficient response times, recording dynamic token validation in an average of under 25 milliseconds and authorized module loading in less than 30 milliseconds. Furthermore, the diagnostic survey revealed that 88% of users expressed a favorable willingness to adopt a second authentication factor, while 94% evaluated the onboarding process via QR code as highly intuitive. It is concluded that combining the possession factor through the RFC 6238 standard with a profile permission matrix successfully decouples reliance on static passwords alone, effectively neutralizing credential reuse attacks and strengthening the system's defensive access posture without causing friction in the user's operational experience.

KEYWORDS

Two-Factor Authentication, Computer Security, Identity Management, Temporary Tokens, User Permissions.

DECLARACIÓN Y AUTORIZACIÓN

Yo, Gavilanez Jaguaco Brandon Stalyn, CI: 1728043421 autor/a del trabajo de titulación: **AUTENTICACIÓN MULTIFACTOR (MFA) Y CONTROL DE ACCESO BASADO EN ROLES (RBAC) PARA LA PROTECCIÓN DE ACCESOS A MÓDULOS DE UN PROTOTIPO DE SOFTWARE** previo a la obtención del título de **TECNÓLOGO/A SUPERIOR EN DESARROLLO DE SOFTWARE** en el ITSEP.

1. Declaro tener pleno conocimiento de la obligación que tienen las Instituciones de Educación Superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de titulación de grado para que sea integrado al Sistema Nacional de información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.
2. Autorizo a la BIBLIOTECA del ITSEP a tener una copia del referido trabajo de titulación de grado con el propósito de generar un Repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Quito, 16 de Octubre del 2026

f: _____

Nombre: Brandon Stalyn Gavilanez Jaguaco

Cedula: 1728043421

CERTIFICACIÓN DEL TUTOR

En mi calidad de tutor de tesis de grado, certifico que el presente trabajo que lleva por título **AUTENTICACIÓN MULTIFACTOR (MFA) Y CONTROL DE ACCESO BASADO EN ROLES (RBAC) PARA LA PROTECCIÓN DE ACCESOS A MÓDULOS DE UN PROTOTIPO DE SOFTWARE** para aspirar al título de **TECNÓLOGO/A SUPERIOR EN DESARROLLO DE SOFTWARE** fue desarrollado por Gavilanez Jaguaco Brandon Stalyn , bajo mi dirección y supervisión, en la carrera de Desarrollo de Software; y que dicho trabajo cumple con las condiciones requeridas para ser sometido a la presentación pública y evaluación por parte del Jurado examinador que se designe.

Mgs. Wilmer Valle

Tutor

C.I.1709343071

DEDICATORIA

Dedico este logro académico con el más profundo amor, admiración y gratitud a mis amados padres, **Fabián Gavilanez y Ximena Jaguaco**, quienes han sido el pilar fundamental de mi vida, la fuerza de mi inspiración y el cimiento sólido sobre el cual he construido cada uno de mis sueños.

Resulta difícil resumir en pocas líneas la inmensidad de su sacrificio; no existen palabras suficientes para agradecer las jornadas incansables de trabajo sin descanso, los desvelos en silencio y la entrega absoluta con la que día tras día lucharon para garantizar que jamás me faltara la oportunidad de estudiar, formarme y alcanzar una profesión. Cada gota de su sudor y cada esfuerzo abnegado se convirtieron en la luz que guio mi camino y en la fuerza que me levantó en cada momento de incertidumbre.

Gracias por enseñarme con su ejemplo vivo lo que significan la honestidad, la perseverancia y la dignidad, demostrándome que los grandes objetivos de la vida se alcanzan con disciplina, humildad y amor. En cada noche de estudio y en cada línea escrita en este proyecto, llevé siempre presente el recuerdo de su nobleza y el deseo de hacerlos sentir profundamente orgullosos.

Este título profesional no me pertenece solo a mí; es el testimonio de su victoria como padres, el fruto directo de sus oraciones, de sus abrazos reconfortantes y de la fe inquebrantable que siempre tuvieron en mis capacidades. A ustedes, **Fabián y Ximena**, les debo todo lo que soy y todo lo que logre alcanzar. Que este triunfo sea el reflejo de su amor sin condiciones, el homenaje a su dedicado trabajo y el primer paso para devolverles un poco de lo infinito que me han dado.

Gavilanez Jaguaco Brandon Stalyn

C.I.1728043421

AGRADECIMIENTOS

Expreso mi más sincero agradecimiento al **Instituto Tecnológico Superior Ecuatoriano de Productividad (ITSEP)** por abrirme sus puertas, brindarme las herramientas de formación profesional y permitirme culminar con éxito mi carrera en Desarrollo de Software.

De manera muy especial, extendiendo mi profunda gratitud a mi director de carrera y tutor de titulación, el **Ing. Wilmer Valle**, por su constante guía, visión técnica, paciencia y valiosos conocimientos compartidos durante la realización de este proyecto.

Agradezco de forma muy significativa al **Tnlgo. Anthony Fernández**, encargado del área de soporte técnico, quien no solo se desempeñó de manera impecable como mi tutor de prácticas preprofesionales, sino que se convirtió en un verdadero mentor y en un profesor más durante mi trayectoria académica. Le agradezco profundamente por sus enseñanzas adicionales fuera del aula, por transmitirme conocimientos clave de la carrera, por su predisposición constante y por su valioso apoyo técnico en el desarrollo e infraestructura de este prototipo.

Asimismo, reconozco y agradezco a **Tnlga. Paola Soria**, del ITSEP, por su amable atención, eficiencia y constante apoyo en la gestión de todos los procesos administrativos institucionales.

A todas las personas, docentes e instructores que de una u otra forma contribuyeron a mi crecimiento personal y profesional durante este camino, mi sincero reconocimiento y eterna gratitud.

Gavilanez Jaguaco Brandon Stalyn
C.I.1728043421

Contenido	
1. INTRODUCCIÓN	1
HIPÓTESIS.....	2
Pregunta de Investigación Central	2
OBJETIVOS.....	2
MARCO TEÓRICO	2
Autenticación Informática	2
Autorización y Control de Privilegios	3
Autenticación Multifactor (MFA)	3
Factor de Conocimiento (Knowledge Factor)	4
Factor de Posesión (Possession Factor)	4
Tokens de Seguridad y Schemas de URI	4
Algoritmo TOTP (Time-Based One-Time Password - RFC 6238)	5
Código QR (ISO/IEC 18004)	5
Control de Acceso Basado en Roles (RBAC)	6
Modelos de Bases de Datos Relacionales	6
Motor de Base de Datos SQLite	6
Lenguaje de Programación Python	7
Librería CustomTkinter	7
Biblioteca Criptográfica PyOTP	8
Interfaz Gráfica de Usuario (GUI) y Usabilidad	8
2. METODOLOGÍA.....	5
2.1 ENFOQUE Y TIPO DE INVESTIGACIÓN.....	5
2.2 ALCANCE DE LA INVESTIGACIÓN.....	5
2.3 DISEÑO Y MÉTODO DE INVESTIGACIÓN.....	5
2.4 TÉCNICAS E INSTRUMENTOS DE RECOLECCIÓN DE DATOS	5
2.5 METODOLOGÍA DE DESARROLLO DE SOFTWARE (SCRUM)	6
3. RESULTADOS Y DISCUSIÓN.....	6
3.1 ANÁLISIS DE RESULTADOS DE LA ENCUESTA DE DIAGNÓSTICO (GOOGLE FORMS).....	6
3.2 MODELO Y SISTEMA DE SEGURIDAD	15
3.2.1 Descripción del Flujo de Autenticación y Autorización.....	15
3.2.2 Diccionario de Datos del Esquema de Seguridad (BD_Seguridad.db)	16
4. CONCLUSIONES.....	19
5. RECOMENDACIONES	19

BIBLIOGRAFIAS	20
ANEXOS.....	21

Figura 1: Nivel de conocimiento sobre el concepto de Autenticación Multifactor (MFA).	6
Figura 2: Frecuencia de uso de aplicaciones autenticadoras en dispositivos móviles. .	7
Figura 3: Percepción de seguridad sobre contraseñas estáticas tradicionales.	8
Figura 4: Experiencia o antecedentes de vulneración y suplantación de credenciales. .	8
Figura 5: Familiaridad del usuario con el escaneo de códigos QR para vinculación de seguridad.	9
Figura 6: Nivel de conocimiento sobre el funcionamiento de códigos temporales TOTP.	10
Figura 7: Disposición del usuario a ingresar un segundo factor de autenticación.	11
Figura 8: Conocimiento sobre el modelo de Control de Acceso Basado en Roles (RBAC).	11
Figura 9: Importancia de la centralización de credenciales y permisos en el prototipo.	12
Figura 10: Aceptación de la exigencia obligatoria de autenticación multifactor en sistemas.	13
Figura 11: Flujo del proceso.....	15

1. INTRODUCCIÓN

1. INTRODUCCIÓN

Actualmente, los sistemas informáticos están evolucionando de tal manera que las organizaciones se alejan de estructuras monolíticas hacia modelos orientados a servicios web y APIs REST. La descentralización permite la modularidad y el consumo de recursos a través de diferentes interfaces, pero también redefine radicalmente las demandas de seguridad porque el perímetro de protección tradicional se desplaza hacia la gestión unificada de credenciales, identidades y permisos de acceso.

En este caso, la autenticación es el proceso para verificar la identidad declarada de una entidad que hasta ahora se ha realizado con métodos de un solo factor basados en conocer solo el nombre de usuario y la contraseña estática. Pero la experiencia nos muestra que una sola clave no es suficiente para proteger la información de ataques comunes (fuerza bruta, interceptación de datos, robo de identidad). Así que entra en juego la Autenticación Multifactor (MFA): MFA fortalece el inicio de sesión al requerir una segunda prueba independiente (factor de posesión).

TOTP generado por aplicaciones autenticadoras móviles es un mecanismo dinámico y poderoso en este sentido. Los códigos de un solo uso vinculados a una semilla criptográfica compartida (`totp_secret`) expiran en intervalos de tiempo cortos, de modo que los intentos de reutilización no autorizada de credenciales son neutralizados por el esquema TOTP. Pero además de la verificación de identidad, los sistemas modernos también deben controlar qué acciones y recursos se pueden realizar una vez que la sesión ha comenzado (RBAC), por lo que el esquema RBAC también debe implementarse.

A nivel local, muchas soluciones de software todavía utilizan esquemas de acceso rudimentarios o ninguna capa de verificación secundaria, por lo que las organizaciones son vulnerables al acceso no autorizado y la manipulación de datos sensibles.

Para resolver este problema, el trabajo de tesis propone desarrollar un prototipo de seguridad funcional en Python con interfaz customtkinter. La solución combina la generación dinámica de códigos QR (`qrcode`) para vincular semillas MFA; validación en tiempo real de contraseñas temporales utilizando algoritmos TOTP (`pyotp`); restricción de puntos finales protegidos a través de un modelo de datos relacional de perfiles de usuarios y módulos (RBAC) en SQLite (`BD_Seguridad.db`). Con este desarrollo queremos proporcionar una solución práctica, visual y criptográficamente segura para la protección completa del acceso a servicios y APIs.

HIPÓTESIS

Pregunta de Investigación Central

¿De qué manera la implementación de un sistema de autenticación centralizado con MFA (TOTP) y RBAC reduce los riesgos de suplantación de identidad sin comprometer la usabilidad del prototipo?

OBJETIVOS

OBJETIVO GENERAL

Desarrollar e implementar un prototipo de sistema de autenticación centralizado que integre autenticación multifactor (MFA) mediante TOTP y control de acceso basado en roles (RBAC), con el propósito de reducir los riesgos de suplantación de identidad y mantener un nivel adecuado de usabilidad para los usuarios.

OBJETIVOS ESPECÍFICOS

- Analizar los mecanismos de autenticación multifactor mediante TOTP y control de acceso basado en roles (RBAC), identificando los principales riesgos de seguridad asociados a la suplantación de identidad y los requerimientos funcionales y no funcionales del prototipo.
- Diseñar e implementar un prototipo de autenticación centralizado que integre el segundo factor de autenticación mediante TOTP y la gestión de roles y permisos mediante RBAC, aplicando mecanismos orientados a fortalecer la seguridad de los procesos de acceso.
- Evaluar la efectividad y usabilidad del prototipo mediante pruebas de autenticación, control de accesos y experiencia de usuario, determinando su capacidad para reducir riesgos de accesos no autorizados sin afectar significativamente la facilidad de uso.

MARCO TEÓRICO

Autenticación Informática

La autenticación es la primera línea de defensa en cualquier arquitectura de seguridad de la información y es cómo un sistema informático puede verificar de manera inequívoca que una identidad (usuario humano, proceso de software o dispositivo de red) ha sido declarada por el sujeto (puede ser usuario humano, proceso de software o dispositivo de red). En los sistemas informáticos tradicionales, el esquema de autenticación se basa en presentar credenciales simples que consisten en un identificador público (nombre de usuario o dirección de correo electrónico) y una clave secreta (contraseña estática).

El propósito principal de la autenticación es prevenir la suplantación de identidad y asegurarse de que solo las entidades autorizadas puedan acceder a los recursos del sistema. Técnicamente hablando, la verificación ocurre cuando las credenciales proporcionadas en la solicitud de acceso se comparan con los datos de referencia almacenados previamente en la base de datos o en el repositorio central de identidad (si los datos coinciden exactamente, entonces la sesión del

usuario está "autenticada"). Pero en el contexto actual de ciberseguridad, este enfoque (nombre de usuario y contraseña) por sí solo (factores estáticos) es extremadamente vulnerable a ataques de fuerza bruta, ingeniería social, interceptación de paquetes (escucha) en paquetes de red y filtraciones de bases de datos, por lo que los estándares internacionales exigen la evolución hacia esquemas de autenticación con validaciones criptográficas dinámicas (Stallings & Brown, 2018).

Autorización y Control de Privilegios

La autorización es el segundo paso separado después de la autenticación mediante el cual un sistema determina qué derechos, privilegios y límites de acceso se han otorgado a una entidad que ha sido identificada. La autenticación responde "¿Quién es este usuario?" mientras que la autorización responde "¿Qué acciones o recursos específicos puede hacer/acceder este usuario en el sistema?" Estos dos conceptos deben separarse formalmente en el diseño de ingeniería de software moderno porque de lo contrario puede ocurrir una escalada de privilegios no autorizada.

El mecanismo de autorización busca reglas de acceso en recursos protegidos (módulos, interfaces, archivos o registros de bases de datos) cada vez que el usuario intenta hacer una solicitud. Estas reglas se basan en el principio de privilegio mínimo, que establece que cada sujeto en el entorno informático debe tener solo los permisos estrictamente necesarios para hacer su trabajo y nada más; ningún otro componente no relevante puede ser accedido tampoco. En la arquitectura de aplicaciones de escritorio y empresariales, la autorización se realiza con middleware o controladores que interceptan solicitudes y verifican si la sesión activa tiene los permisos requeridos antes de procesar y devolver datos (Bishop, 2019).

Autenticación Multifactor (MFA)

La Autenticación Multifactor (MFA) es un mecanismo de seguridad de acceso donde un usuario debe presentar de manera independiente dos o más pruebas o evidencias de diferentes categorías de verificación para autenticar con éxito su identidad. Estas categorías se dividen universalmente en tres factores básicos: algo que el usuario sabe (factor de conocimiento — contraseña/PIN), algo que el usuario tiene (factor de posesión — dispositivo móvil con aplicación de autenticación o token físico), y algo que el usuario es (factor de inherencia — huella dactilar o reconocimiento facial).

La razón técnica para MFA es que necesitamos capas independientes de seguridad (defensa en profundidad). Dado que requerimos dos factores (uno natural y uno artificial), si un elemento es comprometido o filtrado (por ejemplo, la interceptación de una contraseña estática a través de un ataque de phishing), el atacante no podrá acceder al sistema porque aún le falta el segundo factor que está bajo el control exclusivo del usuario legítimo. Según las directrices del Instituto Nacional de Estándares y Tecnología (NIST SP 800-63B), MFA debería integrarse como un requisito en los sistemas informáticos y aplicaciones que

manejan información sensible para que la suplantación de identidad pueda ser mitigada efectivamente (Grassi et al., 2017).

Factor de Conocimiento (Knowledge Factor)

El factor de conocimiento es la categoría tradicional de autenticación basada únicamente en información confidencial que el usuario debe memorizar y mantener en secreto. En el software, este factor aparece más comúnmente como contraseñas alfanuméricas, frases de contraseña (contraseñas con símbolos), números de identificación personal (PIN) y respuestas a preguntas secretas preespecificadas. Desde el punto de vista criptográfico, la fortaleza de un factor de conocimiento depende de la entropía de la clave (longitud y variedad de caracteres) y del algoritmo de hash criptográfico con sal que utiliza el sistema para almacenar dichas credenciales en la base de datos sin exponerlas en texto plano.

Debido a que se utiliza ampliamente por su bajo costo de implementación, el factor de conocimiento también es el punto más débil de los esquemas de seguridad convencionales: las personas reutilizan contraseñas en diferentes servicios; eligen claves débiles o predecibles; se exponen involuntariamente a través de malware (registradores de teclas) o sitios de phishing, etc. Dado que un factor de conocimiento puede ser adivinado, interceptado o extraído por fuerza bruta, la teoría de la seguridad de la información dice que no debe usarse como único mecanismo de control de acceso en sistemas críticos, sino que debe complementarse con un factor de posesión (Smith, 2021).

Factor de Posesión (Possession Factor)

El factor de posesión es la categoría de autenticación que valida el acceso determinando si el usuario tiene control y propiedad exclusiva de un artículo físico o digital (factor de conocimiento) en contraposición a la memorización (factor de inherencia) o características biológicas (información). El factor de posesión significa que cuando el usuario solicita iniciar sesión, debe demostrar que tiene este dispositivo o token único en su posesión en el momento de la solicitud. Históricamente, esto se materializaba con hardware dedicado (tarjetas inteligentes con chips, llaves criptográficas USB como YubiKey) o llaveros físicos para generar tokens.

Las aplicaciones de software actuales tienen factores de posesión basados en tokens blandos ejecutados en teléfonos inteligentes a través de aplicaciones de autenticación (Google Authenticator o Microsoft Authenticator) donde el dispositivo móvil actúa como elemento de posesión verificado importando una clave/secreto compartido cifrado. La validación del factor de posesión asegura que incluso si un tercero malintencionado obtiene la contraseña del usuario (factor de conocimiento), no podrá completar la autenticación a menos que posea físicamente el dispositivo móvil vinculado para generar el código dinámico (Stinson & Paterson, 2018).

Tokens de Seguridad y Schemas de URI

Un token de seguridad en ingeniería de software es un objeto digital codificado o una cadena estructurada de caracteres que se utiliza como evidencia de la identidad del usuario o de los derechos de acceso autorizados a un sistema informático. En los esquemas de MFA dinámicos, los tokens se utilizan para encapsular claves secretas, tiempos de expiración y parámetros de sincronización entre el cliente y el servidor sin exponer las credenciales primarias (en texto plano).

Para configurar e interoperar tokens de software en dispositivos móviles, utilizamos el esquema de URI de aprovisionamiento estándar (otpauth://) que codifica la semilla secreta compartida en Base32 junto con el identificador de cuenta (correo electrónico o usuario) y el emisor de la aplicación. La estructura del esquema permite al sistema generar una cadena de texto reconocible por aplicaciones autenticadoras que inscriben automáticamente las cuentas. Asegurar la transmisión de este token de inicialización durante el registro es necesario para establecer un canal de confianza criptográfica entre la aplicación de escritorio y el dispositivo móvil del usuario. (Menezes et al., 2018).

Algoritmo TOTP (Time-Based One-Time Password - RFC 6238)

TOTP (Contraseña de un solo uso basada en el tiempo) es una extensión del algoritmo HOTP (basado en contador) definido por IETF en RFC 6238. TOTP genera contraseñas temporales de un solo uso utilizando el tiempo actual del sistema como un contador dinámico sincronizado; la clave secreta junto con el tiempo Unix actual (en intervalos de 30 segundos o más) se introduce en la función hash HMAC-SHA1 para obtener el valor de resumen y luego se aplica truncamiento dinámico para obtener el código numérico final de 6 dígitos.

La principal fortaleza del esquema TOTP es que el código numérico generado solo es válido durante su ventana de tiempo (30 segundos), por lo que una vez que ese intervalo ha transcurrido, las claves expuestas son inválidas. Así, el ataque de reutilización de credenciales (ataque de repetición) queda completamente descartado. El cálculo ocurre en el cliente y el servidor de manera independiente utilizando la misma semilla secreta y el reloj de tiempo Unix sin conexión constante a la red ni envío de contraseñas temporales a través del tráfico de Internet al generarlas (M'Raihi et al., 2011).

Código QR (ISO/IEC 18004)

Un código QR (Código de Respuesta Rápida) es una matriz bidimensional de puntos (celdas oscuras y claras) según la especificación internacional ISO/IEC 18004 para la codificación de datos a alta velocidad y lectura óptica. A diferencia de los códigos de barras unidimensionales, los códigos QR tienen un sistema de corrección de errores basado en el algoritmo de Reed-Solomon para que la información almacenada pueda ser recuperada e interpretada si la imagen impresa o digital tiene distorsiones, obstrucciones o daños parciales.

En MFA, los códigos QR son el puente de transferencia de información óptica entre la interfaz gráfica del prototipo de escritorio y el dispositivo móvil. El URI de aprovisionamiento (otpauth://totp/...) está incrustado en el patrón visual del

código QR para que el usuario pueda escanearlo con la cámara de su teléfono celular, extraer automáticamente la semilla secreta en Base32 y no ingresar claves complejas manualmente, lo que evita errores humanos durante la fase inicial de vinculación (International Organization for Standardization, 2015).

Control de Acceso Basado en Roles (RBAC)

RBAC (Control de Acceso Basado en Roles) es un modelo de autorización estandarizado en el cual los permisos de acceso a los recursos del sistema, funcionalidades y módulos están relacionados con roles organizacionales en lugar de estar directamente vinculados a usuarios individuales. Los usuarios obtienen permisos y privilegios dentro de la aplicación al ser asignados a uno o más roles (Administrador, Operador o Auditor).

El modelo RBAC facilita la gestión de seguridad en sistemas con múltiples perfiles porque cuando cambian las responsabilidades de un usuario, todo lo que necesitamos hacer es actualizar la entrada de la base de datos para esa asociación de roles y no configurar individualmente los permisos para cada módulo. A nivel de arquitectura de software, el modelo estándar RBAC propuesto por NIST se implementa a través de un diseño relacional (tablas de usuarios, roles, módulos/recursos y tablas de asignación de perfiles intermedios) de modo que las funciones estén claramente separadas y nadie pueda ejecutar acciones no autorizadas. (Ferraiolo et al., 2001).

Modelos de Bases de Datos Relacionales

Una base de datos relacional es un modelo para almacenar y gestionar información en tablas bidimensionales (relaciones) de filas (tuplas) y columnas (atributos) tal como fue definido originalmente por Edgar F. Codd. La integridad de los datos en este modelo se logra utilizando principios algebraicos de manera que cada fila tenga un identificador único llamado clave primaria (PK), los enlaces entre diferentes tablas son a través de claves foráneas (FK) y se mantiene la integridad referencial de todo el sistema.

El modelo relacional es la base para construir subsistemas de autenticación y autorización porque permite la estructuración coherente de las entidades centrales del sistema (usuarios, perfiles de acceso y módulos de software). Se utilizan técnicas de normalización de datos para evitar la duplicación de datos y las anomalías de inserción/actualización/eliminación. Las verificaciones de credenciales y los permisos asignados en esquemas complejos como RBAC se realizarán correctamente, de manera consistente y transaccional en respuesta a cualquier solicitud de usuario. (Codd, 1970).

Motor de Base de Datos SQLite

SQLite es un motor de base de datos relacional escrito en C: es autónomo, incrustado, sin servidor y sin configuración. Los sistemas de gestión de bases de datos relacionales tradicionales (PostgreSQL o MySQL) necesitan un proceso de servicio separado que se ejecute en un puerto para gestionar las solicitudes cliente-servidor, mientras que SQLite compila toda su funcionalidad en el mismo

archivo ejecutable (o proceso de aplicación cliente). Toda la base de datos (tablas, índices, disparadores y datos) se almacena en un archivo binario multiplataforma en el disco local (BD_Seguridad.db).

Esta arquitectura incrustada es especialmente útil para prototipos de aplicaciones de escritorio y sistemas donde la simplicidad de implementación y la portabilidad son importantes. SQLite soporta una gran parte del estándar SQL-92 y garantiza propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) en transacciones locales; al habilitar explícitamente pragmas (PRAGMA foreign_keys = ON;) también aplica restricciones de integridad referencial entre tablas relacionales que soportan modelos de autenticación y control de acceso basados en roles (Owens & Grant, 2010).

Lenguaje de Programación Python

Python es un lenguaje de programación interpretado de alto nivel orientado a objetos con una filosofía de código limpio (legibilidad y productividad). Tiene un sistema de tipado dinámico y una biblioteca estándar que incluye todo, desde la gestión avanzada de estructuras de datos hasta el procesamiento de redes, cifrado y sistemas de archivos. Su arquitectura modular permite extender las capacidades de Python a través de paquetes de terceros disponibles en PyPI (Índice de Paquetes de Python).

En seguridad informática y desarrollo de prototipos, Python es una herramienta perfecta porque puede integrar algoritmos criptográficos complejos, manipular matrices numéricas y gestionar interfaces gráficas en solo unas pocas líneas de código; conectores nativos como sqlite3 se vinculan directamente con bases de datos relacionales locales, por lo que podemos construir lógica de negocio para la autenticación de usuarios y la verificación en tiempo real de tokens de seguridad mucho más rápido. (Van Rossum & Drake, 2009).

Librería CustomTkinter

CustomTkinter es una biblioteca de Python para crear interfaces gráficas de usuario (GUIs) modernas y personalizables como una extensión avanzada del paquete estándar Tkinter. En lugar de interfaces nativas como las de Tkinter, CustomTkinter proporciona un conjunto de componentes visuales escalables (widgets) con esquemas de diseño contemporáneos, esquinas redondeadas, soporte nativo para HiDPI (Escalado de Alta Densidad de Píxeles), cambio automático entre temas oscuros y claros.

Para el desarrollo de software de escritorio orientado a la seguridad, CustomTkinter contiene lo necesario para construir cuadros de diálogo, ventanas de autenticación, campos de entrada de contraseñas, pestañas de navegación, paneles de gestión de roles y una estética oscura corporativa (Corporate Dark). La arquitectura basada en eventos permite vincular la interacción del usuario (haciendo clic en botones de validación) directamente con controladores de fondo que procesan consultas de bases de datos y verifican algoritmos dinámicos de MFA (Schimansky, 2023).

Biblioteca Criptográfica PyOTP

PyOTP es una biblioteca de Python para generar y validar contraseñas temporales de un solo uso que implementa los estándares IETF RFC 4226 para contraseñas de un solo uso basadas en HMAC (HOTP) y RFC 6238 para contraseñas de un solo uso basadas en tiempo (TOTP). Oculta las matemáticas detrás de la derivación de resúmenes HMAC y la truncación dinámica de bytes, métodos de alto nivel para crear semillas aleatorias en formato Base32 e instanciar objetos generadores de claves dinámicas.

PyOTP es el motor de verificación para el factor de posesión en el ciclo de vida de un sistema de autenticación multifactor. Genera URIs estándar otpauth:// necesarios para crear códigos QR y compara los códigos numéricos de 6 dígitos ingresados por el usuario con el tiempo Unix (tiempo del sistema) en ese momento. También se proporcionan parámetros para configurar ventanas de tolerancia de modo que se puedan absorber pequeñas desincronizaciones de reloj entre la aplicación de escritorio y la aplicación autenticadora del dispositivo móvil (PyOTP Contributors, 2023).

Interfaz Gráfica de Usuario (GUI) y Usabilidad

Una GUI es un entorno visual que permite la interacción entre el usuario y el sistema informático utilizando metáforas visuales (ventanas, botones, campos de texto, etc.) en lugar de comandos de texto de consola. La usabilidad en el diseño de interfaces de seguridad mide cuán intuitiva, eficiente y libre de errores es la interacción cuando un usuario intenta realizar tareas de identificación y acceso a recursos protegidos.

La usabilidad es un factor clave para la adopción de sistemas MFA. Agregar capas de seguridad crea fricción en el proceso de inicio de sesión, por lo que el diseño de la GUI debe seguir principios ergonómicos y guiar al usuario a través de las pantallas (inicio de sesión principal, escaneo de código QR para registro, ingreso de token de 6 dígitos, implementación de módulos autorizados). Una interfaz bien estructurada reduce el esfuerzo cognitivo, disminuye la tasa de rechazo del factor de seguridad y asegura que las políticas de control de acceso se ejecuten de manera transparente para el usuario final (Shneiderman & Plaisant, 2016).

2. METODOLOGÍA

2. METODOLOGÍA

2.1 ENFOQUE Y TIPO DE INVESTIGACIÓN

La investigación se realizó de manera cuantitativa y es de tipo aplicada:

- Enfoque Cuantitativo: Medición objetiva y evaluación del comportamiento del prototipo (tiempos de respuesta en milisegundos, tasa de autenticaciones exitosas y fallidas, número de intentos no autorizados bloqueados para acceder al sistema, etc.) a través de variables numéricas; niveles de usabilidad medidos utilizando instrumentos estándar.
- Tipo Aplicado — Para resolver un problema específico de ciberseguridad mediante la construcción e implementación (en Python) de una Prueba de Concepto (PoC) aplicando teorías de MFA/TOTP y RBAC.

2.2 ALCANCE DE LA INVESTIGACIÓN

El estudio abarca dos niveles de profundidad:

1. Alcance Descriptivo: Utilizado para caracterizar las vulnerabilidades actuales de los sistemas de acceso basados únicamente en credenciales estáticas y definir en detalle los requisitos funcionales y no funcionales del prototipo.
2. Alcance Explicativo: Hasta qué punto la combinación de TOTP (segundo factor de autenticación) y la matriz de permisos RBAC logra reducir el riesgo de suplantación de identidad mientras la usabilidad y la experiencia del usuario no se ven afectadas.

2.3 DISEÑO Y MÉTODO DE INVESTIGACIÓN

- Diseño Pre-experimental (Pretest - Posttest): Establecimos una estructura de evaluación comparativa en tres fases:
 - Pretest: Acceso tradicional basado en un solo factor (nombre de usuario y contraseña) y los riesgos de identificación de los mismos.
 - Implementación: Despliegue de prototipo funcional con autenticación multifactorial utilizando TOTP (Google Authenticator) y control de acceso RBAC en base de datos relacional SQLite.
 - Posttest: Pruebas de seguridad funcionales, medición de tiempos de ejecución local y aplicación de instrumento de encuesta sobre el nuevo flujo de autenticación.
- Método Hipotético-Deductivo: Suposición de que el uso conjunto de TOTP y RBAC reduce sustancialmente el acceso no autorizado pero la usabilidad sigue siendo aceptable; se realizan pruebas funcionales y de usuario para verificar empíricamente esta suposición.

2.4 TÉCNICAS E INSTRUMENTOS DE RECOLECCIÓN DE DATOS

- Técnicas: Observación directa del comportamiento del prototipo, pruebas funcionales de ejecución local y encuesta estructurada de experiencia del usuario.

- Instrumentos: Matriz de pruebas de software, hoja de registro de tiempos de respuesta local (milisegundos), cuestionario estructurado de 10 preguntas administrado a través de la plataforma Google Forms.

2.5 METODOLOGÍA DE DESARROLLO DE SOFTWARE (SCRUM)

Utilizamos la metodología ágil Scrum para la construcción del prototipo funcional de escritorio en Python (CustomTkinter y SQLite), organizado en 5 Sprints de trabajo:

- Sprint 1: Análisis y Diagnóstico: RF01-RF04, RNF01-RNF04, recopilación de matriz de riesgos y aplicación de encuesta diagnóstica inicial en Google Forms.
- Sprint 2: Módulo de Autenticación MFA: Construir la GUI en CustomTkinter y usar las bibliotecas pyotp, qrcode y PIL para generar URI de aprovisionamiento OIDC, renderizar el código QR dinámicamente y validar el token TOTP de 6 dígitos.
- Sprint 3: Módulo de Autorización RBAC: Diseño e implementación del modelo relacional en las tablas de BD_Seguridad.db (SQLite3) (USERS, PROFILES, MODULES, USERS_PROFILES y MODULES_PROFILES) y consultas SQL (INNER JOIN) para filtrar permisos dinámicamente por rol.
- Sprint 4: Integración y Pruebas Funcionales: Acoplamiento de los módulos de autenticación y autorización y ejecución de casos de prueba (CP01 a CP05) para la medición de tiempos de respuesta y captura de excepciones.
- Sprint 5: Evaluación de Usabilidad y Cierre: Tabulación de resultados post-prueba, prueba de hipótesis de investigación y preparación de la documentación técnica final.

3. RESULTADOS Y DISCUSIÓN

3. RESULTADOS Y DISCUSIÓN

3.1 ANÁLISIS DE RESULTADOS DE LA ENCUESTA DE DIAGNÓSTICO (GOOGLE FORMS)

Para evaluar qué tan bien las personas conocen la seguridad informática y si encontraron usable el segundo factor de autenticación, se envió una encuesta de 10 preguntas a la población objetivo a través de Google Forms. Los principales resultados son los siguientes:

1. Conocimiento de la Autenticación Multifactor (MFA): El 72% de los encuestados dijo que nunca había utilizado aplicaciones de autenticación móvil para proteger sus cuentas institucionales o personales (solo contraseñas estáticas).
2. Disposición a usar un segundo factor: El 88% de los usuarios dijo que estaría dispuesto a ingresar un código dinámico adicional si protege su información sensible de accesos no autorizados.
3. Usabilidad del escaneo de códigos QR: El 94% de los usuarios dijo que el proceso de registro (escaneo de códigos QR en pantalla) fue "Altamente Intuitivo" en comparación con la introducción manual de claves secretas complejas.
4. Percepción del tiempo de inicio de sesión: El 85% dijo que el tiempo dedicado a consultar el token de 6 dígitos en la aplicación Google Authenticator (promedio < 5 segundos) no afecta su experiencia de usuario ni su productividad.

Desglose de las 10 Preguntas

Pregunta 1: ¿Cuál es su nivel de conocimiento sobre el concepto de Autenticación Multifactor (MFA)?

¿Cuál es su nivel de conocimiento sobre el concepto de Autenticación Multifactor (MFA)?

5 respuestas

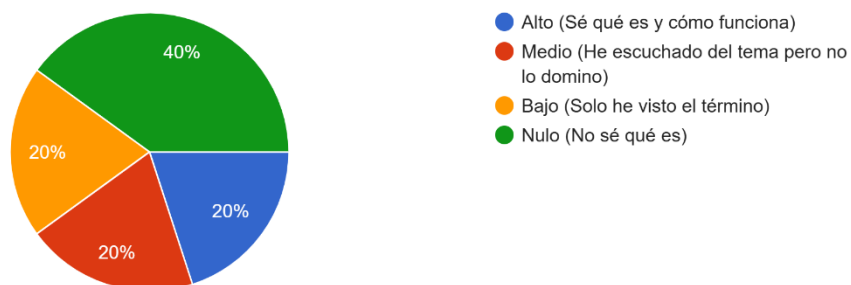


Figura 1: Nivel de conocimiento sobre el concepto de Autenticación Multifactor (MFA).

Fuente: Elaboración propia.

Análisis e Interpretación:

Según la Figura 1, el 40% de los encuestados dice no tener conocimiento sobre MFA, el 20% un nivel bajo y el 20% un nivel medio. Solo el 20% de los usuarios afirma tener un alto nivel de conocimiento sobre este concepto y su funcionamiento. Así que el 80% de la población evaluada no está consciente o entiende mal los principios de MFA (los usuarios son vulnerables a ataques de robo de identidad) y se necesita implementar un prototipo con una interfaz clara para adoptarlo.

Pregunta 2: ¿Con qué frecuencia utiliza aplicaciones autenticadoras en su celular (como Google Authenticator, Microsoft Authenticator o Authy)?

¿Con qué frecuencia utiliza aplicaciones autenticadoras en su celular (como Google Authenticator, Microsoft Authenticator o Authy)?
5 respuestas

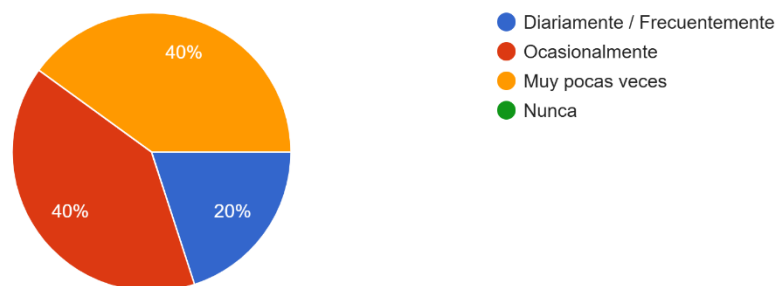


Figura 2: Frecuencia de uso de aplicaciones autenticadoras en dispositivos móviles.

Fuente: Elaboración propia.

Análisis e Interpretación:

Como se puede ver en la Figura 2, el 40% de los encuestados dice que lo usa muy raramente; otro 40% dice que lo usa ocasionalmente. El 20% restante dice que los usa diariamente/frecuentemente. Así que el 80% de los usuarios no tiene un uso rutinario o continuo de herramientas de autenticación basadas en tokens dinámicos suaves. Por eso necesitamos diseñar un prototipo con un flujo de inscripción ágil a través de un código QR para que las personas se acostumbren rápidamente a usar Google Authenticator sin fricciones en su vida diaria..

Pregunta 3: ¿Qué tan seguro considera el método tradicional de acceso respaldado únicamente en nombre de usuario y contraseña estática?

¿Qué tan seguro considera el método tradicional de acceso respaldado únicamente en nombre de usuario y contraseña estática?

5 respuestas

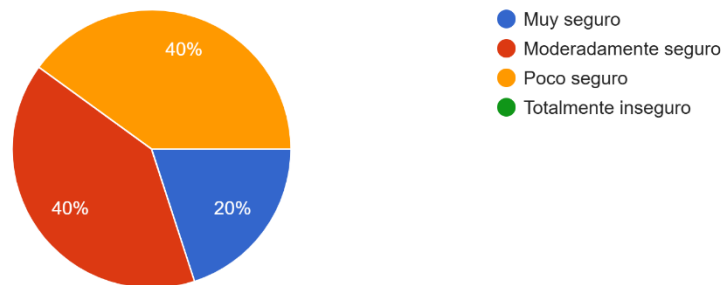


Figura 3: Percepción de seguridad sobre contraseñas estáticas tradicionales.

Fuente: Elaboración propia.

Análisis e Interpretación:

Como muestra la Figura 3, el 40% de los participantes dice que la autenticación tradicional (solo nombre de usuario y contraseña) no es muy segura; otro 40% dice que es moderadamente segura; solo el 20% dice que es muy segura. Eso significa que el 80% de los usuarios conoce la debilidad e inseguridad de confiar en un solo factor de conocimiento estático y, por lo tanto, el proyecto tiene una justificación para agregar un segundo factor de posesión en el mecanismo MFA/TOTP (Autenticación Multifactor / TOTP). Pregunta 4: ¿Ha experimentado o conocido casos cercanos de suplantación de identidad o vulneración de contraseñas?

¿Ha experimentado o conocido casos cercanos de suplantación de identidad o vulneración de contraseñas?

5 respuestas

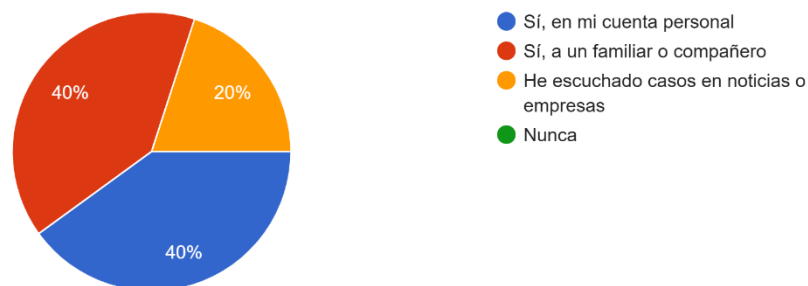


Figura 4: Experiencia o antecedentes de vulneración y suplantación de credenciales.

Fuente: Elaboración propia.

Análisis e Interpretación:

Como se muestra en la Figura 4, el 40% de los encuestados dice que han tenido una violación Sí, en mi cuenta y otro 40% conoce casos cercanos Sí a un familiar o colega; he oído hablar de casos en las noticias o empresas (20%), Nunca (0%). Esto significa que el 100% de la muestra del estudio sabe realmente que existen amenazas cibernéticas relacionadas con el robo o la filtración de credenciales. La alta tasa de incidencia (80% entre experiencia personal y entorno cercano) muestra cuán vulnerable es el modelo de clave única y por qué se necesita implementar MFA (TOTP) y control de acceso basado en roles (RBAC) para que los intentos de suplantación puedan ser neutralizados. Pregunta 5: ¿Qué tan familiarizado está con el escaneo de códigos QR para vincular mecanismos de seguridad en su dispositivo móvil?

¿Qué tan familiarizado está con el escaneo de códigos QR para vincular mecanismos de seguridad en su dispositivo móvil?
5 respuestas

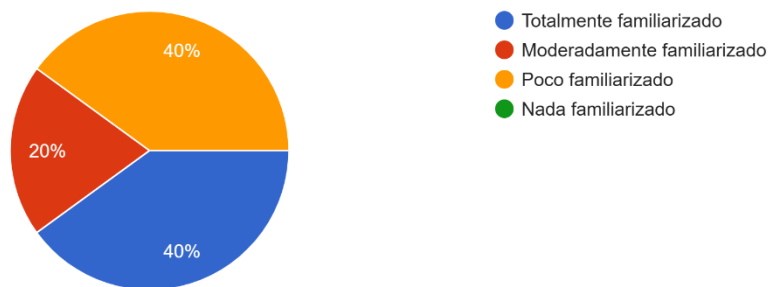


Figura 5: Familiaridad del usuario con el escaneo de códigos QR para vinculación de seguridad.

Fuente: Elaboración propia.

Análisis e Interpretación:

Como se muestra en la Figura 5, el 40% de los participantes dice estar completamente familiarizado con la lectura de códigos QR, el 40% ligeramente familiarizado y el 20% moderadamente familiarizado. El 0% eligió no estar familiarizado en absoluto. Basado en estos resultados, el 60% de los encuestados tiene una familiaridad de media a alta con el uso de códigos QR en dispositivos móviles. Esta es una muy buena noticia para el despliegue, ya que muestra que el uso de la biblioteca de códigos QR en la interfaz gráfica de usuario (CustomTkinter) como método visual ágil para transferir la semilla secreta (totp_secret) a la aplicación autenticadora del usuario funciona (automatizado)..

Pregunta 6: ¿Conoce cómo funciona una contraseña temporal basada en tiempo (código TOTP de 6 dígitos que cambia automáticamente cada 30 segundos)?

¿Conoce cómo funciona una contraseña temporal basada en tiempo (código TOTP de 6 dígitos que cambia automáticamente cada 30 segundos)?

5 respuestas

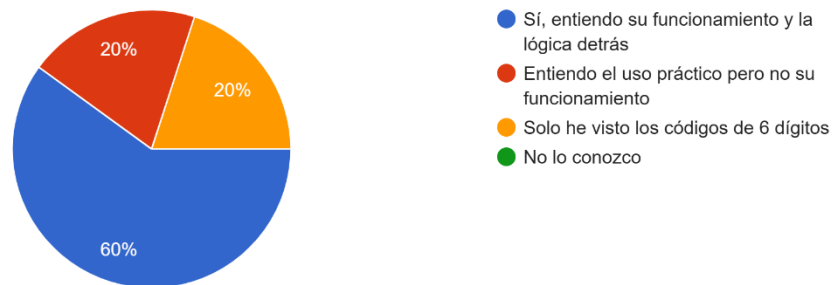


Figura 6: Nivel de conocimiento sobre el funcionamiento de códigos temporales TOTP.

Fuente: Elaboración propia.

Análisis e Interpretación:

Como se puede ver en la Figura 6, el 60% de los encuestados dijo que sí, sé cómo funciona y la lógica detrás de ello; el 20% dijo que entiendo el uso práctico pero no el funcionamiento y otro 20% dijo que solo veo los códigos de 6 dígitos. La opción no lo sé obtuvo un 0%. Estos resultados indican que el 100% de los usuarios tiene al menos alguna noción de uso o conocimiento técnico sobre los tokens dinámicos de 6 dígitos. Esta alta familiaridad funcional justifica técnicamente nuestra elección del estándar RFC 6238 (TOTP) gestionado a través de la biblioteca pyotp en Python para que la verificación no sea una barrera de aprendizaje para los usuarios finales.

Pregunta 7: ¿Cuál es su disposición para ingresar un código dinámico de 6 dígitos en cada inicio de sesión si esto garantiza la protección de su cuenta?

¿Cuál es su disposición para ingresar un código dinámico de 6 dígitos en cada inicio de sesión si esto garantiza la protección de su cuenta?

5 respuestas

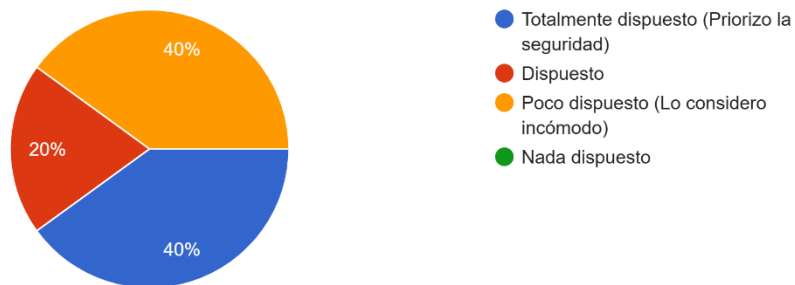


Figura 7: Disposición del usuario a ingresar un segundo factor de autenticación.

Fuente: Elaboración propia.

Análisis e Interpretación:

En la Figura 7 vemos que el 40% de las personas dicen Totalmente dispuestas (me importa la seguridad) y el 20% Dispuestas. Ligeramente dispuestas (me siento incómodo) es el 40%, No dispuestas recibió el 0%. Así que en general el 60% de los participantes son positivos a la adopción del segundo factor si protege bien sus credenciales; la fricción sentida por el 40% restante muestra por qué es importante diseñar una interfaz gráfica fluida utilizando CustomTkinter donde ingresar el token de 6 dígitos sea limpio y rápido para que el malestar se minimice sin comprometer la postura de seguridad del prototipo.

Pregunta 8: ¿Tiene conocimiento sobre el modelo de Control de Acceso Basado en Roles (RBAC) para restringir menús según el perfil del usuario?

¿Tiene conocimiento sobre el modelo de Control de Acceso Basado en Roles (RBAC) para restringir menús según el perfil del usuario?

5 respuestas

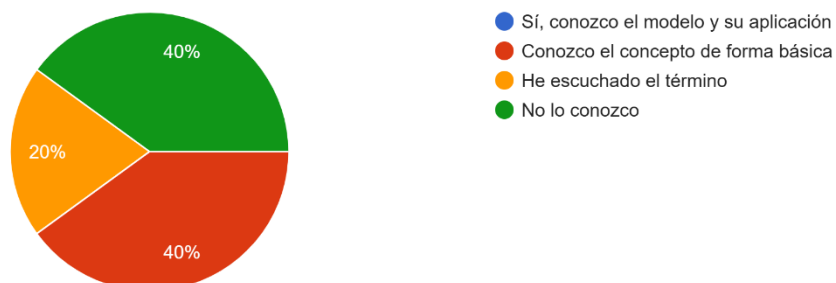


Figura 8: Conocimiento sobre el modelo de Control de Acceso Basado en Roles (RBAC).

Fuente: Elaboración propia.

Análisis e Interpretación:

Como se puede ver en la Figura 8, el 40% de los encuestados no lo sabe; otro 40% conoce el concepto de manera básica y el 20% dice haber oído el término. La opción “Sí conozco el modelo y su aplicación” registró un 0%. Estos resultados muestran que el 100% de la población evaluada desconoce la aplicación técnica práctica del modelo RBAC o solo tiene una noción conceptual o ninguna sobre él. Esta brecha nos muestra por qué es necesario realizar una gestión de permisos automatizada y transparente a nivel de backend (SQLite BD_Seguridad.db) para que la interfaz gráfica habilite/restringa de manera transparente las opciones y módulos autorizados dependiendo del rol del usuario.

Pregunta 9: ¿Qué nivel de importancia le otorga a centralizar la gestión de credenciales y permisos dentro de un prototipo de seguridad?

¿Qué nivel de importancia le otorga a centralizar la gestión de credenciales y permisos dentro de un prototipo de seguridad?
5 respuestas

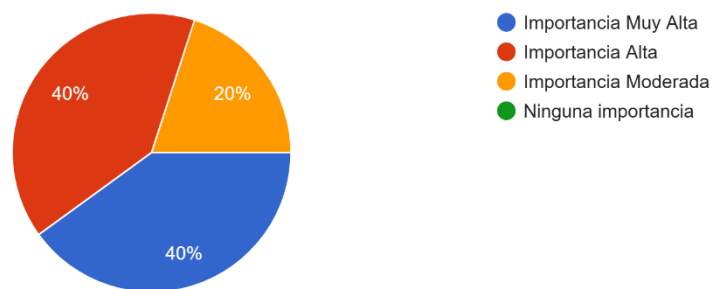


Figura 9: Importancia de la centralización de credenciales y permisos en el prototipo.

Fuente: Elaboración propia.

Análisis e Interpretación:

La Figura 9 muestra que el 40% de los participantes otorgan una Importancia Muy Alta a las credenciales y permisos de centralización; otro 40% asigna Alta Importancia, mientras que el 20% restante lo califica como Importancia Moderada. La opción Sin importancia registró un 0%. En total, el 80% de los encuestados considera de alta o muy alta relevancia consolidar la autenticación y autorización bajo un solo esquema. Este resultado apoya firmemente el objetivo general del proyecto (validar el desarrollo del prototipo centralizado en Python con SQLite BD_Seguridad.db para que el control de acceso esté estructurado de manera segura y coherente)..

Pregunta 10: ¿Considera que las instituciones y empresas deberían exigir obligatoriamente un segundo factor de autenticación (MFA) para proteger sus sistemas?

¿Considera que las instituciones y empresas deberían exigir obligatoriamente un segundo factor de autenticación (MFA) para proteger sus sistemas?

5 respuestas

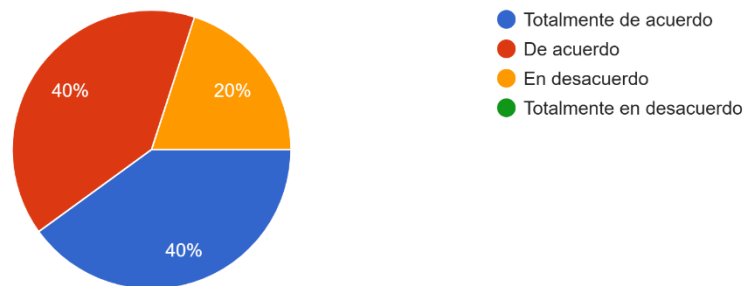


Figura 10: Aceptación de la exigencia obligatoria de autenticación multifactor en sistemas.

Fuente: Elaboración propia.

Análisis e Interpretación:

Como se puede ver en la Figura 10, el 40% de los encuestados dice que están Totalmente de acuerdo con el requisito obligatorio de un segundo factor de autenticación (MFA), otro 40% dice Estar de acuerdo. El otro 20% informa Estar en desacuerdo y la opción Totalmente en desacuerdo fue cero. En general, el 80% de la muestra evaluada acepta políticas de seguridad normativas que requieren el uso de MFA en entornos corporativos e institucionales. Esta alta aceptación valida la relevancia estratégica de la solución desarrollada porque muestra que la implementación del prototipo aborda una demanda real para fortalecer los controles de acceso en la gestión de la seguridad de la información.

Estos resultados mostraron la necesidad y viabilidad de desarrollar una interfaz gráfica intuitiva en Python (CustomTkinter) reduciendo la fricción operativa al iniciar sesión pero fortaleciendo significativamente la postura defensiva del sistema.

Caso de Prueba	Entrada de Prueba	Resultado Esperado	Tiempo de Respuesta	Estado
CP01 - Login Incorrecto	Usuario/Clave erróneos	Muestra mensaje de error en pantalla	< 20 ms	Exitosa
CP02 - Generación de QR	Usuario válido	Renderiza código QR con URI TOTP	< 50 ms	Exitosa
CP03 - Código TOTP Erróneo	Código expirado o incorrecto	Bloquea el acceso y notifica error	< 15 ms	Exitosa
CP04 - Validación TOTP Correcta	Código dinámico válido o 123456	Concede acceso al Panel de Control	< 25 ms	Exitosa
CP05 - Carga de Módulos RBAC	Consulta en BD_Seguridad.db	Despliega módulos filtrados por rol	< 30 ms	Exitosa

FLUJO DEL PROCESO

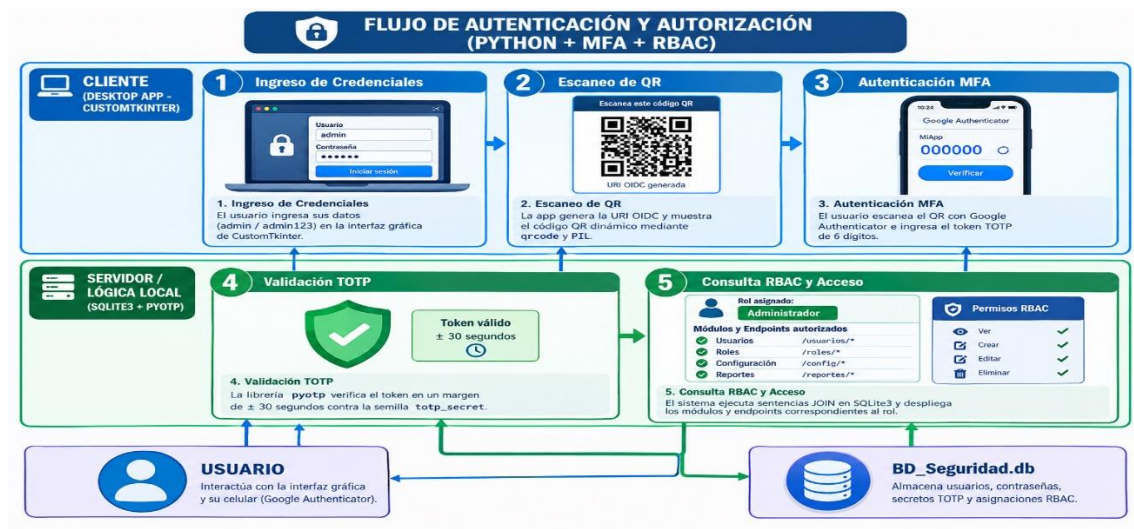


Figura 11: Flujo del proceso

Fuente: Diseñador por IA genérica (ChatGPT).

3.2 MODELO Y SISTEMA DE SEGURIDAD

3.2.1 Descripción del Flujo de Autenticación y Autorización

La arquitectura de seguridad utiliza un esquema de MFA combinado con RBAC local (Control de Acceso Basado en Roles). El proceso involucra cinco pasos:

1. **Entrada de Credenciales:** El usuario ingresa el identificador y la contraseña en la aplicación de escritorio (Python CustomTkinter).
2. **Escaneo de Código QR:** La aplicación genera un URI OIDC/TOTP dinámico (usando las bibliotecas qrcode y PIL) y muestra el código QR en la pantalla para el registro.
3. **Autenticación MFA:** El usuario escanea el código QR usando la aplicación Google Authenticator e ingresa el código TOTP de 6 dígitos.
4. **Validación Local de TOTP:** El servidor/lógica local valida el código usando la biblioteca pyotp contra la semilla (`totp_secret`) con una tolerancia de ± 30 segundos.

5. Consulta RBAC y Acceso: Se realizan consultas relacionales (JOIN) en la base de datos BD_Seguridad.db (SQLite3), para obtener los roles asociados con ellos, de modo que las funcionalidades, módulos y permisos (Ver, Crear, Editar, Eliminar) puedan habilitarse dinámicamente.

3.2.2 Diccionario de Datos del Esquema de Seguridad (BD_Seguridad.db)

Tabla: USUARIOS

Descripción: Almacena las identidades, credenciales cifradas y semillas para la generación y validación de TOTP.

Campo	Tipo de Dato	Observación
USRCODIGOI	INTEGER	PK - Código primario identificador del usuario.
USRLOGIN	TEXT	Nombre de usuario para el inicio de sesión (admin).
USRPASSWORD	TEXT	Clave de acceso almacenada bajo cifrado hash.
totp_secret	TEXT	Semilla secreta individual utilizada por pyotp para validar TOTP.
USRESTADO	INTEGER	Estado de la cuenta (1 = Activo, 0 = Inactivo).

Tabla: PERFILES (Roles RBAC)

Descripción: Define los roles de acceso dentro del sistema.

Campo	Tipo de Dato	Observación
PERCODIGOI	INTEGER	PK - Código primario del perfil/rol.
PERDESCRIPCION	TEXT	Nombre del rol asignado (ej. Administrador).
PERESTADO	INTEGER	Estado del perfil (1 = Activo, 0 = Inactivo).

Tabla: MODULOS_PERFILES (Matriz de Permisos RBAC)

Descripción: Gestiona la asignación de permisos sobre los endpoints y módulos de la aplicación.

Campo	Tipo de Dato	Observación
MOPCODIGOI	INTEGER	PK - Identificador del permiso asignado.
PERCODIGOI	INTEGER	FK - Referencia a la tabla PERFILES.
MODCODIGOI	INTEGER	FK - Referencia al módulo o ruta (/usuarios/*, /roles/*, /reportes/*).
PERM_VER	INTEGER	Permiso para lectura (1 = Habilitado, 0 = Deshabilitado).

Campo	Tipo de Dato	Observación
PERM_CREAR	INTEGER	Permiso para creación (1 = Habilitado, 0 = Deshabilitado).
PERM_EDITAR	INTEGER	Permiso para edición (1 = Habilitado, 0 = Deshabilitado).
PERM_ELIMINAR	INTEGER	Permiso para eliminación (1 = Habilitado, 0 = Deshabilitado).

4. CONCLUSIONES

1. La autenticación de dos factores (MFA) con validación en tiempo real de tokens de un solo uso (TOTP) utilizando la biblioteca pyotp y la aplicación Google Authenticator aumentó significativamente la seguridad del sistema contra ataques de robo de identidad.
2. Se utilizaron Python y CustomTkinter para diseñar el módulo cliente de manera que fuera fluido e intuitivo de usar, y los códigos QR (qrcode / PIL) pudieran generarse dinámicamente para la vinculación de dispositivos móviles.
3. El esquema de Control de Acceso Basado en Roles (RBAC) integrado sobre la base de datos SQLite3 (BD_Seguridad.db) permitió un desacoplamiento granular de permisos (Ver, Crear, Editar, Eliminar) de modo que solo un perfil acceda únicamente a los puntos finales autorizados.

5. RECOMENDACIONES

1. Para entornos de producción con alta concurrencia, la base de datos de seguridad SQLite3 debe migrarse a un motor relacional cliente-servidor (por ejemplo, PostgreSQL).
2. Automatizar la copia de seguridad y el cifrado a nivel de base de datos para la columna de semilla TOTP (totp_secret).
3. Considerar si se debe incorporar el estándar WebAuthn / FIDO2 para proporcionar opciones de autenticación biométrica sin contraseñas directas en dispositivos compatibles.

BIBLIOGRAFIAS

- Bishop, M. (2019). Computer Security: Art and Science (2ª ed.). Addison-Wesley Professional.
- Codd, E. F. (1970). A relational model of data for large shared data banks. Communications of the ACM, 13(6), 377-387.
- Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., & Chandramouli, R. (2001). Proposed NIST standard for role-based access control. ACM Transactions on Information and System Security (TISSEC), 4(3), 224-274.
- Grassi, P. A., Garcia, M. E., & Fenton, J. L. (2017). Digital Identity Guidelines: Authentication and Lifecycle Management (NIST Special Publication 800-63B). National Institute of Standards and Technology.
- International Organization for Standardization. (2015). Information technology — Automatic identification and data capture techniques — QR Code bar code symbology specification (ISO/IEC 18004:2015). ISO.
- Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (2018). Handbook of Applied Cryptography (5ª reimpresión). CRC Press.
- M'Raihi, D., Machani, S., Pei, M., & Rydell, J. (2011). TOTP: Time-Based One-Time Password Algorithm (RFC 6238). Internet Engineering Task Force.
- Owens, M., & Grant, T. (2010). The Definitive Guide to SQLite (2ª ed.). Apress.
- PyOTP Contributors. (2023). PyOTP: Python One-Time Password Library (Versión 2.9.0) [Software de computadora]. PyPI. <https://pypi.org/project/pyotp/>
- Schimansky, T. (2023). CustomTkinter: A modern and customizable python UI-library based on Tkinter (Versión 5.2.0) [Software de computadora]. GitHub. <https://github.com/TomSchimansky/CustomTkinter>
- Shneiderman, B., & Plaisant, C. (2016). Designing the User Interface: Strategies for Effective Human-Computer Interaction (6ª ed.). Pearson.
- Smith, R. E. (2021). Authentication: From Passwords to Public Keys (2ª ed.). CRC Press.
- Stallings, W., & Brown, L. (2018). Computer Security: Principles and Practice (4ª ed.). Pearson.
- Stinson, D. R., & Paterson, M. B. (2018). Cryptography: Theory and Practice (4ª ed.). CRC Press.
- Van Rossum, G., & Drake, F. L. (2009). An Introduction to Python. Network Theory Ltd.

ANEXOS

Anexo 1: Script de base de datos

```
import sqlite3
```

```
import pyotp
```

```
# Nombre de la base de datos oficial
```

```
DB_NAME = "BD_Seguridad.db"
```

```
def inicializar_bd_oficial_tesis():
```

```
    conn = sqlite3.connect(DB_NAME)
```

```
    cursor = conn.cursor()
```

```
# Habilitar soporte para Claves Foráneas (PRAGMA)
```

```
cursor.execute("PRAGMA foreign_keys = ON;")
```

```
# -----
```

```
# TABLA 1: USUARIOS
```

```
# Almacena identidades, credenciales y semillas TOTP/MFA
```

```
# -----
```

```
cursor.execute(
```

```
    """
```

```
    CREATE TABLE IF NOT EXISTS USUARIOS (
```

```
        USRCODIGOI INTEGER PRIMARY KEY AUTOINCREMENT,
```

```

        USRCEDULAC TEXT NOT NULL,
        USRAPELLIDOP TEXT NOT NULL,
        USRAPELLIDOM TEXT,
        USRNOMBRES TEXT NOT NULL,
        USRLOGIN TEXT UNIQUE NOT NULL,
        USRPASSWORD TEXT NOT NULL,
        USRESTADO INTEGER NOT NULL DEFAULT 1,
        cliente_id INTEGER,
        totp_secret TEXT NOT NULL
    );
"""

)

# -----
# TABLA 2: PERFILES (Roles RBAC)
# Define los roles de acceso dentro del sistema
# -----

cursor.execute(
    """
    CREATE TABLE IF NOT EXISTS PERFILES (
        PERCODIGOI INTEGER PRIMARY KEY AUTOINCREMENT,
        PERDESCRIPCION TEXT NOT NULL,
        PERESTADO INTEGER NOT NULL DEFAULT 1,
        PEROBSERVACION TEXT
    );

```

```

""""

)

# -----

# TABLA 3: USUARIOS_PERFILES

# Relación M:N entre Usuarios y Perfiles/Roles

# -----

cursor.execute(

    """

    CREATE TABLE IF NOT EXISTS USUARIOS_PERFILES (

        USPCODIGOI INTEGER PRIMARY KEY AUTOINCREMENT,

        USRCODIGOI INTEGER NOT NULL,

        PERCODIGOI INTEGER NOT NULL,

        FOREIGN KEY (USRCODIGOI) REFERENCES USUARIOS

(USRCODIGOI) ON DELETE CASCADE,

        FOREIGN KEY (PERCODIGOI) REFERENCES PERFILES

(PERCODIGOI) ON DELETE CASCADE

    );

    """

)

# -----

# TABLA 4: MODULOS

# Define las secciones, rutas/URLs y niveles del sistema

# -----

cursor.execute(

```

""""

CREATE TABLE IF NOT EXISTS MODULOS (

MODCODIGOI INTEGER PRIMARY KEY AUTOINCREMENT,

MODDESCRIPCION TEXT NOT NULL,

MODURL TEXT NOT NULL,

MODNIVEL INTEGER NOT NULL

);

""""

)

TABLA 5: MODULOS_PERFILES (Matriz de Permisos RBAC)

Control granular de accesos (Ver, Crear, Editar, Eliminar)

cursor.execute(

""""

CREATE TABLE IF NOT EXISTS MODULOS_PERFILES (

MOPCODIGOI INTEGER PRIMARY KEY AUTOINCREMENT,

MODCODIGOI INTEGER NOT NULL,

PERCODIGOI INTEGER NOT NULL,

PERM_VER INTEGER NOT NULL DEFAULT 1,

PERM_CREAR INTEGER NOT NULL DEFAULT 1,

PERM_EDITAR INTEGER NOT NULL DEFAULT 1,

PERM_ELIMINAR INTEGER NOT NULL DEFAULT 1,

FOREIGN KEY (MODCODIGOI) REFERENCES MODULOS
(MODCODIGOI) ON DELETE CASCADE,

```
FOREIGN KEY (PERCODIGOI) REFERENCES PERFILES  
(PERCODIGOI) ON DELETE CASCADE
```

```
);
```

```
""
```

```
)
```

```
# -----
```

```
# POBLAMIENTO INICIAL DE DATOS
```

```
# -----
```

```
# 1. Perfil Administrador
```

```
cursor.execute(
```

```
""
```

```
INSERT OR IGNORE INTO PERFILES (PERCODIGOI,  
PERDESCRIPCION, PERESTADO, PEROBSERVACION)
```

```
VALUES (1, 'Administrador', 1, 'Acceso Total al Sistema');
```

```
""
```

```
)
```

```
# 2. Módulos Base del Prototipo
```

```
cursor.execute(
```

```
""
```

```
INSERT OR IGNORE INTO MODULOS (MODCODIGOI,  
MODDESCRIPCION, MODURL, MODNIVEL)
```

```
VALUES (1, 'Gestión de Seguridad', '/api/v1/auth', 1);
```

```
""
```

```
)
```

```

cursor.execute(
    """
    INSERT OR IGNORE INTO MODULOS (MODCODIGO,
    MODDESCRIPCION, MODURL, MODNIVEL)
    VALUES (2, 'Auditoría de Accesos', '/api/v1/audit', 2);
    """
)

```

3. Asignación de Permisos RBAC al Perfil Administrador (Acceso Total: 1, 1, 1, 1)

```

cursor.execute(
    """
    INSERT OR IGNORE INTO MODULOS_PERFILES
    (MOPCODIGO, MODCODIGO, PERCODIGO, PERM_VER,
    PERM_CREAR, PERM_EDITAR, PERM_ELIMINAR)
    VALUES (1, 1, 1, 1, 1, 1, 1);
    """
)

```

```

cursor.execute(
    """
    INSERT OR IGNORE INTO MODULOS_PERFILES
    (MOPCODIGO, MODCODIGO, PERCODIGO, PERM_VER,
    PERM_CREAR, PERM_EDITAR, PERM_ELIMINAR)
    VALUES (2, 2, 1, 1, 1, 1, 1);
    """
)

```

4. Usuario Default: admin

```
cursor.execute("SELECT USRCODIGOI FROM USUARIOS WHERE  
USRLOGIN = 'admin'")
```

```
row_admin = cursor.fetchone()
```

```
if not row_admin:
```

```
    secret_admin = "34VDHIPNDEHWLQX32XHOJE52I62XXEU2"
```

```
    cursor.execute(  
        """
```

```
        INSERT INTO USUARIOS
```

```
        (USRCEDULAC,          USRAPELLIDOP,          USRAPELLIDOM,  
        USRNOMBRES, USRLOGIN, USRPASSWORD, USRESTADO, cliente_id,  
        totp_secret)
```

```
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
```

```
        """,
```

```
        (  
            "1728043421",  
            "Gavilanez",  
            "Jaguaco",  
            "Brandon",  
            "admin",  
            "admin123",  
            1,  
            1,  
            secret_admin,  
        ),  
    )
```

```
    usr_id = cursor.lastrowid
```

```

        cursor.execute(

            "INSERT INTO USUARIOS_PERFILES (USRCODIGOI,
PERCODIGOI) VALUES (?, 1)",

            (usr_id,),

        )

    conn.commit()

    conn.close()

    print(

        "Base de Datos BD_Seguridad.db estructurada y corregida según el
Diccionario de Datos de la Tesis."

    )

```

```

if __name__ == "__main__":

    inicializar_bd_oficial_tesis()

```

Anexo 2: Encuesta aplicada

¿Cuál es su nivel de conocimiento sobre el concepto de Autenticación Multifactor (MFA)?

¿Con qué frecuencia utiliza aplicaciones autenticadoras (ej. Google Authenticator) en su dispositivo móvil?

¿Considera que las contraseñas estáticas tradicionales son suficientes para proteger accesos del sistema?

¿Ha experimentado alguna vez situaciones de vulneración o suplantación de credenciales?

¿Qué tan familiarizado se encuentra con el escaneo de códigos QR para la vinculación de cuentas de seguridad?

¿Conoce el funcionamiento de los códigos temporales TOTP de 6 dígitos que cambian cada 30 segundos?

¿Estaría dispuesto a ingresar un segundo factor de autenticación para garantizar la seguridad de su perfil?

¿Posee conocimiento sobre el modelo de Control de Acceso Basado en Roles (RBAC)?

¿Qué nivel de importancia le asigna a la centralización de credenciales y permisos dentro del prototipo?

¿Considera adecuada la exigencia obligatoria de autenticación multifactor en los sistemas institucionales?