Interamericana de Puerto Rico

Recinto de Bayamón

Programa de Ciencias en Computadora

Mini-App Segura Help Desk (Ticket System)

Informe del Proyecto de Ciberseguridad

COMP 2700 – Cybersecurity

Sección: 92558

Presentado por:

Dean M. Ortiz Díaz

Profesor:

Prof. Hacniel Cardona

Fecha de entrega:

26 de noviembre de 2025

INTRODUCCIÓN

Este proyecto consiste en una aplicación sencilla de línea de comandos que permite manejar un sistema de tickets de soporte. La función principal es que un usuario pueda crear un ticket cuando tenga un inconveniente y que un agente pueda administrarlo y resolverlo. Todo el sistema trabaja de forma local y la información se guarda en archivos que el mismo programa maneja.

El enfoque está dirigido a aplicar principios básicos de ciberseguridad dentro de la programación. Por esa razón, se incluyen prácticas como contraseñas protegidas mediante hashing, control de acceso según el rol del usuario, validación de los datos que se ingresan y un archivo de auditoría donde se registran las acciones importantes que ocurren dentro del sistema.

Para el desarrollo se utilizó el lenguaje Python, sin bases de datos ni interfaces gráficas. Se buscó mantener la aplicación sencilla, pero demostrando que es posible reforzar la seguridad desde el diseño, evitando accesos indebidos y protegiendo la información de los usuarios. Este proyecto sirve como ejemplo de cómo las buenas prácticas de seguridad pueden implementarse incluso en programas pequeños.

OBJETIVOS DEL PROYECTO

Objetivo General

Desarrollar una aplicación de línea de comandos que permita gestionar tickets de soporte de manera segura, aplicando controles de acceso, manejo adecuado de credenciales y registro de auditoría, con el fin de demostrar prácticas básicas de ciberseguridad en el desarrollo de software.

Objetivos Específicos

• Implementar un sistema de autenticación que almacene contraseñas utilizando hashing seguro.

• Establecer permisos según el rol del usuario (user o agent) para garantizar el principio de mínimo privilegio.

• Registrar en un archivo de auditoría las acciones relevantes del sistema como inicio de sesión, creación y cambios en los tickets.

• Validar y sanitizar la entrada de datos para evitar errores lógicos y la manipulación maliciosa de la información.

• Utilizar archivos planos como mecanismo de almacenamiento de datos para asegurar un sistema sencillo y funcional sin dependencias externas.

• Incluir una demostración de seguridad que muestre la prevención de errores comunes como índices fuera de rango y límites incorrectos en la creación de datos.

DESCRIPCIÓN DEL SISTEMA

La aplicación desarrollada permite administrar un sistema de tickets de soporte utilizando una interfaz de texto. Los usuarios pueden registrarse, iniciar sesión y, dependiendo del rol que tengan, realizar diferentes acciones dentro del programa. Toda la información se almacena de forma local en archivos de texto con formato JSON por línea, lo que permite que el sistema funcione sin conexión a internet y sin necesidad de bases de datos externas.

El sistema reconoce dos tipos de usuarios: “user” y “agent”. El usuario común puede crear tickets y ver solo los que le pertenecen. El agente, por otro lado, puede ver todos los tickets, actualizarlos y cerrarlos según sea necesario. También se registra un historial de acciones en un archivo de auditoría llamado audit.log, lo que permite conocer quién realizó cada operación y cuándo la hizo, añadiendo trazabilidad al sistema.

El funcionamiento del programa se organiza a través de menús interactivos en consola. Una vez que el usuario inicia sesión, se le presentan las opciones disponibles según su rol, como crear un ticket, ver detalles de uno existente o cerrar sesión. El diseño se enfocó en que la navegación fuera sencilla y que cada acción del usuario quedara registrada correctamente para cumplir con los objetivos de seguridad del proyecto.

FUNCIONALIDADES PRINCIPALES

El sistema permite realizar diferentes acciones dependiendo del rol del usuario que haya iniciado sesión. Cada funcionalidad fue desarrollada considerando las buenas prácticas de seguridad establecidas para este proyecto.

Las funcionalidades principales son las siguientes:

• Registro de usuarios: Permite crear cuentas nuevas utilizando un nombre de usuario y contraseña. La contraseña no se guarda en texto claro, sino que se convierte en un hash mediante el algoritmo SHA-256.

• Inicio de sesión: Valida que las credenciales sean correctas antes de otorgar acceso al sistema.

• Control de acceso por rol: La aplicación identifica si el usuario es “user” o “agent” y solo muestra las opciones permitidas para su rol.

• Creación de tickets: Los usuarios pueden documentar problemas o solicitudes creando un ticket con título, descripción y prioridad.

• Lectura de tickets: El usuario común puede ver únicamente sus tickets; el agente puede ver todos los tickets del sistema.

• Actualización de tickets: Se permite modificar el estado, la prioridad, la descripción y la asignación del ticket, dependiendo del rol del usuario.

• Eliminación de tickets: Solo los agentes pueden eliminar un ticket, y únicamente si ya está en estado “closed”.

• Auditoría de acciones: Se registra en un archivo de log cada acción importante del sistema para asegurar transparencia y trazabilidad.

• Demostración de seguridad: El sistema incluye funciones para mostrar cómo se previenen errores lógicos como índices fuera de rango y exceder límites permitidos.

Con estas funciones, el sistema cumple con los requisitos principales establecidos para el proyecto y demuestra un enfoque claro hacia la seguridad desde el diseño.

CONTROL DE ACCESO POR ROL

El sistema utiliza un control de acceso basado en roles para decidir qué acciones puede realizar cada usuario dentro del programa. Esto se hace con el propósito de aplicar el principio de mínimo privilegio, el cual indica que cada usuario solo debe tener acceso a las funciones que realmente necesita.

En la aplicación existen dos tipos de usuarios: “user” y “agent”. El usuario común puede crear tickets y administrar únicamente los que le pertenecen. Por otro lado, el agente tiene más permisos, ya que es la persona encargada de gestionar y resolver los casos creados en el sistema.

La siguiente tabla resume las acciones disponibles para cada rol:

Acción del Sistema	Usuario (user)	Agente (agent)
Registrarse	Sí	Sí
Iniciar sesión	Sí	Sí
Crear tickets	Sí	Sí
Ver tickets	Solo los propios	Sí, puede ver todos
Editar tickets	Solo los propios	Sí, puede editar todos
Cambiar estado del ticket	Solo los propios	Sí, puede cambiar todos
Reasignar ticket a otro agente	No	Sí
Eliminar tickets	No	Sí, solo si el ticket está cerrado

Este control de permisos ayuda a proteger la integridad del sistema y los datos almacenados, evitando que cualquier usuario pueda modificar información que no le corresponde.

ESTADOS Y PRIORIDADES DEL TICKET

Cada ticket dentro del sistema cuenta con dos propiedades importantes que ayudan a organizar su manejo: el estado y la prioridad. Estos dos elementos permiten que los usuarios y agentes sepan en qué parte del proceso se encuentra un ticket y qué tan urgente es atenderlo.

El sistema utiliza tres estados principales para los tickets. Estos estados permiten llevar un control básico del flujo de trabajo:

- open: Indica que el ticket fue creado y aún no ha sido atendido.
- in_progress: Señala que alguien está trabajando en el ticket.
- closed: Se utiliza cuando el ticket ya fue atendido y resuelto. En este estado no se permiten modificaciones adicionales.

Además de los estados, cada ticket tiene una prioridad asignada. Esta prioridad ayuda a organizar el orden en el que se deben atender los casos:

- low: El ticket no requiere atención inmediata.
- medium: Requiere ser atendido dentro de un tiempo razonable.
- high: Es una solicitud urgente y debe ser atendida lo antes posible.

Para evitar inconsistencias, el sistema valida que los estados y prioridades solo puedan ser uno de los valores permitidos. También se controlan las transiciones entre estados. Por ejemplo, un ticket en “closed” no puede regresar a “open”, y uno en “open” no puede saltar directamente a

“closed” sin pasar por el proceso correspondiente. Estas restricciones ayudan a mantener un flujo claro y ordenado dentro del programa.

SEGURIDAD IMPLEMENTADA

La aplicación se desarrolló tomando en consideración medidas básicas de seguridad para proteger la información de los usuarios y evitar accesos no autorizados. Aunque se trata de un programa sencillo, se aplicaron distintas técnicas para reforzar la seguridad desde el diseño y en la ejecución del sistema.

Una de las primeras medidas aplicadas fue el uso de hashing para las contraseñas. En lugar de guardar la contraseña escrita por el usuario, el programa genera un hash utilizando el algoritmo SHA-256 y almacena únicamente ese valor en los archivos del sistema. De esta manera, aunque alguien tuviera acceso al archivo de usuarios, no podría conocer las contraseñas reales.

Otra medida importante es el control de acceso por roles. El programa diferencia entre usuarios comunes y agentes, y solamente permite a cada uno realizar las acciones que le corresponden. Esta estructura evita que un usuario sin permisos pueda eliminar o modificar información que no le pertenece.

También se valida la información que escribe el usuario antes de procesarla. Con esto se evita que datos incorrectos o malintencionados puedan causar errores o afectar el funcionamiento del sistema. Los menús verifican que la entrada sea un número válido y los textos se revisan para que no rompan el formato de los archivos.

Finalmente, todas las acciones relevantes del sistema son registradas en un archivo de auditoría. Esto permite tener un historial claro de quién hizo qué dentro del sistema y en qué momento sucedió. Esta funcionalidad ayuda a detectar malos usos, intentos de acceso indebido o cualquier comportamiento sospechoso dentro del programa.

En conjunto, estas medidas demuestran que es posible desarrollar software sencillo utilizando prácticas fundamentales de ciberseguridad y manteniendo la integridad de los datos manejados por la aplicación.

1.1 Hashing de Contraseñas (SHA-256)

Para proteger las credenciales de los usuarios, el sistema utiliza hashing al momento de almacenar las contraseñas. Esto significa que la contraseña nunca se guarda en texto claro dentro de los archivos del programa. En su lugar, el sistema aplica el algoritmo SHA-256 para generar un valor único basado en la contraseña original.

El hash resultante es irreversible, lo que significa que no se puede recuperar la contraseña a partir de ese valor. De esta manera, si alguien obtiene acceso al archivo donde se guardan los usuarios, no podrá conocer las contraseñas reales ni utilizarlas para entrar al sistema.

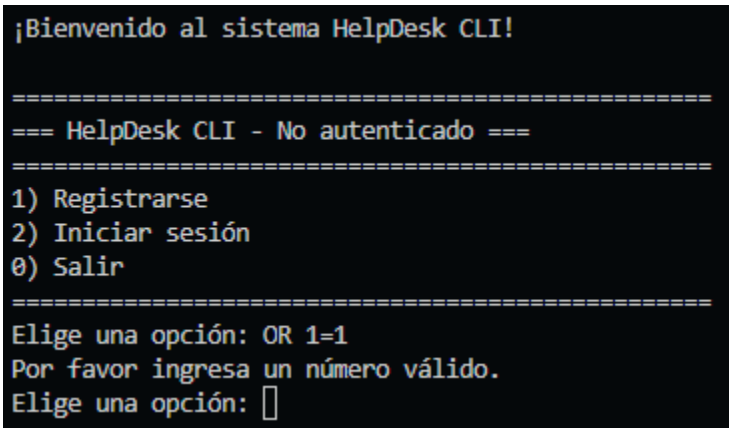
Esta práctica se considera esencial dentro de la ciberseguridad, ya que protege la información sensible de los usuarios incluso si ocurre una filtración accidental o un acceso directo al archivo de usuarios.

1.2 Validación y Sanitización de Datos

El sistema valida toda la información que escribe el usuario antes de procesarla. Esto incluye verificar que el nombre de usuario tenga un formato permitido, que los números de los menús sean válidos y que los campos de texto no excedan la longitud establecida. Con esto se evita que entradas incorrectas causen errores en la aplicación.

También se aplica sanitización al texto ingresado en los tickets, eliminando saltos de línea innecesarios y caracteres que puedan afectar la estructura de los datos almacenados. Esto evita que una entrada malintencionada pueda dañar el formato de los archivos del sistema o alterar la forma en que la aplicación los interpreta.

Gracias a estas validaciones, se mantiene la integridad de los datos y se previene que los usuarios ingresen información que pueda generar fallos o comportamientos inesperados dentro del programa.



1.3 Auditoría de Eventos (Logging)

El sistema registra en un archivo de auditoría todas las acciones importantes que realizan los usuarios dentro del programa. Entre los eventos que se guardan se encuentran el inicio y cierre de sesión, la creación de tickets, los cambios en su estado o prioridad y cualquier modificación importante relacionada al sistema.

El archivo de auditoría permite tener un historial claro de lo que ha ocurrido, quién lo hizo y en qué momento. Esta información es útil para detectar intentos de uso indebido, validar acciones realizadas y mantener transparencia en el manejo de los datos. Además, el log no almacena información sensible como contraseñas o hashes, cumpliendo con las buenas prácticas de seguridad.

Este mecanismo añade trazabilidad al sistema y forma parte de los controles que ayudan a identificar y responder ante posibles amenazas o fallos.

1.4 Prevención de Vulnerabilidades e Inyección de Datos

Uno de los objetivos del proyecto es evitar que el usuario pueda manipular la información del sistema de forma maliciosa. Aunque la aplicación no utiliza bases de datos, se tomó en cuenta la prevención de ataques como la inyección de datos o el uso de caracteres que alteren el comportamiento normal del programa.

Para esto, todas las entradas del usuario pasan por un proceso de validación antes de ser guardadas. Los menús verifican que las opciones ingresadas sean números dentro del rango

correcto y que solo se escriba texto válido en los campos del ticket. Además, el título y la descripción del ticket se revisan para que no contengan saltos de línea inesperados o símbolos que puedan dañar el formato de los archivos.

De esta forma, si un usuario intenta ingresar cadenas como "' OR 1=1 --" o cualquier otra que generalmente se utiliza en ataques de inyección, el sistema simplemente las guarda como texto sin afectar la lógica del programa. Esto protege el funcionamiento interno de la aplicación y evita la corrupción de los archivos donde se almacena la información.

1.5 Demostración de Overflow/Underflow Lógico

El sistema incluye una pequeña demostración cuyo propósito es mostrar cómo se pueden prevenir errores lógicos comunes en programación, como el desbordamiento de valores (overflow) o el acceso fuera del rango permitido (underflow). Estos errores ocurren cuando se manejan índices o cantidades sin verificar sus límites.

En la aplicación, antes de permitir que un usuario seleccione un ticket, se valida primero que el número ingresado corresponda a un ticket existente. Esto evita que el programa intente acceder a una posición inválida dentro de la lista, lo cual produciría un error o una interrupción inesperada del sistema.

Asimismo, el programa controla la cantidad de registros que pueden crearse y valida que los valores ingresados se mantengan dentro de lo permitido. Con estas verificaciones se reduce el riesgo de que se generen fallos, bloqueos o comportamientos erróneos durante la ejecución.

Esta parte del proyecto demuestra la importancia de validar tanto los límites como los accesos a los datos dentro de un sistema, ya que incluso aplicaciones sencillas pueden fallar si no se consideran estas medidas de seguridad.

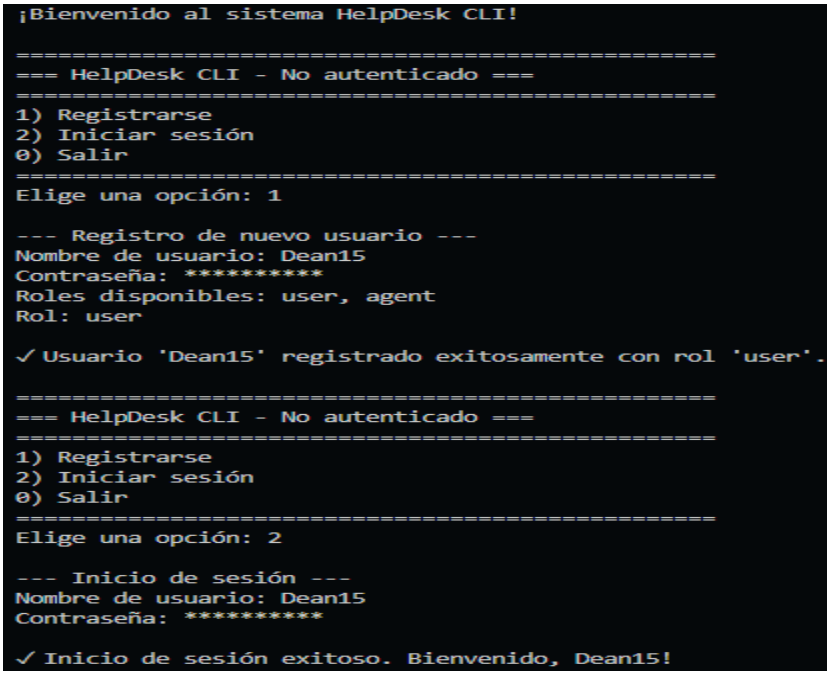
Evidencia del Funcionamiento del Sistema

A continuación se presenta evidencia del uso y funcionamiento correcto de la aplicación. En estas capturas se puede observar el proceso de registro, inicio de sesión y gestión de tickets, así como el registro de acciones en el archivo de auditoría.

En la primera imagen se muestra el menú principal del sistema luego de que el usuario inicia sesión correctamente. Dependiendo del rol asignado, el sistema presenta las opciones correspondientes para administrar los tickets.

La segunda imagen presenta la creación de un ticket desde la cuenta de un usuario, donde se ingresa el título, descripción y prioridad. También se valida que la información ingresada cumpla con los formatos establecidos.

En la tercera evidencia se muestra un fragmento del archivo audit.log, donde se registran acciones como el inicio de sesión, la creación del ticket y los cambios de estado realizados por un agente. Estos registros permiten mantener trazabilidad y verificar el cumplimiento de las medidas de seguridad.



```
✓ Inicio de sesión exitoso. Bienvenido, Dean15!

================================================
=== HelpDesk CLI - Usuario: Dean15 (user) ===
================================================
1) Crear ticket
2) Ver lista de tickets
3) Ver detalle de un ticket
4) Editar un ticket
5) Eliminar ticket (solo agentes)
6) Demostración de seguridad (overflow / underflow)
7) Cerrar sesión
0) Salir
================================================
Elige una opción: 1

--- Crear nuevo ticket ---
Título del ticket: Problemas de crear usuario
Descripción del problema: Los usuarios no pueden crear sus cuentas
Prioridades disponibles: low, medium, high
Prioridad: high

✓ Ticket #3 creado exitosamente.
```


```
{"timestamp": "2025-11-26T23:48:53.931603Z",
"user_id": "anonymous", "username": "unknown",
"role": "unknown", "action": "REGISTER", "entity":
"user", "entity_id": "2", "status": "success",
"details": "Nuevo usuario Dean15 con rol user"}
{"timestamp": "2025-11-26T23:49:02.798483Z",
"user_id": "2", "username": "Dean15", "role":
"user", "action": "LOGIN", "entity": "user",
"entity_id": "2", "status": "success", "details":
"Inicio de sesión correcto"}
{"timestamp": "2025-11-26T23:50:23.976938Z",
"user_id": "2", "username": "Dean15", "role":
"user", "action": "TICKET_CREATE", "entity":
"ticket", "entity_id": "3", "status": "success",
"details": "Título: Problemas de crear usuario,
Prioridad: high"}
```

CONCLUSIÓN

Este proyecto permitió aplicar conceptos esenciales de ciberseguridad dentro del desarrollo de una aplicación sencilla. A través del manejo de tickets en un sistema CLI, se integraron medidas como el hashing de contraseñas, validaciones de entrada, auditoría de eventos y control de accesos por rol, lo que demuestra que la seguridad puede formar parte del diseño desde las primeras etapas del software.

El uso de archivos locales como almacenamiento también resultó útil para comprender cómo proteger datos sin depender de bases de datos externas. Además, la demostración de prevención de errores como accesos fuera de rango ayudó a reforzar la importancia de validar correctamente los datos y límites dentro del programa.

Como resultado final, el sistema cumple con los requisitos establecidos, funciona de manera estable y mantiene la integridad de la información que maneja. Este proyecto sirve como una base práctica para continuar desarrollando aplicaciones seguras y mejorando las habilidades en programación y ciberseguridad.

REFERENCIAS

National Institute of Standards and Technology. (2018). Digital identity guidelines: Authentication and lifecycle management (Special Publication 800-63B). U.S. Department of Commerce.

National Institute of Standards and Technology. (2015). Guide to application security (Special Publication 800-95). U.S. Department of Commerce.

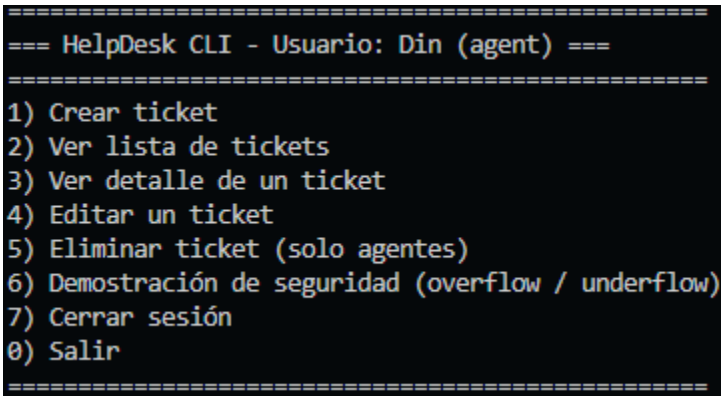
OpenAI. (2024). OpenAI documentation: Models and capabilities. https://platform.openai.com/docs (La utilice para que me ayudara en algunas cosas)

ANEXO

A. Evidencia Visual del Funcionamiento

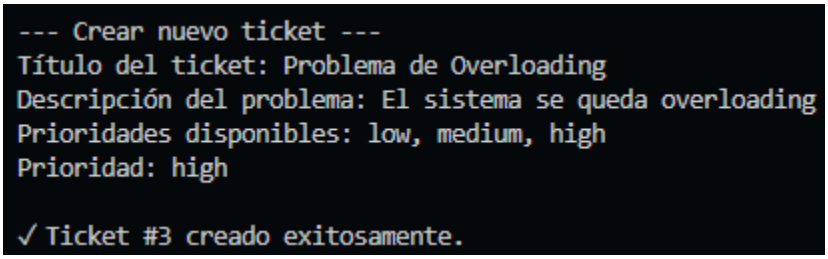
A.1 Menú principal después del inicio de sesión

En esta captura se observa la interfaz principal que aparece luego de que el usuario inicia sesión exitosamente. Dependiendo del rol asignado, el sistema despliega las opciones correspondientes para la gestión de tickets.



A.2 Creación de un ticket por parte del usuario

Aquí se muestra el proceso de creación de un ticket ingresando título, descripción y prioridad. El sistema valida los datos antes de almacenarlos.



A.3 Gestión del ticket por parte del agente y registro en auditoría

En esta evidencia se demuestra cómo un agente puede cambiar el estado o prioridad de un ticket y cómo estas acciones quedan registradas en el archivo de auditoría para mantener trazabilidad.

```
--- Editar ticket ---
ID del ticket a editar: 1

--- Editando ticket #1 ---
Título actual: Error de deploy
Descripción actual: Problemas al hacer deploy en azure...
Prioridad actual: high
Estado actual: open
Asignado a: Sin asignar

Opciones:
1) Cambiar título
2) Cambiar descripción
3) Cambiar prioridad
4) Cambiar estado
5) Reasignar ticket
0) Guardar y volver
Elige una opción: 5
Ingresa el ID del agente a asignar (o deja vacío para desasignar):
ID del agente: 2
✓ Asignación actualizada.

--- Editando ticket #1 ---
Título actual: Error de deploy
Descripción actual: Problemas al hacer deploy en azure...
Prioridad actual: high
Estado actual: open
Asignado a: 2
```


```
{"timestamp": "2025-11-27T00:52:10.325325Z",
"user_id": "3", "username": "Din", "role":
"agent", "action": "TICKET_ASSIGNEE_CHANGE",
"entity": "ticket", "entity_id": "1", "status":
"success", "details": "Ticket 1: Sin asignar ->
2"}
```


B. Fragmentos de Código Relevantes

B.1 Hashing de contraseñas con SHA-256

Este fragmento evidencia el uso del hashing para almacenar credenciales de manera segura sin exponer contraseñas reales.

```
def hash_password(password: str) -> str:
    return hashlib.sha256(password.encode()).hexdigest()
```

B.2 Control de acceso basado en roles

Este fragmento evita que un usuario sin permisos pueda realizar acciones restringidas, aplicando el principio de mínimo privilegio.

```
if current_user.role != "agent":
    print("No tiene permisos para realizar esta acción.")
    return
```

B.3 Prevención de acceso fuera de rango

Este control asegura que el usuario no pueda acceder a posiciones no válidas en la lista de tickets, evitando errores lógicos.

```
if index < 0 or index >= len(tickets):
    print("Ticket inválido.")
    return
```