

Manual del Programador — MultiDesk v5.0

Proyecto: MultiDesk

Lenguaje: Python 3.10+

Librerías utilizadas: tkinter, http.server, socketserver, requests, sqlite3, json, hashlib, tkinterdnd2, windnd, shutil, threading, urllib, entre otras.

Autores: Rocío Monzón, Augusto Kurtz, Andrés Ochoa, Thiago López

Materia: Laboratorio de Programación — Prof. Yamil Ganduglia / York Mansilla

Fecha: 07/11/2025

1. Introducción técnica

MultiDesk es una aplicación modular desarrollada en Python que permite compartir archivos en una red local mediante una arquitectura cliente-servidor basada en HTTP.

Su diseño combina una interfaz gráfica construida con Tkinter, un backend multihilo que utiliza el módulo socketserver y un sistema de almacenamiento local mediante SQLite y archivos JSON. El programa está diseñado para ser portable, multiplataforma y autosuficiente, eliminando dependencias externas de servicios en la nube.

2. Arquitectura general del sistema

La arquitectura de MultiDesk sigue el patrón Modelo-Vista-Controlador (MVC) implícito, donde cada componente cumple un rol específico:

-Modelo: se implementa mediante la base de datos SQLite y los archivos JSON que guardan la configuración del sistema y el registro de transferencias.

-Vista: está compuesta por las interfaces Tkinter (ventanas principales, formularios de login, panel de host, Multi-Square y panel de diagnóstico).

-Controlador: abarca las clases MultiDeskApp, CustomHandler y AuthTCPServer, que coordinan la interacción entre la red, el almacenamiento y la interfaz de usuario.

El sistema soporta simultáneamente múltiples clientes conectados a un host, manejando peticiones HTTP concurrentes y actualizando las listas de archivos en tiempo real.

3. Clases principales y responsabilidades

AuthTCPServer: Subclase de TCPServer con reutilización de direcciones habilitada. Gestiona las conexiones entrantes, almacena la lista de IPs participantes y un mapa de usuarios activos.

Permite que el host mantenga registro de quién está conectado y qué archivos fueron subidos.

CustomHandler: Extiende SimpleHTTPRequestHandler para manejar rutas HTTP personalizadas. Define los métodos:

- do_GET(): procesa solicitudes de estado, lista de archivos y descargas.

- do_POST(): recibe archivos enviados por los clientes.

- do_DELETE(): gestiona la desconexión de clientes o eliminación de archivos específicos.

También filtra archivos protegidos (como la base de datos o logs) e implementa seguridad básica mediante headers HTTP.

MultiDeskApp: Clase principal que inicializa la interfaz Tkinter, la configuración, el servidor y las conexiones. Contiene la lógica de negocio: registro y autenticación de usuarios, carga y guardado de configuración, inicialización del modo host o cliente, y control de subidas de archivos. Además, maneja la persistencia de las preferencias (idioma, modo temporal, tamaño del Multi-Square, etc.) en un archivo JSON.

HostControlPanel: Ventana secundaria que muestra información sobre usuarios conectados y archivos compartidos. Permite eliminar archivos, copiar el código de conexión ARAT y cerrar la sala.

Su actualización periódica se realiza con el método after() de Tkinter, garantizando sincronización no bloqueante con la interfaz.

DebugPanel: Consola visual de diagnóstico en tiempo real. Muestra mensajes de log categorizados por color (verde para debug, rojo para error).

Utiliza el método update_info() para añadir mensajes de forma thread-safe y marcar eventos con timestamp.

4. Base de datos local (SQLite)

MultiDesk utiliza SQLite como sistema de persistencia para almacenar usuarios locales. Esto permite autenticar y registrar accesos sin necesidad de servidores externos.

La base de datos `multidesk.db` contiene una única tabla denominada `Usuarios` con la siguiente estructura:

```
CREATE TABLE IF NOT EXISTS Usuarios (
    id INTEGER PRIMARY KEY,
    username TEXT UNIQUE NOT NULL,
    password_hash TEXT NOT NULL
);
```

La autenticación se realiza mediante el algoritmo SHA-256. Las funciones principales son:

- setup_db(): inicializa la base si no existe.
- register_user(): inserta un nuevo usuario, controlando duplicados.
- authenticate_user(): compara hashes para validar credenciales.

Cada usuario tiene un único registro local, y la sesión activa se guarda temporalmente en memoria dentro del objeto MultiDeskApp.

5. Comunicación HTTP y flujo de red

El servidor se implementa usando las clases http.server y socketserver. Cada conexión cliente se maneja en un hilo independiente, garantizando concurrencia y evitando bloqueos de interfaz.

El puerto predeterminado es el 8000, pero puede ser configurado dinámicamente con find_free_port().

Rutas disponibles:

- /status → verifica disponibilidad del host.
- /files_list → devuelve un JSON con la lista de archivos compartidos.
- /<nombre_de_archivo> → permite la descarga directa.
- /delete_file → permite eliminar archivos (solo por uploader o host).
- /leave → desconecta un cliente.

Cada petición se valida mediante headers personalizados:

- X-Username: usuario autenticado.
- X-Client-Ip: dirección IP del cliente.
- X-Filename: nombre del archivo en subida.

La transferencia de archivos se realiza binariamente. La escritura en disco se efectúa con buffers controlados y verificaciones de seguridad para evitar sobreescrituras de archivos protegidos.

6. Sistema ARAT

El sistema ARAT permite traducir direcciones IP a secuencias de palabras comprensibles, facilitando la conexión entre usuarios sin necesidad de conocer direcciones numéricas.

Ejemplo:

192.168.0.1 → "qoq qaq.sol mam.lo l lel"

Esto se realiza mediante los diccionarios WORDS_MAP y las funciones:

- encode_ip(ip): convierte una IP a texto codificado.
- decode_ip(string): convierte de texto a IP numérica.
- encode_number(number): traduce números en secuencias aleatorias de palabras.
- decode_number(string): realiza la decodificación inversa.

El sistema genera códigos únicos por sesión y mantiene compatibilidad con el formato legible para humanos.

7. Interfaz gráfica (Tkinter)

MultiDesk cuenta con una interfaz gráfica modular construida completamente en Tkinter. Se utiliza un enfoque basado en ventanas independientes (Toplevel) para cada sección.

La interfaz incluye:

- Pantalla principal: con botones para hostear, conectarse o registrar usuario.
- Ventana de host: muestra usuarios conectados, archivos disponibles y controles administrativos.
- Ventana de cliente: permite subir, descargar y eliminar archivos propios.
- Panel de configuración: ajusta idioma, tamaño de Multi-Square y modo temporal.
- Multi-Square: ventana flotante sin bordes para arrastrar archivos (Drag & Drop).

El diseño mantiene compatibilidad tanto con Windows como Linux y macOS, utilizando detección de plataforma mediante sys.platform.

Los mensajes están traducidos mediante el diccionario TRANSLATIONS, soportando inglés y español.

8. Módulo Multi-Square

El componente Multi-Square es una ventana auxiliar opcional diseñada para aumentar la productividad mediante la función de arrastrar y soltar archivos directamente en el sistema. Permite registrar eventos DnD usando dos librerías alternativas:

- tkinterdnd2: implementación multiplataforma.
- windnd: alternativa optimizada para Windows.

El sistema detecta automáticamente cuál backend está disponible y registra el canvas como destino de archivos.

El archivo arrastrado se procesa por la función `_handle_drop_event()` que extrae las rutas y ejecuta la subida automática.

También se guarda el tamaño y posición del cuadrado en el archivo de configuración para restaurarse en sesiones futuras.

9. Persistencia y configuración JSON

El archivo `multidesk_config.json` almacena los ajustes de sesión y configuración persistente.

Campos incluidos:

```
{  
    "temporal_mode": true/false,  
    "multi_square_enabled": true/false,  
    "multi_square_size": [ancho, alto],  
    "multi_square_position": [x, y],  
    "language": "es/en"  
}
```

Se manipula mediante las funciones `load_config()` y `save_config()`, que manejan conversiones seguras y valores por defecto en caso de errores de lectura o corrupción del archivo.

10. Panel de depuración y logs

El sistema de diagnóstico registra mensajes en tiempo real sobre el estado de la aplicación, conexiones y errores.

El panel `DebugPanel` implementa un `Listbox` con colores dinámicos y timestamps para facilitar la trazabilidad.

Los mensajes se generan desde el método centralizado `update_debug_info()`, que redirige tanto a la GUI como a la consola si el panel está cerrado.

Categorías de log:

- [DEBUG]: eventos normales del sistema.
- [ERROR]: fallos de red, archivo o autenticación.
- [INFO]: mensajes de estado o conexión exitosa.

11. Modo temporal y limpieza automática

Cuando se activa el modo temporal, todos los archivos compartidos se eliminan automáticamente al cerrar la sesión o desconectar el host.

La función `cleanup_multidesk()` recorre el directorio `MultiDesk` y elimina archivos no protegidos (excluyendo logs y base de datos).

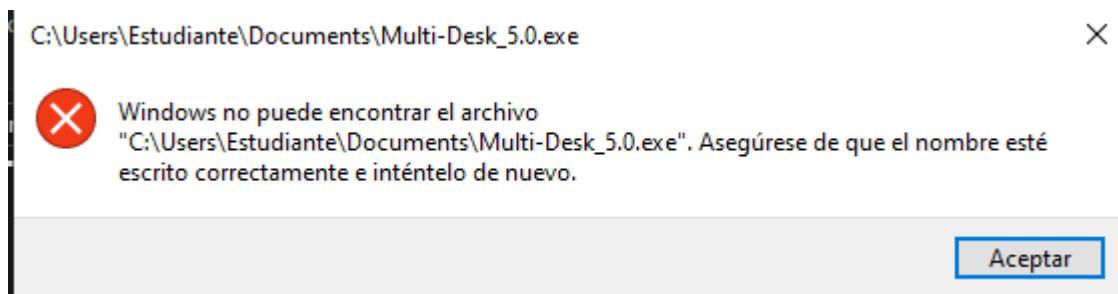
Esto se utiliza para laboratorios o aulas donde no se desea conservar archivos tras cada uso.

12. Error de instalación.

Va a surgir un problema que no le va a permitir al usuario la instalación del programa, y tiene que ver con el tipo de licencia que poseemos y la detección del mismo por el sistema operativo WINDOWS, debido a que utilizamos datos sensibles como la dirección IP.

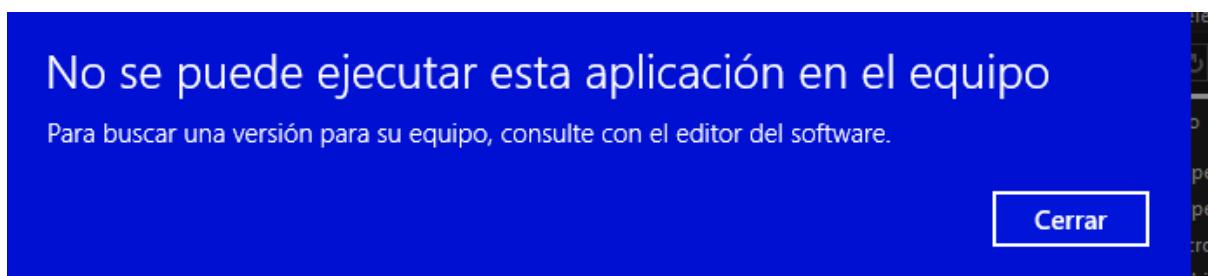
El primer obstáculo encontrado durante la descarga y ejecución del proyecto es la interferencia directa del software antivirus (como Windows Defender o la protección de Google Chrome).

El Error:



Además, durante la descarga de la aplicación ejecutable (.exe) desde GitHub, el navegador o el sistema operativo detectan el archivo como una amenaza y lo bloquean o eliminan automáticamente. Este comportamiento se traduce en dos mensajes secuenciales para el usuario:

Advertencia de Virus/Seguridad durante la descarga.



El error de "Windows no puede encontrar el archivo..." (como se observa en la Imagen) al intentar ejecutarlo.

La Causa:

El ejecutable no posee una firma digital válida de un editor reconocido, lo que provoca un falso positivo. El software de seguridad lo categoriza como una amenaza potencial y, en consecuencia, lo pone en cuarentena o lo elimina del sistema justo después de la descarga. El segundo error es una consecuencia del primero: el sistema operativo no puede encontrar el archivo porque ha sido borrado por el antivirus.

2. Incompatibilidad de Entorno de Ejecución

Una vez que el archivo logra ser descargado e intenta ser ejecutado, puede surgir un error relacionado con la compatibilidad del sistema.

El Error:

Al hacer doble clic en el archivo Multi-Desk_5.0.exe, aparece la pantalla de fondo azul con el mensaje: "No se puede ejecutar esta aplicación en el equipo" (como se observa en la Imagen). El sistema operativo no permite el inicio del proceso.

La Causa:

La causa más probable es una incompatibilidad de arquitectura entre el ejecutable y el sistema operativo del usuario. Si la aplicación fue compilada para una arquitectura de 64 bits (x64) y se intenta ejecutar en una versión de Windows de 32 bits (x86), el sistema operativo rechaza la ejecución.

Una causa secundaria es la ausencia de dependencias de runtime críticas. Si la aplicación fue construida con frameworks como .NET o Visual C++, y el equipo de destino no tiene instalados los paquetes redistribuibles correspondientes, la aplicación no puede inicializarse correctamente y muestra este error genérico de incompatibilidad.

13. Créditos

Desarrolladores:

- Rocío Monzón
- Augusto Kurtz
- Andrés Ochoa
- Thiago López

Docentes:

- Yamil Ganduglia
- York Mansilla

Versión actual: v5.0 (20/11/2025)