Report Software:   
Scanner di Rete e Verifica HTTP

# Introduzione

I progetti "Scanner di Rete” e “Verifica HTTP" sono stati sviluppati per fornire un'analisi dettagliata delle porte di rete e dei metodi HTTP supportati da server remoti. Lo scopo principale è quello di facilitare le diagnosi di rete e verificare la conformità dei server ai protocolli standard. Attraverso un'implementazione modulare e l'uso di librerie Python, il progetto offre uno strumento semplice ma potente per il monitoraggio della rete e l'analisi dei servizi.

## Obiettivi del Progetto

Il progetto si compone di due strumenti software distinti: uno per la scansione delle porte di rete e l'altro per la verifica dei metodi HTTP. Entrambi sono progettati per affrontare specifiche necessità di diagnostica di rete e sicurezza, con obiettivi chiari e complementari.

### Scansione delle Porte sulla Rete

L'obiettivo principale di questo software è fornire un'analisi dettagliata delle porte di rete di uno o più dispositivi per scopi di diagnostica, monitoraggio e sicurezza. In particolare:

* Identificazione dello stato delle porte: Determinare quali porte siano aperte, filtrate o chiuse per rilevare vulnerabilità o configurazioni anomale.
* Supporto alla configurazione di firewall: Aiutare gli amministratori di rete a ottimizzare le regole del firewall attraverso un'analisi delle porte attualmente accessibili.
* Scalabilità: Fornire una soluzione efficace per la scansione di un ampio intervallo di porte su dispositivi singoli o multipli, mantenendo tempi di risposta accettabili grazie alla parallelizzazione.
* Reportistica chiara e dettagliata: Documentare i risultati della scansione in file log leggibili e consultabili in un secondo momento, con timestamp e classificazione dei risultati.

### Verifica dei Metodi HTTP

Questo software è stato progettato per analizzare il comportamento dei server web rispetto ai metodi HTTP supportati, al fine di migliorare la sicurezza e la compatibilità. Gli obiettivi specifici includono:

* Determinazione dei metodi supportati: Verificare quali metodi HTTP (GET, POST, PUT, DELETE, etc.) siano accettati dal server e ottenere informazioni sulle relative risposte.
* Identificazione di metodi insicuri: Rilevare l'uso di metodi come PUT o DELETE, che potrebbero rappresentare un rischio se lasciati attivi senza necessità.
* Validazione delle configurazioni: Assicurare che il server risponda correttamente ai metodi standard, rispettando i requisiti di conformità ai protocolli HTTP/HTTPS.
* Sicurezza proattiva: Offrire suggerimenti per migliorare le configurazioni del server web basandosi sui metodi rilevati.
* Registrazione e analisi: Generare log strutturati per tracciare il comportamento dei server testati e consentire un'analisi approfondita successiva.

In sintesi, entrambi i software condividono l'obiettivo di migliorare la sicurezza e l'efficienza della gestione di rete, affrontando però problemi diversi con strumenti specializzati.

### Strumenti Utilizzati

Il progetto si basa su una combinazione di strumenti e librerie software che contribuiscono a soddisfare gli obiettivi prefissati, ognuno con un ruolo specifico e complementare.

#### **1. Python**

Python è stato scelto come linguaggio di programmazione principale per il progetto grazie alla sua flessibilità, leggibilità e alla disponibilità di librerie estese per il networking e l'elaborazione HTTP. Le sue caratteristiche principali includono:

* Supporto nativo per lo sviluppo multi-thread, essenziale per la scansione parallela delle porte.
* Una sintassi semplice che accelera lo sviluppo e facilita il mantenimento del codice.
* Una vasta comunità che fornisce supporto e librerie open-source per migliorare le capacità del programma.

#### **2. Librerie**

**a) Socket**La libreria standard **socket** di Python è stata utilizzata per gestire la comunicazione di rete di basso livello. È uno strumento indispensabile per stabilire connessioni TCP e determinare lo stato delle porte. Le sue caratteristiche includono:

* Supporto per connessioni TCP e UDP.
* Configurazione di timeout personalizzati per evitare blocchi durante la scansione.
* Flessibilità nell'implementare protocolli di rete personalizzati.

**b) Request**La libreria **request** è stata impiegata per verificare i metodi HTTP supportati dai server remoti. Essa consente l'invio di richieste HTTP personalizzate e fornisce strumenti per analizzare le risposte dei server. I suoi punti di forza sono:

* Un'interfaccia intuitiva per configurare richieste HTTP.
* Capacità di gestire intestazioni, parametri di query e metodi avanzati come PUT e DELETE.
* Gestione robusta degli errori e dei codici di risposta HTTP.

**c) Requests  
Requests** è una libreria di alto livello per la gestione delle connessioni HTTP, ampiamente utilizzata per interazioni standard con server web. Nel progetto, ha facilitato l’invio di richieste standardizzate e l’interpretazione delle risposte HTTP.

* Include supporto integrato per HTTPS e certificati SSL.
* Fornisce funzionalità per gestire redirect e sessioni.
* Consente l'aggiunta di payload JSON nelle richieste POST.

**d) os**La libreria **os** è stata utilizzata per interagire con il file system locale. Le sue principali applicazioni nel progetto includono:

* Creazione di file per i log e i report dei risultati.
* Navigazione nel file system per identificare percorsi e verificare la presenza di directory.
* Gestione delle autorizzazioni di lettura/scrittura per i file.

**e) Datetime  
Datetime** è stata impiegata per aggiungere timestamp ai log e ai report generati, migliorando la tracciabilità dei risultati. Le sue funzionalità principali includono:

* Generazione di timestamp accurati per ogni evento registrato.
* Conversione di date e ore in formati leggibili o compatibili con sistemi esterni.
* Calcolo della durata delle operazioni di scansione.

**f) Logging**La libreria **logging** è stata integrata per fornire un sistema centralizzato di registrazione degli eventi, che consente di documentare tutte le operazioni eseguite dal programma. Le sue caratteristiche includono:

* Creazione di log in diversi livelli di severità (info, warning, error).
* Supporto per formati personalizzati che includono timestamp, messaggi e dettagli tecnici.
* Possibilità di salvare i log sia in console che in file permanenti.

# 

# Port Scanner

Il software "DD Port Scanner" è un’applicazione per la scansione delle porte di rete, rivolta sia a professionisti, sia a chi in modo amatoriale vuole accertarsi della sicurezza della propria rete.

## Gli obiettivi:

### Gli obiettivi voluti e raggiunti nello sviluppo del software sono:

#### **1. Identificazione delle porte e dei loro stati**

* **Porte aperte**: Identificare quali porte accettano connessioni attive, fornendo informazioni su quali servizi sono in ascolto (ad esempio, server web, FTP, SSH). Questo permette agli utenti di individuare configurazioni errate o servizi non necessari esposti a internet.
* **Porte filtrate**: Rilevare le porte protette da firewall o regole di rete che impediscono connessioni dirette, aiutando a valutare la configurazione di sicurezza.
* **Porte chiuse**: Mostrare le porte che non rispondono, indicando potenziali superfici d’attacco ridotte.

#### **2. Usabilità e accessibilità**

* **Interfaccia utente intuitiva**: Grazie a un design pulito e chiaro, i parametri di configurazione e i risultati sono facilmente comprensibili.
* **Facilità d’uso per principianti**: Il software non richiede conoscenze avanzate di rete. Campi precompilati e controlli degli input aiutano l’utente a evitare errori durante l’inserimento dei dati.

#### **3. Automazione del processo di scansione**

* L’applicazione consente di scansionare sia intervalli ampi di porte (es. dalla porta 1 alla porta 65535) che porte specifiche definite dall’utente, risparmiando tempo e fatica rispetto a un approccio manuale.
* La funzionalità multi-thread ottimizza le prestazioni, consentendo l’analisi di centinaia di porte in pochi secondi.

#### **4. Sicurezza e gestione delle credenziali**

* **Autenticazione sicura**: Il login integrato utilizza un’API remota per validare l’identità degli utenti, prevenendo accessi non autorizzati.
* **Conformità alle normative**: Il sistema di autenticazione è progettato per supportare la tracciabilità degli utenti, utile in ambiti aziendali o normativi.

#### **5. Flessibilità nell’utilizzo**

* **Compatibilità con diverse reti**: Progettato per funzionare sia in reti locali (LAN) che in ambienti più complessi come VPN o reti pubbliche.
* **Scenari personalizzabili**: Possibilità di combinare porte specifiche con range di porte per adattarsi alle necessità dell’utente.

#### **6. Registrazione e analisi dei risultati**

* **Creazione di log dettagliati**: Ogni scansione genera un file Excel con timestamp, target IP e lo stato di tutte le porte analizzate. Questo file può essere utilizzato per audit di rete o per documentare attività di sicurezza.
* **Raccolta centralizzata dei dati**: L’integrazione del salvataggio in un formato standardizzato garantisce una facile archiviazione e condivisione.

#### **7. Robustezza e affidabilità**

* **Gestione degli errori**: L’applicazione è progettata per resistere a input incompleti o errati (es. indirizzi IP non validi, range di porte non corretti), fornendo messaggi chiari che guidano l’utente verso una correzione.
* **Funzionamento stabile**: L’uso di thread separati per ogni scansione assicura che l’applicazione rimanga reattiva anche durante processi intensivi.

#### **8. Supporto per analisi avanzate**

* Questo software non si limita alla scansione passiva ma può servire come base per attività più avanzate come il controllo delle configurazioni di firewall o la valutazione di vulnerabilità nei servizi aperti.

Con questi obiettivi, il "DD Port Scanner" non è solo uno strumento per la scansione delle porte ma una piattaforma versatile e scalabile per la gestione e la sicurezza delle reti moderne.

### **Metodologia**

La metodologia adottata nel progetto "DD Port Scanner" integra principi di progettazione modulare, ottimizzazione delle performance e una gestione accurata dell'interazione utente. La scansione delle porte è suddivisa in diverse fasi chiave, ciascuna implementata per garantire efficienza, affidabilità e chiarezza nell'output.

#### **1. Autenticazione e Interfaccia Utente**

* **Login tramite API**:  
  Prima di accedere alla funzionalità principale, l'utente deve autenticarsi fornendo username e password. Il processo di autenticazione avviene tramite una richiesta HTTP POST a un'API remota. Questo passaggio assicura che solo utenti autorizzati possano utilizzare il software, incrementando la sicurezza dell'applicazione.
* **Design dell'interfaccia**:  
  L'interfaccia è stata progettata con **CustomTkinter**, una libreria che combina la semplicità di Tkinter con uno stile moderno e altamente personalizzabile. Sono stati creati due pannelli principali:
  + **Colonna sinistra**: Area di input per l'inserimento dell'IP, del range di porte e delle porte specifiche da scansionare.
  + **Colonna destra**: Tre sezioni distinte per visualizzare i risultati delle porte aperte, chiuse e filtrate.
* **Gestione degli errori di input**:
  + Gli input dell'utente, come IP, porte iniziali/finali o liste di porte specifiche, vengono validati. L'applicazione verifica che l'IP fornito sia valido e che le porte siano numeri compresi nel range consentito (1-65535).
  + In caso di errori, l'applicazione utilizza finestre di messaggio (**messagebox**) per informare l'utente, prevenendo crash o comportamenti imprevisti.

#### **2. Configurazione della Scansione**

* **Raccolta degli input**:  
  L'utente può inserire un range di porte (porta iniziale e porta finale) e/o un elenco di porte specifiche separate da virgola. Gli input vengono combinati in un'unica lista di porte da analizzare, evitando duplicati.
* **Creazione della coda di risultati**:  
  Viene utilizzata una coda (**queue.Queue**) per gestire in modo thread-safe i risultati raccolti durante la scansione. Questo approccio consente ai thread di lavorare contemporaneamente senza conflitti.

#### **3. Esecuzione della Scansione**

* **Struttura multi-threading**:  
  La scansione delle porte viene effettuata in parallelo utilizzando il modulo **threading**. Ogni thread è responsabile di analizzare una singola porta, verificandone lo stato attraverso un tentativo di connessione TCP. Questo design aumenta significativamente la velocità del processo, soprattutto quando si analizzano molte porte.
* **Tecniche di connessione**:
  + Per ogni porta, viene creato un socket TCP utilizzando la libreria standard di Python.
  + Il socket tenta di stabilire una connessione con il target sulla porta specificata.
  + I risultati vengono categorizzati in:
    - **Aperta**: La connessione è stata stabilita con successo.
    - **Chiusa**: Il server ha rifiutato la connessione.
    - **Filtrata**: Nessuna risposta è stata ricevuta, indicando possibili blocchi da firewall.

#### **4. Raccolta e Visualizzazione dei Risultati**

* **Categorizzazione automatica**:  
  Dopo la scansione, i risultati vengono raccolti dalla coda e suddivisi in tre liste: porte aperte, filtrate e chiuse. Questo processo garantisce che i dati siano organizzati e facili da interpretare.
* **Aggiornamento dell’interfaccia**:  
  L'interfaccia utente è progettata per aggiornarsi dinamicamente:
  + I risultati delle porte aperte, chiuse e filtrate vengono mostrati in tre caselle di testo separate, con un formato chiaro e leggibile.
  + Viene visualizzato un messaggio di conferma al termine della scansione per notificare l'utente.

#### **5. Salvataggio dei Risultati**

* **File di log in formato Excel**:  
  I risultati della scansione vengono esportati in un file Excel utilizzando una libreria come **openpyxl**. Ogni scansione viene salvata in un foglio dedicato, che include:
  + Data e ora della scansione.
  + Indirizzo IP del target.
  + Range di porte analizzate.
  + Dettagli delle porte aperte, filtrate e chiuse.
* **Organizzazione e struttura del file**:  
  Il file Excel segue uno schema organizzato, con colonne dedicate a ciascun tipo di risultato. Questo approccio facilita la consultazione e permette di filtrare o ordinare i dati secondo necessità.

#### **6. Gestione degli Errori**

* **Riconoscimento e segnalazione**:  
  La metodologia integra una gestione robusta delle eccezioni per ogni fase del processo:
  + Errori di connessione all'API.
  + Timeout durante la scansione delle porte.
  + Input malformati o assenti.
* **Resilienza del software**:  
  In caso di errori imprevisti, il programma non termina l’esecuzione in modo anomalo. Viene invece mostrato un messaggio di errore specifico e i dettagli del problema sono registrati in un file di log separato.

#### **7. Integrazione API**

* L'applicazione comunica con un server remoto tramite richieste API RESTful:
  + **Autenticazione utente**: Utilizzando il metodo POST per verificare credenziali e ottenere permessi.
  + Eventuali errori di connessione o risposta dal server vengono gestiti e comunicati chiaramente all'utente.

Questa metodologia assicura che il "DD Port Scanner" sia efficiente, sicuro e affidabile, garantendo al contempo una buona esperienza utente.

### **Risultati**

La scansione effettuata dal "DD Port Scanner" restituisce un report dettagliato sulle porte analizzate, categorizzandole in base al loro stato operativo. Questa segmentazione fornisce una visione chiara della configurazione di rete e permette agli utenti di identificare immediatamente eventuali criticità. Di seguito, un'analisi approfondita dei risultati ottenuti:

#### **1. Porte Aperte**

Le porte aperte rappresentano i punti di accesso attivi sul sistema. Questi risultati sono particolarmente utili per:

* **Identificare i servizi in ascolto**: ogni porta aperta potrebbe indicare un servizio attivo, come server web (porta 80), server SSH (porta 22), o database (porta 3306).
* **Rilevare vulnerabilità potenziali**: le porte aperte possono essere sfruttate da attacchi esterni se i servizi associati non sono aggiornati o correttamente configurati.
* **Verificare la disponibilità**: per aziende o sviluppatori, queste informazioni confermano che i servizi critici sono operativi e accessibili.

Nel report finale, ogni porta aperta è elencata con il numero della porta e una breve descrizione (se il servizio è identificabile).

#### **2. Porte Filtrate**

Le porte filtrate indicano che i pacchetti inviati non ricevono risposta, generalmente a causa di firewall o regole di rete che bloccano le connessioni. Questi dati sono significativi per:

* **Valutare l’efficacia delle misure di sicurezza**: la presenza di porte filtrate è un indicatore positivo per la protezione del sistema, in quanto riduce la superficie di attacco.
* **Individuare configurazioni restrittive**: in alcuni casi, le porte filtrate possono interferire con la comunicazione legittima, e il report può aiutare a individuare configurazioni troppo rigide.
* **Riconoscere tentativi di mascheramento**: in ambienti di rete complessi, alcune porte filtrate potrebbero indicare attività sospette, come meccanismi di cloaking.

Ogni porta filtrata nel report viene descritta come "inaccessibile", con eventuali dettagli aggiuntivi sulla causa rilevata (es. *timeout* o *firewall detected*).

#### **3. Porte Chiuse**

Le porte chiuse sono quelle non in uso sul sistema o sull'indirizzo IP specificato. Sebbene meno rilevanti dal punto di vista delle vulnerabilità, i dati sulle porte chiuse aiutano a:

* **Confermare la protezione passiva**: le porte chiuse non accettano connessioni, riducendo il rischio di esposizione non necessaria.
* **Ridurre i falsi positivi**: distinguendo tra porte chiuse e filtrate, il software consente una comprensione più accurata dello stato della rete.
* **Analizzare configurazioni di rete inattive**: una lista di porte chiuse potrebbe indicare servizi che non sono più in uso o configurati per operare su porte diverse.

#### **4. Log dei Risultati**

Un aspetto cruciale del programma è la capacità di generare un file log dettagliato in formato Excel. Questo file include:

* **Metadati della scansione**: timestamp, indirizzo IP target, range di porte analizzate e durata complessiva.
* **Dettagli per categoria**:
  + Elenco completo delle porte aperte con eventuali note sul servizio associato.
  + Porte filtrate, corredate di dettagli sui possibili motivi del filtraggio.
  + Porte chiuse, organizzate in ordine crescente per una consultazione rapida.
* **Utilità del file Excel**:
  + **Conformità normativa**: il log è utile per dimostrare la conformità a politiche di sicurezza, come quelle imposte da standard ISO o PCI-DSS.
  + **Analisi retrospettiva**: i risultati archiviati possono essere utilizzati per confronti periodici e per monitorare variazioni nella configurazione di rete.
  + **Condivisione**: il file Excel può essere facilmente distribuito tra team tecnici o management per supportare decisioni informate.

#### **5. Esperienza Utente**

* **Segnalazioni visive nella UI**: il software aggiorna dinamicamente l'interfaccia utente per mostrare i risultati in tempo reale. Le porte aperte, filtrate e chiuse vengono presentate in appositi box di testo con un’interfaccia chiara e facilmente navigabile.
* **Notifica di completamento**: una finestra popup informa l'utente al termine della scansione, fornendo un riepilogo sintetico e un'opzione per aprire direttamente il file Excel generato.

Questi risultati combinati garantiscono un'analisi approfondita dello stato delle porte, fornendo agli utenti tutti gli strumenti necessari per interpretare e utilizzare le informazioni raccolte in modo efficace.

#### **Analisi**

Il "DD Port Scanner" si distingue per il suo approccio bilanciato tra funzionalità avanzate e semplicità d'uso, rendendolo uno strumento versatile e applicabile a diversi contesti, dalla diagnostica di rete alla sicurezza informatica.

##### **Punti di forza**

1. **Interfaccia utente**L'uso di **CustomTkinter** offre un design moderno e interattivo, che si allontana dalle limitazioni visive e funzionali di Tkinter tradizionale. L'interfaccia è ben organizzata in sezioni per input e risultati, permettendo una chiara separazione delle informazioni. Questo design riduce il rischio di confusione per l'utente, rendendo lo strumento adatto sia ai principianti che agli utenti esperti.
2. **Efficienza nella scansione**L'implementazione multi-thread consente di scansionare un ampio intervallo di porte in tempi ridotti. Questo approccio è particolarmente utile in scenari dove la tempestività è cruciale, ad esempio durante un'analisi di rete per rilevare vulnerabilità in tempo reale o per controlli periodici su server di produzione. La gestione delle connessioni tramite socket garantisce affidabilità, mentre il timeout impostato per ogni porta riduce il rischio di blocchi durante la scansione.
3. **Gestione degli errori e input validazione**Il programma include robusti meccanismi per prevenire input errati o incompleti, come:
   * Controlli su indirizzi IP malformati o fuori standard (ad esempio, validazione del formato IPv4).
   * Controlli su intervalli di porte non validi (es. porta finale minore della porta iniziale).
   * Segnalazioni immediate all'utente tramite finestre di errore informative, che migliorano l'esperienza utente e riducono il rischio di operazioni errate.
4. **Reportistica dettagliata**L'uso di file Excel per documentare i risultati garantisce una facile integrazione con processi aziendali esistenti. Ogni file include:
   * Timestamp della scansione.
   * Indirizzo IP target e intervallo di porte.
   * Dettagli sulle porte aperte, filtrate e chiuse.  
     Questo formato facilita la condivisione dei dati tra team e fornisce un archivio storico per analisi future, utile anche in contesti di compliance o audit.

##### **Limitazioni**

Nonostante i suoi punti di forza, il "DD Port Scanner" presenta alcune limitazioni che potrebbero essere migliorate nelle versioni successive:

1. **Mancanza di scansioni avanzate**Attualmente, il software si focalizza esclusivamente sulle connessioni TCP. L'aggiunta di funzionalità per la scansione di porte UDP o modalità avanzate come la scansione SYN o FIN (comuni negli strumenti di penetration testing come Nmap) amplierebbe l'utilità dello strumento.
2. **Assenza di visualizzazioni grafiche**Sebbene i risultati siano dettagliati e organizzati, una rappresentazione visiva (come grafici a barre o diagrammi a torta) dei dati analizzati potrebbe migliorare l'impatto comunicativo, soprattutto durante presentazioni o report aziendali.
3. **Sicurezza delle credenziali**L'attuale sistema di autenticazione utilizza un'API remota per verificare l'identità dell'utente, ma la trasmissione delle credenziali non è dichiarata crittografata. Questo potrebbe rappresentare un rischio in ambienti sensibili. L'adozione di HTTPS o di tecniche di hashing per le password migliorerebbe significativamente la sicurezza.
4. **Scalabilità su grandi reti**La gestione multi-thread è efficace per scansioni limitate, ma potrebbe incontrare difficoltà su grandi intervalli di porte o reti complesse. In questi casi, l'integrazione con sistemi distribuiti o architetture asincrone (ad esempio, tramite asyncio) potrebbe garantire una maggiore scalabilità.

##### **Opportunità di sviluppo**

Il software presenta un'ottima base per implementare nuove funzionalità che ne aumenterebbero il valore:

* **Monitoraggio continuo**: Permettere la schedulazione di scansioni periodiche e notificare l'utente in caso di variazioni nei risultati.
* **Integrazione con database**: Archiviare i risultati delle scansioni in un database centralizzato, offrendo funzionalità di ricerca e confronto tra scansioni.
* **Supporto multi-piattaforma**: Sebbene il programma sia scritto in Python e quindi eseguibile su più sistemi operativi, l'ottimizzazione per ambienti specifici (ad esempio, pacchetti precompilati per Windows o distribuzioni Docker per Linux) migliorerebbe la diffusione.

# Verifica dei metodi HTTP

Il programma consente agli utenti di eseguire richieste HTTP personalizzate verso un endpoint specifico, selezionando uno o più metodi HTTP e personalizzando URL, intestazioni e corpo della richiesta. Fornisce una GUI interattiva basata su *customtkinter* per facilitare queste operazioni.

## Gli Obiettivi

### **1. Test di Endpoint HTTP**

* Consentire agli utenti di verificare il funzionamento di API e servizi web attraverso l'invio di richieste HTTP configurabili.
* Supportare tutti i principali metodi HTTP (*GET, POST, PUT, DELETE, PATCH, HEAD, OPTIONS, TRACE*), permettendo un'analisi approfondita dei comportamenti degli endpoint.
* Fornire feedback immediato sulle risposte del server, inclusi codice di stato, intestazioni e corpo della risposta.

#### **2. Usabilità e Accessibilità**

* Offrire un'interfaccia grafica user-friendly sviluppata con **CustomTkinter**, rendendo lo strumento intuitivo e adatto anche a utenti con conoscenze limitate di programmazione.
* Integrare funzioni pratiche come la selezione rapida di metodi HTTP (individuale o globale) e campi di input organizzati per semplificare la configurazione delle richieste.
* Supportare il salvataggio dei log in formato leggibile, facilitando la condivisione dei risultati.

#### **3. Sicurezza e Gestione delle Credenziali**

* Implementare un sistema di login per controllare l'accesso all'applicazione, proteggendo le funzionalità principali.
* Offrire la possibilità di autenticarsi tramite credenziali gestite da un backend, garantendo un accesso sicuro.
* Utilizzare protocolli sicuri per la comunicazione con server esterni, riducendo il rischio di esposizione di informazioni sensibili.

#### **4. Flessibilità nell’Utilizzo**

* Permettere agli utenti di lavorare con backend diversi, sia in ambienti di **produzione** che di **test/mock**, grazie alla modalità selezionabile.
* Consentire la personalizzazione completa delle richieste HTTP, inclusi URL, intestazioni in formato dizionario e corpo del messaggio.
* Gestire scenari complessi di debugging o verifica API in modo adattabile, senza limitazioni imposte dallo strumento.

#### **5. Registrazione e Analisi dei Risultati**

* Fornire un sistema integrato per registrare sia le risposte HTTP che eventuali errori in un file di log.
* Organizzare i risultati in modo chiaro per facilitare l’analisi successiva o il confronto tra più test.
* Abilitare l’esportazione dei log per garantire la tracciabilità delle verifiche e agevolare la documentazione dei problemi riscontrati.

#### **6. Robustezza e Affidabilità**

* Gestire in modo resiliente errori comuni, come problemi di connessione o configurazioni errate delle richieste, informando l’utente con messaggi chiari.
* Garantire un'esperienza stabile, evitando crash anche in presenza di input non validi o situazioni impreviste.
* Utilizzare librerie consolidate come **Requests** e un approccio modulare al codice per migliorare la manutenibilità e la stabilità del programma.

# Metodologia

#### **Autenticazione e Interfaccia Utente**

L'applicazione inizia con una finestra di login che richiede un nome utente e una password. L'accesso è gestito attraverso una funzione di autenticazione che verifica le credenziali inserite. Questa fase serve a garantire che solo utenti autorizzati possano accedere alle funzionalità principali. La modalità di autenticazione supporta sia richieste reali che simulazioni locali (mock), selezionabili attraverso un menu a tendina.

#### **Design dell'interfaccia**

L'interfaccia utente è progettata utilizzando *customtkinter*, una libreria che consente di creare un layout moderno e altamente personalizzabile. È divisa in due sezioni principali:

* Pannello Input (a sinistra): contiene campi per URL, intestazioni e corpo della richiesta, e opzioni per selezionare i metodi HTTP.
* Pannello Output (a destra): mostra le risposte dei server e gli errori riscontrati.

Il layout responsivo permette agli utenti di lavorare comodamente su schermi di diverse dimensioni, mantenendo una chiara separazione tra input e output.

#### **Gestione degli errori di input**

Per evitare errori durante l'esecuzione delle richieste, il programma include controlli su:

* Validità dell'URL: se non viene inserito un URL, l'utente riceve un messaggio di errore.
* Formato delle intestazioni: le intestazioni devono essere fornite come un dizionario Python valido. L'uso di eval per interpretarli è pratico ma richiede un controllo aggiuntivo per evitare rischi di sicurezza.
* Campi obbligatori: un controllo verifica che i campi essenziali siano compilati prima dell'invio.

#### **Configurazione della richiesta**

Gli utenti possono personalizzare ogni richiesta HTTP scegliendo:

* URL: l'indirizzo del server o endpoint da interrogare.
* Headers: configurabili manualmente tramite un campo di testo.
* Body: contenuto della richiesta, utile per metodi come POST e PUT.
* Metodi HTTP: opzioni multiple selezionabili tramite checkbox, con la possibilità di selezionarli tutti contemporaneamente.

#### **Esecuzione della richiesta**

Quando l'utente invia la richiesta:

1. Viene iterato su ciascun metodo HTTP selezionato.
2. Per ogni metodo, viene utilizzata la funzione requests.request per inviare la richiesta con i parametri configurati.
3. Ogni risultato (status code, testo della risposta) viene raccolto e mostrato nel pannello output.

Un sistema di *try-except* gestisce eventuali errori durante l'invio, come errori di rete o risposte non valide.

#### **Raccolta e Visualizzazione dei Risultati**

I risultati delle richieste vengono suddivisi in due sezioni:

* Risposte: visualizzate in un pannello testuale, mostrano il codice di stato, il motivo (es. "200 OK") e il contenuto della risposta.
* Errori: raccolti in un pannello separato, utili per identificare problemi specifici (es. timeout, URL errati).

Questa separazione facilita l'analisi, specialmente per test su più metodi HTTP simultanei.

#### **Salvataggio dei Risultati**

L'utente può salvare le risposte e gli errori in un file log:

1. Viene mostrata una finestra di dialogo per selezionare la cartella di destinazione.
2. I risultati vengono scritti in un file .txt organizzato in due sezioni:
   * Risposte: contenuto completo delle risposte.
   * Errori: descrizione dettagliata degli errori.

#### **Gestione degli Errori**

Il programma gestisce diverse tipologie di errori:

* Errori di input: come formati errati o campi mancanti, notificati attraverso finestre di messaggio.
* Errori di rete: timeout, connessioni rifiutate o altri problemi con il server vengono catturati e mostrati all'utente in tempo reale.
* Errori imprevisti: eventuali eccezioni durante l'elaborazione vengono mostrate nel pannello degli errori, garantendo la stabilità dell'applicazione.

### **Risultati**

1. **Interazione con Endpoint**:
   * Il programma permette di inviare richieste HTTP utilizzando metodi standard (*GET, POST, PUT*, ecc.) e meno comuni (*OPTIONS, TRACE, PATCH*), adattandosi a molteplici scenari di test.
   * Supporta l’aggiunta di intestazioni e payload personalizzati, utile per simulare richieste complesse o per interfacciarsi con API che richiedono autenticazione tramite token o parametri specifici.
2. **Gestione degli errori**:
   * Gli errori sono separati dalle risposte positive, migliorando la leggibilità e semplificando il debug.
   * Errori comuni come il timeout, l’impossibilità di connettersi a un endpoint o il formato scorretto degli header vengono chiaramente indicati nella sezione *Errori*.
3. **Risposte ben organizzate**:
   * Ogni metodo HTTP selezionato produce un output distinto nella sezione *Risposte*, con codice di stato, messaggio e contenuto della risposta.
   * Il formato di output strutturato permette una comprensione immediata del risultato di ciascuna richiesta.
4. **Log centralizzati**:
   * I log vengono salvati in un file .txt, rendendo più semplice tracciare le attività di test e condividerle con altri utenti o analisti.
   * L’utente può personalizzare il percorso di salvataggio, garantendo flessibilità nell’organizzazione dei file di log.
5. **Scansione delle porte**:
   * La funzionalità di scansione consente di identificare lo stato delle porte (aperte, chiuse, filtrate, o con errore di connessione).
   * È particolarmente utile per analisi di rete di base, come testare la disponibilità di un server o verificare la configurazione di firewall.
   * I risultati vengono elaborati in parallelo, migliorando significativamente la velocità su un intervallo di porte ampio.
6. **Flessibilità di configurazione**:
   * Grazie alla selezione multi-metodo e alla possibilità di personalizzare intestazioni e corpo della richiesta, il programma si adatta sia a scenari di testing semplici sia complessi.
   * La scansione parallela delle porte garantisce performance adeguate anche per utenti avanzati.

### **Analisi**

### **Pro**

1. **Flessibilità ed Estensibilità**:
   * La capacità di inviare richieste HTTP con molteplici configurazioni lo rende ideale per testare API RESTful.
   * L'integrazione con altre funzionalità (come la scansione delle porte) amplia il contesto d’uso del programma, rendendolo utile non solo per test applicativi, ma anche per analisi di rete.
2. **User Experience (UX)**:
   * L’interfaccia basata su *customtkinter* è moderna, semplice e intuitiva.
   * L’organizzazione in sezioni distinte per input, output e configurazioni garantisce un flusso di lavoro naturale.
3. **Logging dettagliato**:
   * La possibilità di salvare risultati ed errori è un punto di forza per la documentazione e il tracciamento di test.
   * La struttura gerarchica del file di log (*Risposte* ed *Errori* separati) migliora l’organizzazione e facilita il debug.
4. **Scansione Porte**:
   * La scansione parallela migliora notevolmente l’efficienza rispetto a una scansione sequenziale.
   * L’identificazione di stati specifici delle porte (*Aperta, Chiusa, Filtrata*) fornisce un livello di dettaglio adeguato per analisi preliminari.

#### **Contro**

1. **Sicurezza e Validazione**:
   * L’utilizzo di eval per valutare le intestazioni rappresenta un rischio di sicurezza se il programma viene utilizzato con input non affidabili. Una valida alternativa sarebbe l’uso di parser JSON sicuri.
   * Non vengono effettuate validazioni pre-invio per URL, intestazioni e corpo della richiesta, il che potrebbe portare a errori evitabili.
2. **Scansione delle Porte**:
   * L’approccio basato su thread può diventare inefficiente o instabile per scansioni molto grandi senza un sistema di limitazione dei thread (ad esempio, un *ThreadPool*).
   * I risultati della scansione non vengono salvati automaticamente o integrati nel file di log, richiedendo un’implementazione manuale se necessario.
3. **Assenza di Funzionalità Avanzate**:
   * Il programma non supporta funzioni avanzate di analisi, come il parsing della risposta HTTP in formato JSON per estrarre dati specifici.
   * Mancano opzioni di personalizzazione avanzata, come autenticazione basata su certificati o gestione delle sessioni.

#### **Potenziali Miglioramenti**

1. **Miglioramenti alla Sicurezza**:
   * Rimuovere l’uso di eval in favore di parser sicuri.
   * Implementare controlli di validazione sugli input, come l’URL e le intestazioni.
2. **Gestione Migliorata della Scansione Porte**:
   * Introdurre un sistema di limitazione per i thread (ad esempio, un pool di thread).
   * Consentire il salvataggio dei risultati della scansione in un file log separato o nel file principale.
3. **Nuove Funzionalità**:
   * Aggiungere strumenti di analisi delle risposte, come il parsing automatico di JSON o XML.
   * Integrare opzioni per inviare richieste con certificati SSL o per gestire autenticazioni avanzate (OAuth2, Basic Auth, ecc.).
   * Introdurre uno storico per le richieste, che permetta di ripetere facilmente test precedenti.
4. **UI più interattiva**:
   * Aggiungere strumenti per evidenziare sintassi nelle risposte e negli errori.
   * Includere suggerimenti (tooltip) per aiutare gli utenti meno esperti a configurare correttamente intestazioni e corpo delle richieste.