MARINA MILITARE

UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

ANALISI DELLE PRESTAZIONI DI UNA RETE WI-FI

Progetto: Reti di Calcolatori e Comunicazione Digitale

Karima Charafi, Riccardo Lucchi, Vincenzo Malafronte, Francesco Rizzi, Fabrizio Sciacca

27/01/2025

Sommario

Introduzione

Nell'introduzione tratteremo della rete WI-FI e la metodologia adoperata per portare a termine il compito assegnato, l'analisi delle prestazioni di una rete WI-FI.

Il WI-FI è un insieme di tecnologie per reti locali senza fili (WLAN) sugli standard IEEE 802, il quale consente a più dispositivi (per esempio personal computer, smartphone, smart TV, ecc.) di essere connessi tra loro tramite onde radio e scambiare dati.

Il termine "Wi-Fi" sta per "Wireless Fidelity" e si riferisce a una serie di standard tecnologici che permettono la trasmissione di dati tra dispositivi attraverso onde radio, senza l'uso di cavi. Questa tecnologia è utilizzata in molti dispositivi, come computer, smartphone, tablet e altri dispositivi smart.

Il Wi-Fi, come termine, è entrato nel linguaggio comune e viene spesso utilizzato per descrivere qualsiasi tipo di connessione internet wireless. Tuttavia, la sua definizione tecnica e le sue origini sono molto più specifiche.

Il Wi-Fi opera principalmente su due bande di frequenza: 2,4 GHz e 5 GHz. La banda dei 2,4 GHz ha una portata maggiore ma è più suscettibile alle interferenze, dato che molti altri dispositivi, come i telefoni cordless e i forni a microonde, operano su questa stessa frequenza. La banda dei 5 GHz offre velocità più elevate e meno interferenze, ma ha una portata leggermente inferiore.

La sicurezza è fondamentale nel Wi-Fi. Per proteggere i dati trasmessi, il Wi-Fi utilizza protocolli di sicurezza come WEP (ora obsoleto e non sicuro), WPA e WPA2. Questi protocolli criptano i dati in modo che solo il mittente e il destinatario previsto possano leggerli.

Una rete WiFi è un sistema che ti permette di connetterti a Internet utilizzando onde radio anziché cavi. Gli access point trasmettono segnali wireless che consentono ai dispositivi compatibili di connettersi a Internet. Questo offre un'ampia copertura e ti permette di navigare in modo conveniente ovunque ci sia una connessione disponibile. Le reti WiFi pubbliche sono comuni in luoghi come caffè, hotel e aeroporti. Consentono una connessione rapida e gratuita, ma comportano anche rischi di sicurezza. Gli hacker possono intercettare il traffico sulla rete e accedere ai tuoi dati sensibili. È essenziale prendere precauzioni quando ti connetti a reti WiFi pubbliche.

OBIETTIVO

Il **"WiFi Performance Analyzer"** è un'applicazione basata su Python che consente agli utenti di analizzare le prestazioni della rete Wi-Fi. Il software permette di:

- Scansionare le reti Wi-Fi nelle vicinanze, mostrando la potenza del segnale.
- Eseguire test di velocità per misurare la latenza (ping), la velocità di download e upload.

L'interfaccia è progettata per essere intuitiva, con un design reattivo e strumenti grafici per presentare i risultati in modo chiaro e informativo.

Lo scopo del progetto consiste nel testare le varie caratteristiche di una determinata rete Wi-Fi.

Mediante le conoscenze acquisite durante il corso di Reti offerto dal prof. Ugo Lopez, abbiamo sviluppato in linguaggio Python un programma che soddisfi la richiesta. L'applicativo verrà eseguito mediante PowerShell.

ANALISI PRELIMINARE

Requisiti e Tecnologie utilizzate

Requisiti

- **Sistema Operativo**: Compatibile con Windows, macOS e Linux.
- **Hardware**: Una scheda di rete con supporto per la scansione Wi-Fi.
- **Python**: Versione 3.7 o superiore.
- **Browser**: Qualsiasi browser moderno (Chrome, Firefox, Edge).

Tecnologie Utilizzate

- **Dash**: Utilizzato per costruire l'interfaccia utente interattiva.
- **Plotly**: Per generare grafici interattivi (grafici a barre e indicatori gauge).
- **pywifi**: Per accedere ai dati delle reti Wi-Fi locali.
- **speedtest-cli**: Per effettuare i test di velocità della connessione
- è un'applicazione basata su Dash che funge da **analizzatore di performance Wi-Fi**. L'app consente all'utente di effettuare una scansione delle reti Wi-Fi disponibili e di eseguire test di velocità internet.

Panoramica Tecnica

L'app è suddivisa in tre sezioni principali:

A. **Configurazione iniziale e layout dell'app Dash**.
B. **Funzioni operative per la scansione delle reti e il test di velocità**.
C. **Callback per gestire l'interazione utente e aggiornare l'interfaccia**.

A. Configurazione e Layout

La sezione iniziale definisce l'app Dash:

- **Proprietà dell'app**:
  - Nome: WiFi Performance Analyzer.
  - Stile: Utilizzo di temi scuri con layout responsivo.
- **Elementi del layout**:
  - Titolo (H1).
  - Pulsanti: "Scan Networks" e "Run Speed Test".
  - Contenitori dinamici per i risultati (usando dcc.Loading per feedback visivi durante i caricamenti).

B. Funzioni operative

1. **scan_networks()**:
   - Scansiona le reti Wi-Fi disponibili.
   - Filtra le reti duplicate basandosi sull'SSID, mantenendo quella con segnale più forte.
   - Restituisce una lista di dizionari: {SSID: Nome della rete, Signal: Potenza del segnale in dBm}.
2. **run_speed_test()**:

- Esegue un test di velocità tramite la libreria Speedtest.
- Restituisce tre valori:
  - **Ping** (ms).
  - **Velocità di download** (Mbps).
  - **Velocità di upload** (Mbps).

C. Callback Dash

1. **display_networks(n_clicks)**:
   - Trigger: Clic sul pulsante "Scan Networks".
   - Funzionamento:
     - Se non ci sono clic, mostra un messaggio predefinito.
     - In caso di clic, esegue scan_networks() e visualizza i risultati in un grafico a barre.
2. **display_speed_test(n_clicks)**:
   - Trigger: Click sul pulsante "Run Speed Test".
   - Funzionamento:
     - Se non ci sono clic, mostra un messaggio predefinito.
     - In caso di clic, esegue run_speed_test() e visualizza:
       - Un testo con i valori di ping, download e upload.
       - Un grafico gauge per rappresentare visivamente la velocità di download.

Dettagli delle funzioni presenti

scan_networks()

Utilizza pywifi per scansionare le reti Wi-Fi disponibili.

- **Passaggi principali**:
  1. Seleziona la prima interfaccia disponibile.
  2. Esegue la scansione.
  3. Attende 3 secondi per simulare il tempo necessario.
  4. Filtra le reti duplicate basandosi sull'SSID.
- **Output**: Lista di dizionari con SSID e potenza del segnale.

run_speed_test()

Effettua un test di velocità Internet.

- **Passaggi principali**:
  1. Identifica il server ottimale per il test.
  2. Esegue il test di ping, download e upload.
  3. Converte le velocità in Mbps.
- **Output**: Ping (ms), velocità di download (Mbps), velocità di upload (Mbps).

display_networks(n_clicks)

Callback che aggiorna il contenitore con i risultati della scansione delle reti.

- **Input**: Numero di clic sul pulsante.
- **Output**: HTML con i risultati o un grafico a barre.

display_speed_test(n_clicks)

Callback che aggiorna i risultati del test di velocità e visualizza un grafico gauge.

- **Input**: Numero di clic sul pulsante.
- **Output**: HTML con risultati testuali e un grafico gauge.

STRUTTURA CODICE

Creazione dell'Applicazione

L'applicazione è inizializzata con una Dash (Dynamic Adaptive Streaming over http) (DASH), noto anche come MPEG-DASH, è una tecnica di streaming (con bitrate adattivo che permette lo streaming di contenuti multimediali su Internet tramite server web HTTP convenzionali) che analizza la performance Wi-Fi dell'utente, consentendo la scansione delle reti disponibili dell'utente in quel momento sul suo dispositivo e l'esecuzione di un test di velocità (speed test). L'app utilizza librerie come Dash, Plotly, PyWiFi e Speedtest per creare un'interfaccia utente interattiva e visualizzazioni dei dati. Un framework open-source basato su Python che consente di creare applicazioni web interattive con visualizzazioni di dati.
Dash è particolarmente utilizzato per creare dashboard e visualizzazioni di dati interattive. Utilizza **Plotly** per la parte grafica (grafici, mappe, ecc.) e si basa su **Flask** per gestire il backend. Il layout dell'app è definito utilizzando componenti come html.Div, dcc.Graph, e dcc.Slider per creare la struttura dell'interfaccia. Un **callback** è utilizzato per aggiornare dinamicamente il contenuto in base all'interazione dell'utente (in questo caso, quando si muove lo slider). Dash supporta applicazioni molto granbdi e può essere eseguito su server come Heroku o AWS.

I componenti interattivi sono integrati in un layout strutturato, con attenzione alla semplicità e all'estetica.

Importazioni

L'inizio del file importa tutte le librerie e i moduli necessari:

C: > Users > fabri > Desktop > wifi_speed_test.py > ...

```
1  from dash import Dash, html, dcc, Input, Output
2  import plotly.graph_objects as go
3  import pywifi
4  import time
5  import speedtest
6
```

Configurazione dell'App Dash

- Creazione dell'applicazione Dash:

```
8  # Creazione dell'app Dash
9  app = Dash(__name__)
10 app.title = "WiFi Performance Analyzer"
11
```

- Definizione del layout dell'applicazione:

```
# Layout dell'app
app.layout = html.Div(
    style={···
    children=[
        html.H1("WiFi Performance Analyzer", style={···

        dcc.Loading(···

        html.Div(style={'textAlign': 'center', 'marginBottom': '40px'}, children=[···

        html.Div(style={'textAlign': 'center'}, children=[···
    ]
)
```

Funzioni Operative

Scansione reti Wi-Fi:

```
# Funzione per effettuare la scansione delle reti Wi-Fi
def scan_networks():
    wifi = pywifi.PyWiFi()
    iface = wifi.interfaces()[0]
    iface.scan()
    time.sleep(3)  # Simulazione del tempo necessario per la scansione
    networks = iface.scan_results()

    # Filtrare duplicati basati sull'SSID
    unique_networks = {}
    for network in networks:
        if network.ssid not in unique_networks or network.signal > unique_networks[network.ssid]:
            unique_networks[network.ssid] = network.signal  # Conserva il segnale più forte

    return [{"SSID": ssid, "Signal": signal} for ssid, signal in unique_networks.items()]
```

Test di Velocità:

```
# Funzione per il test di velocità
def run_speed_test():
    st = speedtest.Speedtest()
    st.get_best_server()
    ping = st.results.ping
    download_speed = st.download() / 1_000_000  # Convert to Mbps
    upload_speed = st.upload() / 1_000_000      # Convert to Mbps
    return ping, download_speed, upload_speed
```

Callback Dash

Callback per la scansione delle reti:
serve ad aggiornare i risultati della scansione delle reti Wi-Fi

```python
# Callback per la scansione delle reti
@app.callback(
    Output("network-scan-results", "children"),
    Input("scan-networks-button", "n_clicks")
)
def display_networks(n_clicks):
    if not n_clicks:
        return html.P("Click 'Scan Networks' to see available Wi-Fi networ
    networks = scan_networks()  # Chiama la funzione aggiornata
    if not networks:
        return html.P("No networks found. Please try again.", style={'text

    # Creazione del grafico per la potenza del segnale
    ssids = [net['SSID'] for net in networks]
    signals = [net['Signal'] for net in networks]

    fig = go.Figure()
    fig.add_trace(go.Bar(
        x=ssids,
        y=signals,
        text=[f"{signal} dBm" for signal in signals],
        textposition='auto',
        marker_color='#4caf50',
    ))
    fig.update_layout(
        title="Wi-Fi Signal Strength",
        xaxis_title="SSID",
        yaxis_title="Signal Strength (dBm)",
        yaxis=dict(autorange='reversed'),  # Invertire l'asse Y
        template="plotly_dark",
    )

    return html.Div([
        html.H3("Available Networks:", style={'color': '#ffffff'}),
        dcc.Graph(figure=fig)
    ])
```

Callback per aggiornare i risultati del test di velocità:

```python
# Callback per il test di velocità
@app.callback(
    [Output("speedtest-results", "children"), Output("speed-gauge", "figure")],
    Input("run-speedtest-button", "n_clicks")
)
def display_speed_test(n_clicks):
    if not n_clicks:
        return html.P("Click 'Run Speed Test' to see results."), go.Figure()

    time.sleep(2)  # Simulazione del caricamento
    ping, download_speed, upload_speed = run_speed_test()
    results_text = html.Div([
        html.P(f"Ping: {ping:.2f} ms", style={'color': '#ffffff'}),
        html.P(f"Download Speed: {download_speed:.2f} Mbps", style={'color': '#ffffff'}),
        html.P(f"Upload Speed: {upload_speed:.2f} Mbps", style={'color': '#ffffff'})
    ])

    # Grafico gauge per la velocità di download
    gauge_figure = go.Figure(go.Indicator(
        mode="gauge+number",
        value=download_speed,
        title={'text': "Download Speed (Mbps)"},
        gauge={
            'axis': {'range': [0, 200], 'tickwidth': 1, 'tickcolor': "#ffffff"},
            'bar': {'color': "#4caf50"},
            'steps': [
                {'range': [0, 50], 'color': "#f44336"},
                {'range': [50, 100], 'color': "#ffeb3b"},
                {'range': [100, 150], 'color': "#4caf50"},
                {'range': [150, 200], 'color': "#2196f3"}
            ]
        }
    ))
    gauge_figure.update_layout(
        paper_bgcolor="#1e1e2f",
        font={'color': "#ffffff", 'family': "Arial"}
    )

    return results_text, gauge_figure

```

Esecuzione del Server

Come viene avviato il server:

```python
# Avvio del server Dash
if __name__ == "__main__":
    app.run_server(debug=True)

```

Schema della Struttura

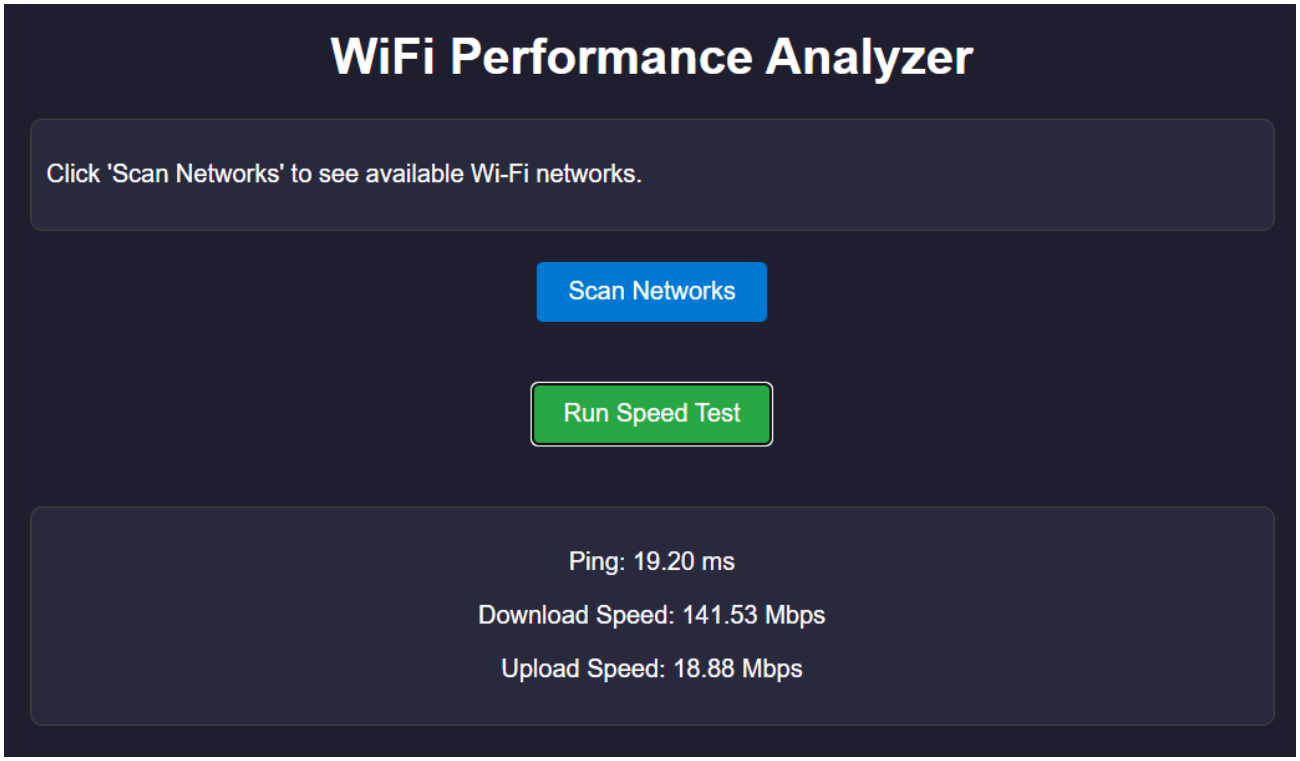
1. **Importazioni**:
    - Librerie Dash, Plotly, PyWiFi, Speedtest e Time.
2. **Inizializzazione**:
    - Configurazione dell'app Dash.
    - Layout HTML con stile CSS e componenti interattivi.
3. **Funzioni**:
    - scan_networks(): Scansiona le reti Wi-Fi.
    - run_speed_test(): Esegue il test di velocità Internet.
4. **Callback Dash**:
    - display_networks(n_clicks): Aggiorna i risultati della scansione delle reti.
    - display_speed_test(n_clicks): Mostra i risultati del test di velocità.
5. **Esecuzione del Server**:
    - Il server è avviato quando il file è eseguito direttamente

LAYOUT INTERFACCIA UTENTE

L'interfaccia utente è composta da due sezioni principali:

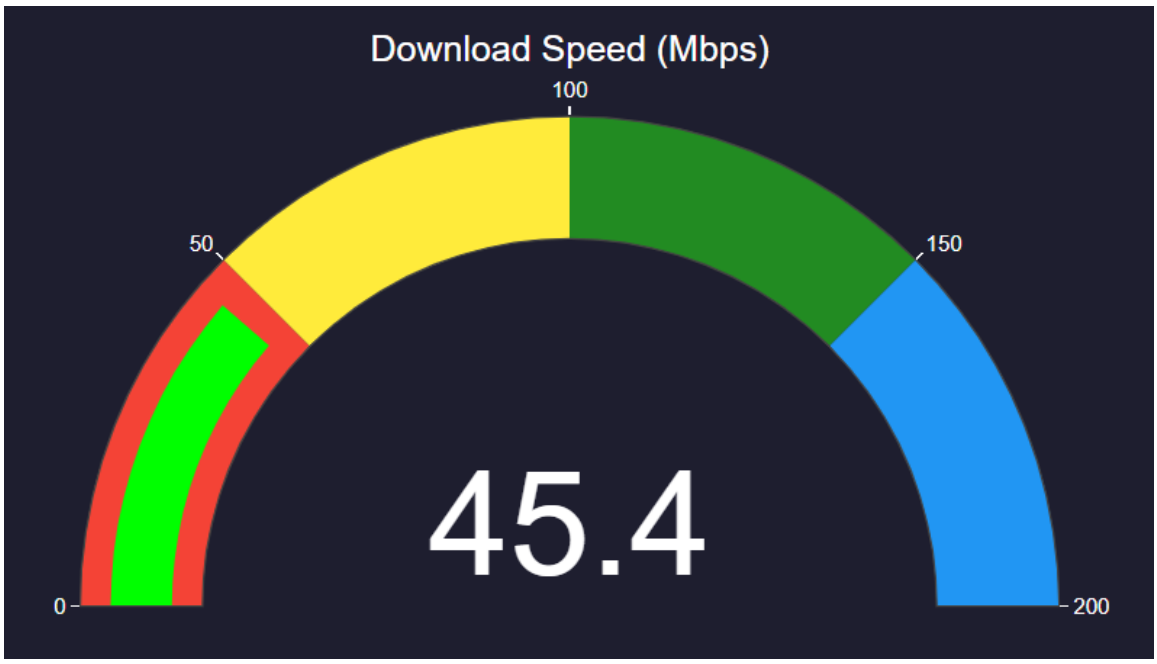
1. Scansione Reti Wi-Fi:

- Pulsante "Scan Networks" per avviare la scansione.
- Una finestra di caricamento (dcc.Loading) per mostrare l'avanzamento.
- Grafico dei risultati con SSID e potenza del segnale.



2. Test di Velocità:

- Pulsante "Run Speed Test".
- Risultati visualizzati in testo e gauge per la velocità di download.



Nel grafico soprastante si possono evidenziare le seguenti caratteristiche **(Velocità di download):**

1. **Valore:** Velocità di download (Mbps).
2. **Intervalli di colore:**

- Rosso: [0-50 Mbps].
- Giallo: [50-100 Mbps].
- Verde: [100-150 Mbps].
- Blu: [150-200 Mbps]
- Verde lime: [intervallo che corrisponde alla velocità effettiva del Wi-Fi]

Funzionalità principali del codice

Il file wifi_speed_test.py implementa diverse funzionalità per analizzare la performance dei Wi-Fi e della connessione Internet.

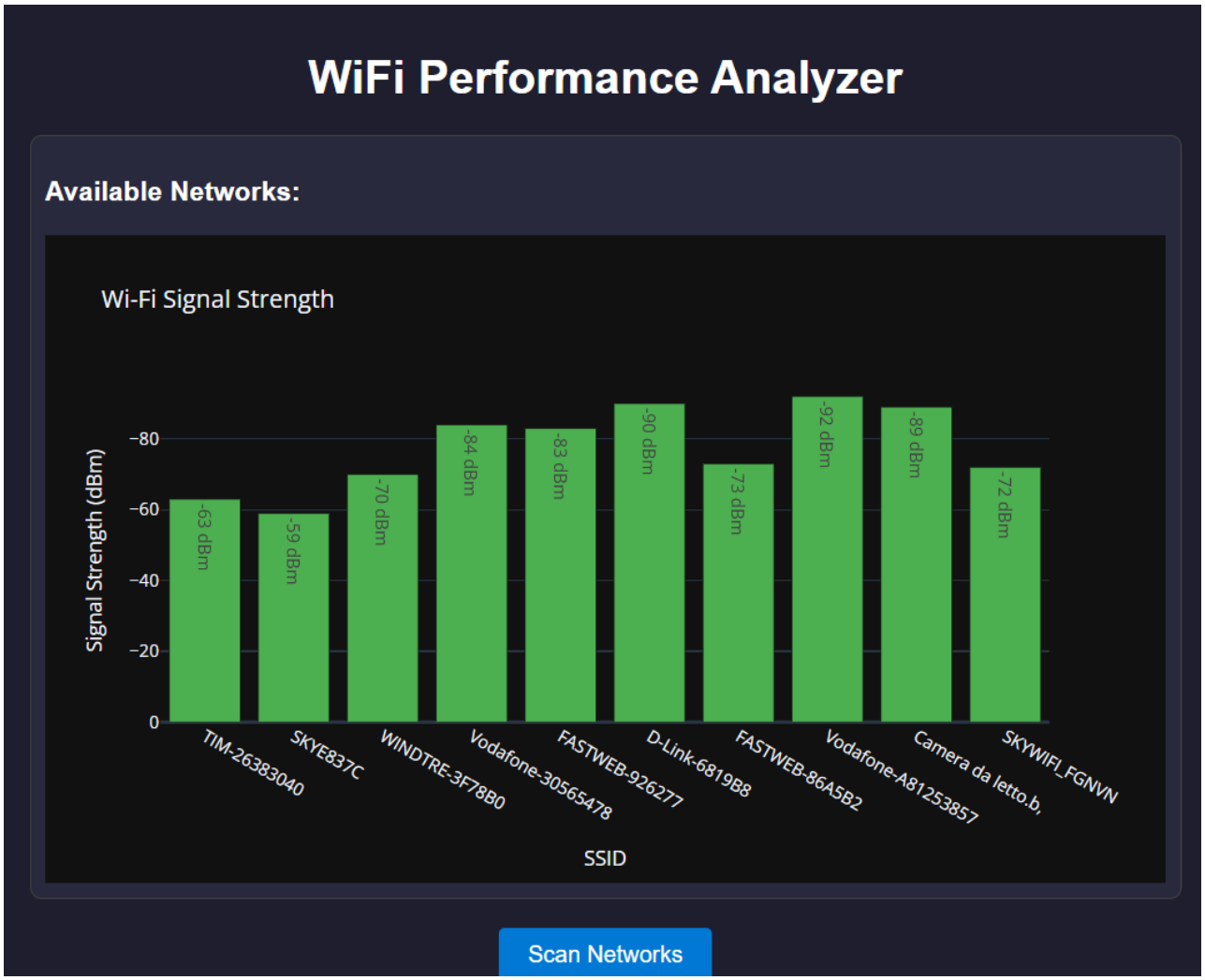
Di seguito viene mostrato un dettaglio delle principali funzionalità implementate:

Funzionalità	Dettaglio	Componente Interfaccia	Output Utente
Scansione delle Reti Wi-Fi	Analizza reti disponibili e forza del segnale	Pulsante "Scan Networks"	Grafico a barre con potenza del segnale
Test di Velocità Internet	Misura ping, download, upload	Pulsante "Run Speed Test"	Testo e grafico gauge interattivo
Interfaccia Intuitiva	Tema scuro, layout responsive	Layout Dash	Stile moderno e facile da usare
Feedback Visivo	Messaggi e animazioni di caricamento	`dcc.Loading`	Esperienza utente migliorata
Grafici Dinamici	Barre e gauge per rappresentazioni visive	Grafici Plotly	Visualizzazione chiara e interattiva

RISULTATI

Scansione Reti Wi-Fi

Quando si esegue la scansione della rete Wi-Fi è possibili osservare un grafico a barre, il quale mostra le reti disponibili e la potenza del segnale. Dal grafico è possibile evincere che le reti più vicine avranno un segnale più forte (valori dBm più alti), come lo si può vedere nella Figura sottostante.



Precisazioni tecniche

I valori del segnale Wi-Fi vengono spesso espressi in numeri negativi perché si utilizza l'unità di misura chiamata dBm (decibel milliwatt), che rappresenta un modo standardizzato per misurare la potenza del segnale.

Il segnale Wi-Fi è una forma di onda elettromagnetica, e la sua potenza, quando ricevuta da un dispositivo (ad esempio uno smartphone o un computer), è molto debole. La potenza tipica di un segnale Wi-Fi è inferiore a 1 milliwatt, che si traduce in un valore negativo in dBm.

- In scala dBm, 0 dBm corrisponde esattamente a 1 milliwatt (mW).
- Per segnali più deboli, il valore dBm diventa negativo, perché la potenza è una frazione di 1 mW.

Esempi pratici:

- -30 dBm: Segnale molto forte (vicino al router).
- -50 dBm: Segnale buono.
- -70 dBm: Segnale accettabile per navigare.
- -90 dBm o inferiore: Segnale molto debole, probabilmente inutilizzabile.

Test di Velocità

Determina i risultati per **ping**, **download** e **upload**.

- Un indicatore gauge rappresenta la velocità di download
- Colori diversi indicano fasce di prestazioni.
- Asse X: SSID delle reti, **Service Set Identifier**, cioè il nome identificativo della **rete wireless WLAN**.
- Asse Y: Potenza del segnale (dBm, invertito).

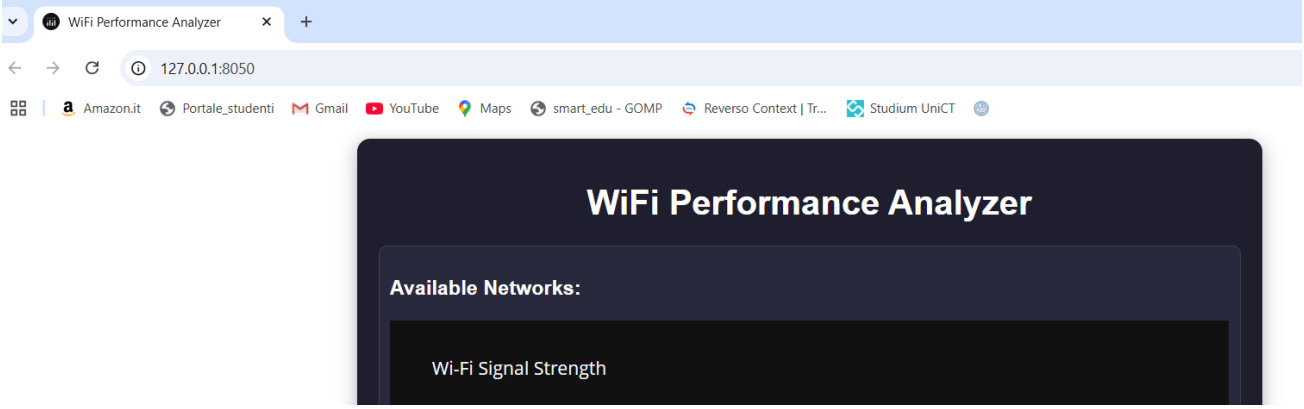
Esecuzione del Progetto

1. Installare i prerequisiti: `pip install dash plotly pywifi speedtest-cli`

2. Salvare il file come `wifi_speed_test.py`.

3. Eseguire il comando: python wifi_speed_test.py, si apre il browser su http://127.0.0.1:8050/

Questo è un indirizzo di rete locale, noto anche come "localhost". In pratica, punta al tuo computer o dispositivo locale, in cui è in esecuzione un'applicazione o un server web sulla porta 8050.



Questo tipo di link è utilizzato da sviluppatori per testare applicazioni web in fase di sviluppo. La porta 8050 è una porta di rete comunemente utilizzata per applicazioni web, specialmente quelle sviluppate con il framework **Dash**. Dash è un framework

open source utilizzato per creare applicazioni web interattive, in particolare quelle per visualizzazioni di dati.

Debug e Risoluzione dei Problemi

Problemi Comuni

- Errore: Nessun adattatore Wi-Fi rilevato:
  o Bisogna assicurarsi che il dispositivo abbia un adattatore Wi-Fi attivo.
- Scansione senza risultati:
  o Attendere qualche secondo e riprovare.
- Test di velocità lento o fallito:
  o Verificare la connessione a Internet.

Possibili sviluppi futuri

L'applicativo fornisce funzionalità base che ci si aspetterebbe da uno speed test moderno, quali ping, presenza di reti nella zona con relativo segnale e ovviamente lo strumento per misurare la velocità di rete in download e upload. In futuro l'applicativo potrebbe essere aggiornato con ulteriori funzionalità quali un sistema di cronologia di speed test precedenti, una dashboard personalizzabile e la geolocalizzazione del server a cui ci si sta connettendo. Inoltre si potrebbe prevedere un supporto multilinguistico.

L'analisi delle prestazioni di una rete (ad esempio una rete informatica o una rete aziendale) può avere molti sbocchi futuri utili e strategici. Ecco alcune possibili direzioni e applicazioni:

1. Miglioramento dell'Infrastruttura:

• Ottimizzazione delle Risorse: Identificare i colli di bottiglia e allocare risorse in modo più efficiente, come maggiore larghezza di banda o hardware più performante.
• Progettazione Scalabile: Sviluppare infrastrutture più robuste che possono crescere con l'aumento della domanda, sia in termini di traffico che di carico.
• Implementazione di Tecnologie Avanzate: Valutare l'introduzione di tecnologie innovative, come reti SD-WAN, segmentazione della rete o virtualizzazione.

2. Miglioramento della Sicurezza:

• Identificazione delle Vulnerabilità: Le analisi possono rivelare punti deboli della rete che potrebbero essere sfruttati da attori malevoli.
• Migliore Monitoraggio del Traffico: Implementare sistemi di rilevamento delle intrusioni (IDS) e di prevenzione (IPS) basati sui dati di performance.
• Piani di Recupero: Creare procedure di disaster recovery o migliorare la resilienza della rete contro attacchi come DDoS.

3. Aumento dell'Efficienza Operativa:

- Automazione dei Processi: Utilizzare i dati delle prestazioni per implementare strumenti di automazione della gestione della rete (ad esempio configurazioni dinamiche o auto-riparazioni).
- Riduzione dei Tempi di Inattività: Migliorare i protocolli di manutenzione preventiva per minimizzare i downtime.
- Supporto al Monitoraggio Proattivo: Sviluppare dashboard intelligenti che allertano in tempo reale su anomalie o cali di performance.

4. Supporto alle Decisioni Aziendali:

- Analisi dei Costi: Stimare i costi associati al mantenimento o all'espansione della rete, facilitando decisioni informate.
- Pianificazione Strategica: Proporre investimenti mirati per supportare nuove iniziative aziendali, come il lancio di nuovi servizi o l'espansione geografica.
- Sviluppo di KPI: Creare indicatori di performance chiave che monitorano il successo degli interventi sulla rete.

5. Innovazione Tecnologica:

- Intelligenza Artificiale e Machine Learning: Utilizzare i dati raccolti per sviluppare modelli predittivi che anticipano problemi o prevedono trend futuri.
- Supporto alla Trasformazione Digitale: Adeguare la rete alle esigenze di innovazioni aziendali, come l'adozione di IoT, big data, o servizi cloud.
- Introduzione di Reti 5G: Valutare l'impatto e i benefici del passaggio a reti di nuova generazione.

6. Miglioramento dell'Esperienza Utente:

- Qualità del Servizio (QoS): Garantire una maggiore qualità del servizio per gli utenti, ad esempio migliorando la latenza, riducendo la perdita di pacchetti e aumentando la velocità di connessione.
- Personalizzazione dei Servizi: Offrire servizi basati sui profili di utilizzo o sulla domanda specifica.

7. Compliance e Reporting:

- Adempimento Normativo: Assicurarsi che la rete sia conforme a normative come GDPR, HIPAA, o ISO 27001.

- Audit e Certificazioni: Prepararsi meglio per audit di sicurezza o per ottenere certificazioni di qualità e sicurezza.

Un'analisi ben fatta delle prestazioni di una rete è uno strumento potente per guidare il cambiamento tecnologico e strategico, portando vantaggi a lungo termine sia dal punto di vista operativo che competitivo.

CONFIGURAZIONE

Visual Studio

Visual Studio è un editor di codice sorgente sviluppato da Microsoft per Windows, Linux e macOS. Include il supporto per debugging, un controllo per Git integrato, syntax highlighting, IntelliSense, snippet e refactoring del codice. Sono personalizzabili il tema dell'editor, le scorciatoie da tastiera e le preferenze, e permette di installare estensioni che aggiungono ulteriori funzionalità. È un software libero e gratuito per uso personale e commerciale, anche se la versione ufficiale è sotto una licenza proprietaria.

Visual Studio code è un editor di codici per lo sviluppo di applicazioni per tablet, smartphone e computer, oltre a siti e *servizi web*. La caratteristica principale del programma è la quantità di linguaggi informatici che riesce a supportare: C#, Visual Basic, F#, C ++, Python, Node.js e Html/Java, solo per citarne alcuni.

Il 18 novembre 2015 il codice sorgente di Visual Studio Code è stato rilasciato con licenza MIT e reso disponibile su GitHub, ed è stata anche annunciata la possibilità di installare estensioni per aggiungere nuove funzionalità all'editor. Il 14 aprile 2016, è stata rilasciata la versione 1.0 di Visual Studio Code.

Produzione

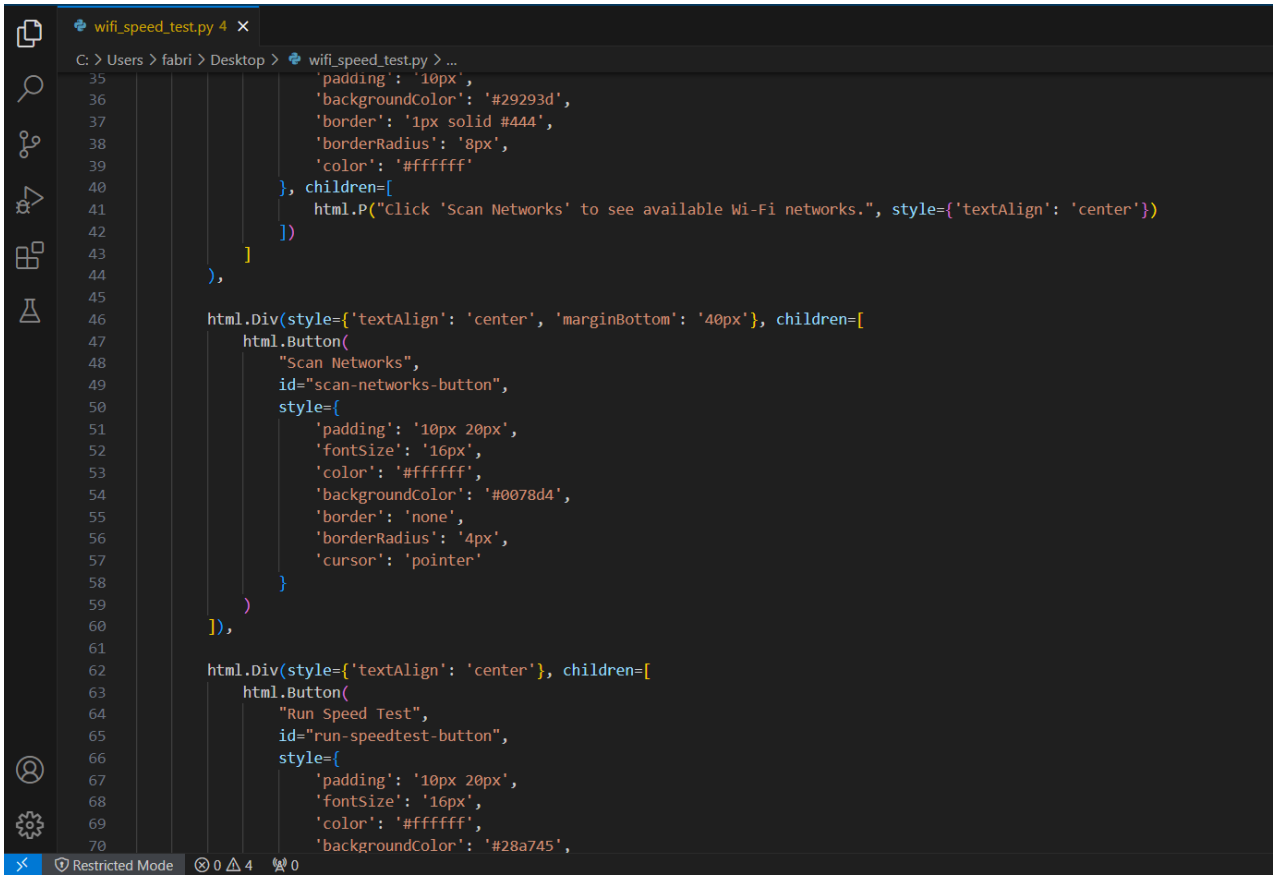
Lo script si presenta in questo modo sul proprio computer. La stesura del codice è stata realizzata tramite lo strumento offerto da Microsoft, ovvero Visual Studio. Nella seguente immagine è possibile visualizzare parte dello script python realizzato.

```
wifi_speed_test.py 4
C: > Users > fabri > Desktop > wifi_speed_test.py > ...
from dash import Dash, html, dcc, Input, Output
import plotly.graph_objects as go
import pywifi
import time
import speedtest


# Creazione dell'app Dash
app = Dash(__name__)
app.title = "WiFi Performance Analyzer"

# Layout dell'app
app.layout = html.Div(
    style={
        'fontFamily': 'Arial, sans-serif',
        'padding': '20px',
        'backgroundColor': '#1e1e2f',
        'maxWidth': '800px',
        'margin': 'auto',
        'boxShadow': '0 4px 16px rgba(0, 0, 0, 0.3)',
        'borderRadius': '12px'
    },
    children=[
        html.H1("WiFi Performance Analyzer", style={
            'textAlign': 'center',
            'color': '#ffffff'
        }),

        dcc.Loading(
            id="loading-scan-networks",
            type="circle",
            children=[
                html.Div(id="network-scan-results", style={
                    'marginBottom': '20px',
                    'padding': '10px',
```



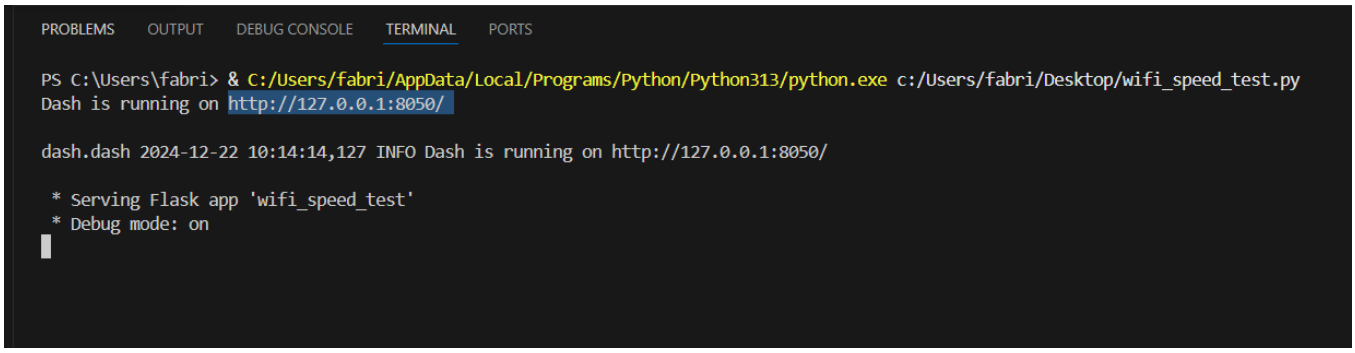
```
wifi_speed_test.py 4
C: > Users > fabri > Desktop > wifi_speed_test.py > ...
                    'padding': '10px',
                    'backgroundColor': '#29293d',
                    'border': '1px solid #444',
                    'borderRadius': '8px',
                    'color': '#ffffff'
                }, children=[
                    html.P("Click 'Scan Networks' to see available Wi-Fi networks.", style={'textAlign': 'center'})
                ])
            ]
        ),

        html.Div(style={'textAlign': 'center', 'marginBottom': '40px'}, children=[
            html.Button(
                "Scan Networks",
                id="scan-networks-button",
                style={
                    'padding': '10px 20px',
                    'fontSize': '16px',
                    'color': '#ffffff',
                    'backgroundColor': '#0078d4',
                    'border': 'none',
                    'borderRadius': '4px',
                    'cursor': 'pointer'
                }
            )
        ]),

        html.Div(style={'textAlign': 'center'}, children=[
            html.Button(
                "Run Speed Test",
                id="run-speedtest-button",
                style={
                    'padding': '10px 20px',
                    'fontSize': '16px',
                    'color': '#ffffff',
                    'backgroundColor': '#28a745',
Restricted Mode  0 4  0
```

Esecuzione

Per eseguire il tool bisogna, cliccare Start Debugging e successivamente Run python File, dopodiche apparirà una finestra come quella sotto in figura, dove basterà fare Ctrl+tasto sinistro del mouse sul link evidenziato e si aprirà la pagina di analisi della rete Wi-Fi.



Librerie

Al fine di runnare il programma nel migliore dei modi, si fa riferimento alle seguenti librerie: Dash, Plotly, PyWiFi, Speedtest e Time. Di seguito vediamo in dettaglio le varie librerie.

Dash: È una libreria per creare interfacce web interattive. È utile se desideri visualizzare in tempo reale i dati raccolti, ad esempio, per mostrare il risultato di un test di velocità di connessione o altre informazioni in un applicazione web.

Plotly: È una libreria per la creazione di grafici interattivi. Può essere usata insieme a Dash per visualizzare i dati raccolti, come ad esempio i risultati di test di velocità di rete, in forma grafica (grafici a linee, a barre, ecc.).

PyWiFi: Questa libreria ti permette di interagire con la rete Wi-Fi, come ottenere informazioni sulle reti Wi-Fi disponibili, connetterti a una rete, o anche eseguire scansioni per cercare nuove reti.

Speedtest-cli: Questa libreria permette di eseguire test di velocità della connessione Internet, restituendo il ping, la velocità di download e upload. Può essere utilizzata per raccogliere dati sulla qualità della connessione e per visualizzarli tramite Dash/Plotly.

Time: La libreria time è utile per gestire e formattare i tempi, come per esempio programmare la frequenza di esecuzione del test di velocità o visualizzare l'ora in cui è stato effettuato il test.