



**Politecnico
di Torino**

Dipartimento di Ingegneria Gestionale e della Produzione
Corso di Laurea Triennale in Ingegneria Gestionale
Classe L8 – Ingegneria dell'Informazione
A.A. 2023/2024

Sistema di Collaudo dei Dispositivi IoT per il Monitoraggio Veicoli

Relatore

Giuseppe Bruno Averta

Candidato

Marco Donatucci

Matricola

293556

Luglio 2024

SOMMARIO

1 - Proposta di progetto	4
1. Studente Proponente	4
2. Titolo della Proposta	4
3. Descrizione del problema proposto	4
1.4 Descrizione della rilevanza gestionale del problema.....	4
1.5 Descrizione del data-set per la valutazione	4
1.6 Descrizione preliminare degli algoritmi coinvolti.....	4
1.7 Descrizione preliminare delle funzionalità previste per l'applicazione software.....	5
2. – Descrizione dettagliata del problema	5
3. – Descrizione del dataset utilizzato.....	6
4. – Descrizione delle strutture dati.....	7
4.1 Struttura del progetto	7
4.1.1 Package Connection	7
4.1.2 Package Model	7
4.1.3 Package UI.....	7
4.1.4 main.py	7
5. - Descrizione degli algoritmi coinvolti.....	7
5.1 Login.....	8
5.2 Ricerca	8
5.3 Decodifica codice QR.....	10
6. – Alcune videate dell'applicazione e link al video YT.....	12
7. - Risultati sperimentali ottenuti.....	15
8. - Valutazioni sui risultati ottenuti e conclusioni	15

1 - Proposta di progetto

1. Studente Proponente

s293556 Marco Donatucci

2. Titolo della Proposta

Sistema di Collaudo dei Dispositivi IoT per il Monitoraggio Veicoli

3. Descrizione del problema proposto

Il progetto si propone di sviluppare un sistema per il monitoraggio e il collaudo di dispositivi IoT dell'azienda Topfly s.r.l. installati sui veicoli. Il problema affrontato riguarda la necessità di verificare e testare l'efficacia e l'affidabilità dei dispositivi IoT durante la fase di collaudo prima della loro messa in opera. Questo sistema consentirà di raccogliere e analizzare i dati provenienti dai sensori installati sui veicoli per identificare eventuali malfunzionamenti, ottimizzare le prestazioni e garantire che i dispositivi siano pronti per l'uso sul campo. La sfida principale è sviluppare un'interfaccia user-friendly che permetta ai tecnici di collaudo di eseguire test completi e ottenere feedback immediati sui dispositivi.

1.4 Descrizione della rilevanza gestionale del problema

Dal punto di vista gestionale, l'applicazione sviluppata rappresenta un significativo miglioramento nei processi operativi di Topfly s.r.l. L'ottimizzazione del collaudo e dell'installazione dei dispositivi riduce i tempi di inattività dei veicoli e migliora la qualità del servizio offerto ai clienti. La possibilità di eseguire controlli e comandi da remoto aumenta l'efficienza degli operatori, mentre la gestione centralizzata dei dati di collaudo permette una migliore tracciabilità e analisi delle performance. Inoltre, l'integrazione di funzionalità di contatto diretto con i tecnici e l'invio automatico di report via email semplifica la comunicazione interna e con i clienti.

1.5 Descrizione del data-set per la valutazione

I dati per la valutazione saranno ottenuti dai server aziendali, dove i sensori dei dispositivi inviano le loro informazioni durante la fase di collaudo. Questi dati includono misurazioni di posizione, temperatura, tempi di funzionamento, livelli di batteria, dati del tachigrafo e altre metriche operative. La piattaforma server dell'azienda fornirà l'accesso ai dati storici e in tempo reale tramite API. Questi dati saranno utilizzati per monitorare le performance dei dispositivi, rilevare anomalie e garantire che soddisfino gli standard richiesti prima di essere messi in servizio.

1.6 Descrizione preliminare degli algoritmi coinvolti

I principali problemi algoritmici da affrontare includono:

1. Autenticazione: Implementazione di un sistema di login sicuro utilizzando tecniche di hashing per le password e protocolli di autenticazione standardizzati;
2. Elaborazione delle immagini per il riconoscimento QR: Utilizzo di algoritmi per la scansione e la decodifica di codici QR, garantendo un'alta precisione e velocità di riconoscimento;
3. Ricerca e filtraggio dei dispositivi: Sviluppo di algoritmi di ricerca ottimizzati per la rapida identificazione dei dispositivi tramite il loro codice univoco;
4. Esecuzione di comandi remoti: Implementazione di algoritmi per l'invio e la gestione di comandi ai dispositivi, assicurando la corretta esecuzione e la gestione degli errori;
5. Generazione e invio di report via email: Automazione della raccolta dati e generazione di report, inclusi gli algoritmi per la formattazione e l'invio delle email.

1.7 Descrizione preliminare delle funzionalità previste per l'applicazione software

L'applicazione prevista permetterà ai tecnici di collaudo di monitorare e testare i dispositivi IoT in tempo reale. Le funzionalità principali includeranno:

- *Login*: Accesso all'applicazione tramite username e password, con autenticazione sicura;
- *Ricerca dispositivi*: Possibilità di cercare i dispositivi utilizzando un codice identificativo univoco o tramite la scansione di un codice QR presente nelle immagini della galleria;
- *Visualizzazione dati dispositivo*: Display dettagliato delle informazioni del dispositivo, inclusi i dati dei sensori in tempo reale;
- *Esecuzione comandi*: Interfaccia per eseguire comandi specifici sui dispositivi, come cambio del nome, blocco/sblocco del motore, controllo del tachigrafo e abilitazione del lettore;
- *Contatto tecnico*: Funzionalità per contattare un tecnico tramite telefonata diretta dall'applicazione in caso di problemi specifici;
- *Invio report via email*: Generazione automatica e invio di un'email all'azienda contenente tutti i dati del collaudo effettuato;

2. – Descrizione dettagliata del problema

2.1 Contesto operativo aziendale

Topfly s.r.l. si occupa dello sviluppo di hardware e software per il monitoraggio, controllo e gestione di veicoli da remoto. L'azienda necessita di semplificare l'installazione e il collaudo dei dispositivi di monitoraggio, migliorando l'efficienza operativa e riducendo i tempi di intervento. Nel contesto operativo dell'azienda, il problema affrontato riguarda il collaudo e la verifica dei dispositivi IoT installati sui veicoli, il sotto-problema specifico riguarda l'automatizzazione di questo processo, che tradizionalmente richiede molto tempo e comporta un alto rischio di errore umano.

La soluzione sviluppata consiste in un'applicazione che include la ricerca dei dispositivi tramite codice identificativo o QR, l'esecuzione di comandi remoti e la gestione delle comunicazioni tecniche.

2.2 Input

Dati provenienti dai sensori dei dispositivi IoT, come posizione GPS, livello di carburante, stato del motore, etc. Da parte dell'utente: Codice identificativo del dispositivo, immagini contenenti codici QR, comandi da eseguire.

2.3 Output

Report di collaudo inviato via email che indica il corretto funzionamento dei dispositivi ed eventuali anomalie riscontrate, informazioni del dispositivo, esito dei comandi eseguiti.

2.4 Criticità

Assicurare la precisione dei dati raccolti, sicurezza nella gestione dei dati, gestire grandi volumi di dati in tempo reale, garantire la sicurezza delle comunicazioni tra i dispositivi e il server, accuratezza nella decodifica dei codici QR, affidabilità dei comandi remoti, semplicità dell'interfaccia utente.

2.5 Potenzialità

Migliorare l'efficienza del processo di collaudo, riduzione dei tempi di intervento, riduzione dei costi operativi, aumento della soddisfazione del cliente grazie a una maggiore affidabilità dei dispositivi, semplificazione del tracciamento dei dati di collaudo.

2.6 Rilevanza

La soluzione proposta migliora significativamente i processi operativi di Topfly s.r.l., aumentando l'efficienza e la qualità del servizio offerto.

3. – Descrizione del dataset utilizzato

I dati utilizzati provengono direttamente dai server operativi aziendali, dove vengono raccolti continuamente dai sensori installati sui tutti i veicoli presenti nel database.

L'accesso ai dati è protetto da una procedura di autenticazione tramite token di accesso, è possibile ottenere un token per accedere al servizio tramite il login con username e password di un account registrato nel sistema aziendale e munito dei privilegi adatti.

L'accesso al server è garantito tramite l'implementazione di un api per le applicazioni python da loro fornita (<https://github.com/wialon/python-wialon.git>), e può essere gestito seguendo la relativa documentazione (<https://sdk.wialon.com/wiki/en/sidebar/remotepi/apiref/apiref>).

I dati utilizzati includono:

- Informazioni sull'account dell'utente
- Informazioni generali sul dispositivo in esame (nome, id, tipologia, ecc.)
- Posizione GPS (latitudine e longitudine)
- Livello di carburante
- Stato del motore (acceso/spento)
- Batteria
- Altri parametri specifici dei sensori collegati (Es. stato del sensore di blocco oppure temperatura dell'acqua)

I dati vengono trasmessi dai sensori tramite protocollo di comunicazione TCP e memorizzati nel server per essere poi elaborati e visualizzati nell'applicazione in tempo reale.

4. – Descrizione delle strutture dati

4.1 Struttura del progetto

Il progetto è stato realizzato seguendo il pattern MVC (Model View Controller).

Le directory che includono i file eseguiti dall'applicazione sono Connection, Model e UI.

4.1.1 Package Connection

Il package Connection contiene tutti i file relativi alla gestione dell'interazione con server, API e connessioni, all'interno del package sono presenti i seguenti file:

- connector.py: file che implementa la classe Connection, utilizzata per gestire tutte le richieste http e in generale le connessioni tramite API;
- device_DAO.py: file che si occupa dell'elaborazione dei dati ottenuti dalla classe Connection per trasformarli in strutture facilmente utilizzabili all'interno dell'applicazione.

4.1.2 Package Model

Il package Model contiene tutti i file relativi all'elaborazione dei dati ottenuti dalla connessione e anche file che definiscono la struttura degli oggetti utilizzati per memorizzare i dati, all'interno del package sono presenti i seguenti file:

- model.py: file che implementa tutti gli algoritmi di elaborazione dei dati e di logica dell'applicazione;
- device.py: per rappresentare il dispositivo in esame cercato dall'utente all'interno dell'applicazione si è scelto di utilizzare un oggetto di classe Device, questa non è altro che una dataclass utilizzata principalmente per memorizzare i dati relativi ai dispositivi e per semplici funzioni di visualizzazione come `__str__`;
- deviceN.py: i file deviceN (dove N va da 1 a 14, cioè il numero di tipi di dispositivi disponibili) definiscono classi device che estendono la super classe device descritta nel punto precedente, in queste classi vengono memorizzati dati specifici e tipici di ogni tipo di dispositivo.

4.1.3 Package UI

Il package UI contiene tutti i file relativi alla gestione dell'interfaccia grafica, all'interno del package sono presenti i seguenti file:

- controller.py: file utilizzato per gestire gli input dell'utente, effettuare controlli sulle stringhe, gestire la visualizzazione degli oggetti grafici, gestire i risultati degli algoritmi del model;
- view.py: file utilizzato per definire l'interfaccia grafica delle varie pagine dell'applicazione, è stata utilizzata la libreria Flet per l'implementazione della parte grafica dell'applicazione (<https://flet.dev/>).

4.1.4 main.py

Il file main.py è il punto di accesso all'applicazione, contiene la logica che permette alla libreria Flet di lanciare l'applicazione e costruire la pagina iniziale.

5. - Descrizione degli algoritmi coinvolti

Gli algoritmi utilizzati nel progetto che senza dubbio rivestono un ruolo fondamentale nel funzionamento dell'applicazione sono quelli relativi all'autenticazione, alla ricerca dei dispositivi, alla decodifica del codice QR e all'invio dei comandi remoti.

5.1 Login

```
marcodonatucci
def getTokenRequest(self, username, password):
    """Permette di effettuare il login tramite username e password"""
    url = 'https://hst-api.wialon.com/oauth/authorize.html?lang=en'
    data = {
        'response_type': 'token',
        'wialon_sdk_url': 'https://hst-api.wialon.com',
        'success_uri': '',
        'login_uri': 'https://tracking.topflyiot.com/login.html',
        'client_id': 'wialon',
        'redirect_uri': 'https://tracking.topflyiot.com/login.html',
        'access_type': '-1',
        'activation_time': '0',
        'duration': '0',
        'flags': '7',
        'login': username,
        'passw': password
    }
    headers = {
        'origin': 'https://tracking.topflyiot.com',
        'content-type': 'application/x-www-form-urlencoded',
        'refer': 'https://tracking.topflyiot.com/'
    }
    response = requests.post(url, data=data, headers=headers) # richiesta http post
    if response.status_code == 200:
        get_url = response.request.url
        # se la richiesta è andata a buon fine la risposta contiene un url di reindirizzamento con dentro un campo
        # token, il metodo setToken estrae il token e lo assegna al valore self
        self.setToken(get_url)
    else:
        return None
```

La procedura di login ha lo scopo di ottenere il token di accesso al server che servirà per autenticare qualsiasi operazione tramite un session id. Il token è generato al momento dell'autenticazione con username e password che vengono passati come parametri in un dizionario che rappresenta i dati della richiesta http. Tramite il modulo requests si effettua la richiesta e si ottiene la risposta che contiene anche il dato relativo al token, per estrarre il token si utilizza il metodo setToken che analizza la risposta del server e salva il token in una variabile self della classe Connection.

5.2 Ricerca

```
marcodonatucci
def getConnection(self, imei):
    """Stabilisce una connessione al server ed effettua la ricerca del dispositivo"""
    try:
        self.wialon_api = Wialon()
        result = self.wialon_api.token_login( # il login tramite token è necessario per ottenere il session id
            token=self.token)
        self.wialon_api.sid = result['eid'] # ottiene il session id che servirà ad autenticare qualsiasi richiesta

        search_spec = {
            'itemsType': 'avl_unit',
            'propName': 'sys_unique_id',
            'propValueMask': imei, # codice di identificazione univoco del dispositivo
            'sortType': 'sys_name'
        }

        search_result = self.wialon_api.search_items(spec=search_spec)
        if len(search_result['items']) == 0:
            return None
        item_data = search_result['items'][0]
        # ATTIVAZIONE UNITA!
        if item_data['act'] == 0:
            activation = self.wialon_api.active_unit(item_data['id'])
            # le unità disattivate vengono automaticamente
            # attivate durante la ricerca (su richiesta dell'azienda)
            item_data['act'] = activation['active']
        self.wialon_api.core_logout()
        return item_data # ottengo un dizionario con tutti i valori dei sensori dell'unità cercata
    except WialonError as e:
        print(e)
```


Per la ricerca viene utilizzato un metodo fornito direttamente dall'api del server che permette di inviare una richiesta specificando i parametri di ricerca (tra cui il codice identificativo univoco del dispositivo passato come parametro nel dizionario).

La procedura di ricerca, su richiesta dell'azienda, implementa anche l'attivazione del dispositivo cercato nel caso in cui sia esistente ma disattivato.

```
1 marcodonatucci
def getDevice(imei, selected_type, connection):
    """Crea gli oggetti Device e li include nel model"""
    imei = str(imei)
    connessione = connection.getConnection(imei) # ricerca del dispositivo tramite codice identificativo univoco
    if connessione is not None:
        # i tipi di dispositivi disponibili sono 14, con caratteristiche diverse, diversi tipi di comandi,
        # il metodo utilizza il costruttore della classe device relativa al numero del tipo di dispositivo
        if selected_type == '1' or selected_type == '2' or selected_type == '3' or selected_type == '4':
            try:
                result_device = device1234.Device1234(
                    uid=connessione.get('uid'),
                    itemId=connessione.get('id'),
                    name=connessione.get('nm'),
                    position=(
                        get_nested_value(connessione, keys: ['pos', 'x']),
                        get_nested_value(connessione, keys: ['pos', 'y'])
                    ),
                    device_status=connessione.get('act'),
                    object_status=get_nested_value(connessione, keys: ['prms', 'in', 'v']),
                    selected_type=selected_type,
                    battery=get_nested_value(connessione, keys: ['prms', 'pwr_ext', 'v']),
                    total_km=get_nested_value(connessione, keys: ['prms', 'mileage', 'v']),
                    blocco=get_nested_value(connessione, keys: ['prms', 'relay', 'v'])
                )
                return result_device
            except KeyError:
                return None
    elif selected_type == '5':
        try:
```

La risposta ottenuta dal server in seguito alla richiesta è in formato JSON, quindi occorre utilizzare dei metodi di parsing tramite librerie esterne per convertire il formato in una struttura dati a dizionario in cui ogni parametro registrato dai sensori del dispositivo ha un nome a cui è associato un ulteriore dizionario di valori letti dal sensore stesso. Accedendo ai giusti indici del dizionario creato si possono estrarre i dati relativi al dispositivo cercato e questi dati vengono memorizzati all'interno delle classi device relative al tipo di dispositivo in questione. L'oggetto device cercato viene impostato come variabile self della classe model in modo da poterlo elaborare con facilità.

5.3 Decodifica codice QR

```

└─ marcodonatucci
def read_qr_code(self, path):
    """Metodo per la lettura del qr code a partire da un file immagine"""
    # Apro l'immagine dal percorso usando Pillow
    image = Image.open(path)

    # Converto l'immagine nel formato utilizzato da opencv (BGR) utilizzando un array numpy
    open_cv_image = cv2.cvtColor(np.array(image), cv2.COLOR_RGB2BGR)

    # Uso la classe QRCodeDetector per rilevare e decodificare il QR code
    qr_detector = cv2.QRCodeDetector()
    data, points, _ = qr_detector.detectAndDecode(open_cv_image)

    # Se vengono rilevati i punti in cui è presente un codice qr viene restituito l'imei estratto
    if points is not None:
        return self.extract_imei(data)
    else:
        return None

└─ marcodonatucci
@staticmethod
def extract_imei(decoded_results):
    """Metodo per la ricerca del codice identificativo univoco del dispositivo (IMEI) nel testo di output del server"""
    imei_match = re.search(pattern=r'IMEI:(\d+)', decoded_results) # Cerca il campo IMEI
    if imei_match:
        return imei_match.group(1) # Restituisce solo il numero IMEI
    return 'IMEI non riconosciuto!'

```

I dispositivi IoT presentano un adesivo con codice QR che contiene alcuni dati relativi all'identificazione del dispositivo compreso il codice identificativo univoco.

Attraverso un oggetto FilePicker del modulo Flet relativo alla grafica è possibile selezionare un'immagine dall'archivio di sistema, è quindi possibile caricare un'immagine contenente uno dei codici QR citati poco fa.

L'algoritmo di decodifica del codice presente nell'immagine si avvale di tre moduli esterni che combinati risultano molto efficaci per le operazioni di elaborazione delle immagini: Pillow, numpy e opencv. Attraverso l'oggetto Image del pacchetto Pillow si riesce ad aprire l'immagine dal suo percorso in un formato facile da elaborare attraverso codice python, siccome la libreria opencv lavora con un sistema di colori BGR, utilizzando un metodo della libreria stessa che elabora l'immagine rappresentata da un array numpy, si cambia il sistema di colori dell'immagine caricata da RGB a BRG.

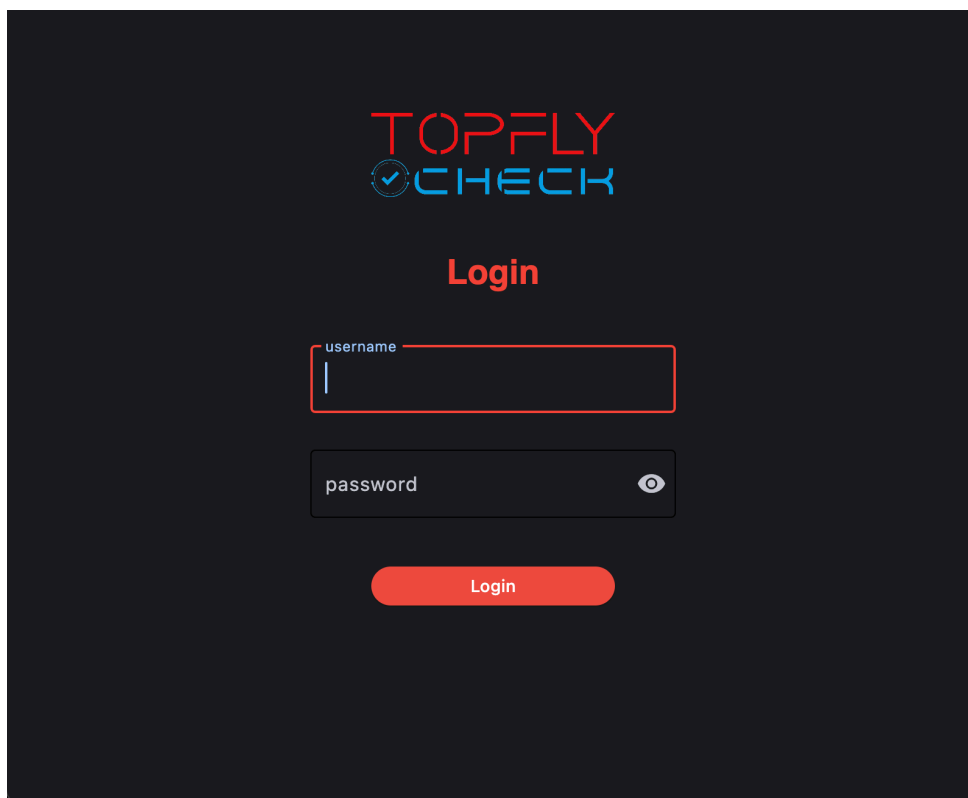
Viene quindi creato un oggetto QRCodeDetector per la scansione e la decodifica di possibili codici QR presenti nell'immagine, se il codice viene rilevato si utilizza un metodo per l'elaborazione di stringhe per estrarre il codice identificativo univoco dai dati contenuti nel codice.

5.4 Invio di comandi

```
marcodonatucci
def doTacho(self):
    """Metodo per l'invio del comando tacho check"""
    # definizione dei parametri (nomi e messaggi forniti dall'azienda)
    msg = {
        '9': 'tachocheck',
        '11': 'tachocheck',
        '13': 'AT+GTTR=gv355ceuv,10,,,,,,,,FFFF$',
    }
    name = {
        '9': 'check',
        '11': 'tacho',
        '13': 'tacho check'
    }
    itemId = self.device.itemId
    var_name = name.get(self.device.selected_type)
    var_msg = msg.get(self.device.selected_type)
    # esecuzione del comando con il metodo della classe connection
    result = self.connection.executeCommand(var_name, var_msg, itemId)
    if result:
        try:
            output = self.connection.getConnection(self.device.uid)['prms']['text']['v']
        except KeyError as e:
            output = True
        return output
    else:
        return False
```

La struttura dell'algoritmo di invio dei comandi remoti è comune per tutti i comandi possibili, vengono definiti i parametri da inserire all'interno della richiesta http come dizionari e viene quindi utilizzato il metodo fornito dall'api del server per inoltrare il comando. La risposta del server viene elaborata e visualizzata.

6. – Alcune videate dell'applicazione e link al video YT



The screenshot shows the login interface of the TOPFLY CHECK application. At the top, the logo "TOPFLY CHECK" is displayed in red and blue. Below it, the word "Login" is written in red. There are two input fields: "username" and "password". The "password" field has a red eye icon to toggle visibility. A red "Login" button is positioned below the fields.

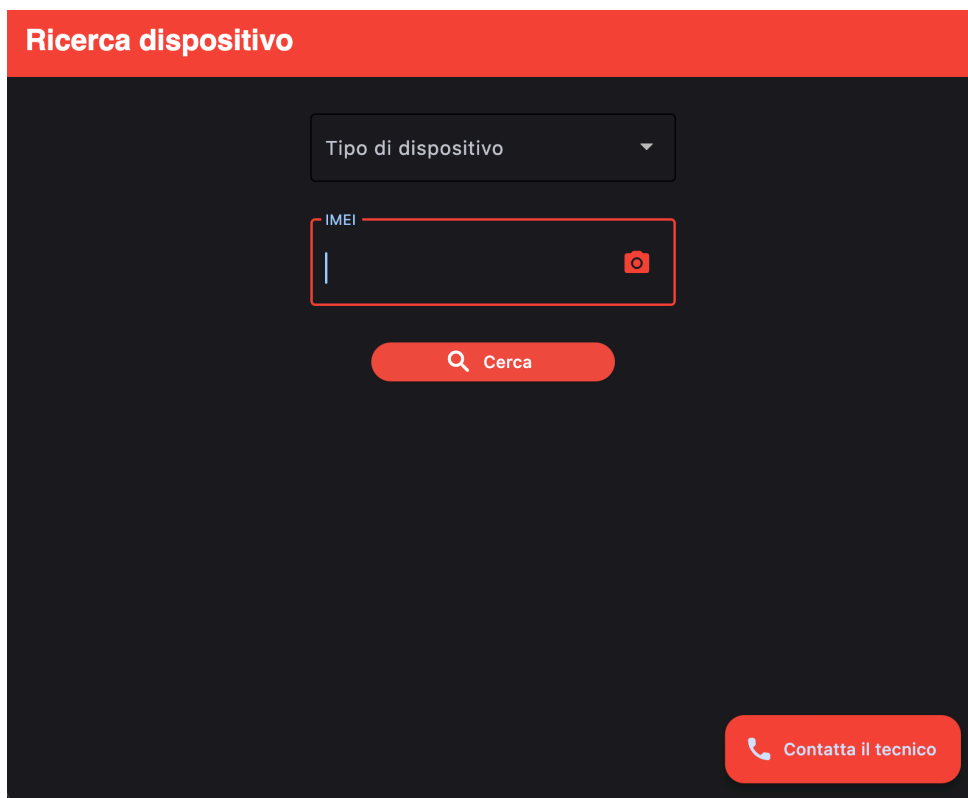
TOPFLY CHECK

Login

username

password

Login



The screenshot shows the "Ricerca dispositivo" (Device Search) screen. It has a red header with the title "Ricerca dispositivo". Below the header, there is a dropdown menu labeled "Tipo di dispositivo". Below that is an "IMEI" input field with a red camera icon. A red "Cerca" (Search) button with a magnifying glass icon is below the IMEI field. At the bottom right, there is a red button labeled "Contatta il tecnico" (Contact the technician) with a phone icon.

Ricerca dispositivo

Tipo di dispositivo

IMEI

Cerca

Contatta il tecnico

← Risultati della ricerca

↻

nome	testFMC150 Tipo
Status dispositivo	On
Posizione	(13.6388349, 43.4375816)
Accensione	On
Batteria	13.788V
Giri motore	3944rpm
Percentuale carburante	0%
Km totali	153335.0Km
Temperatura acqua motore	84.0°C
Sensore di blocco	Off

Hai bisogno di aiuto?
Contatta il tecnico!

☎

← Risultati della ricerca

↻

Comandi

Blocca motore

Sblocca motore

Abilita lettore

Disabilita lettore

Tacho check

Cambia nome

✓ Termina collaudo

Hai bisogno di aiuto?
Contatta il tecnico!

☎

← Termina collaudo

Note

Nome Cliente

✓ Termina collaudo

📞 Contatta il tecnico

Link al video dimostrativo: https://youtu.be/1W2-TwrOiPI?si=FSwgDIUK_MQH14QA

7. - Risultati sperimentali ottenuti

Si riporta di seguito una tabella con i tempi medi di esecuzione degli algoritmi più importanti e il tasso di successo, a causa della dipendenza dalla connessione internet la variabilità di questi risultati è molto alta, tuttavia è stato facile identificare un valore medio per ognuno dei seguenti punti

Algoritmo di autenticazione	~0.8s	95%
Algoritmo di ricerca	~1.12s	97%
Algoritmo invio comandi	~2s	90%
Algoritmo codice QR	~0.3s	90%
Algoritmo invio email	~1.6s	99%

8. - Valutazioni sui risultati ottenuti e conclusioni

8.1 Punti di forza

- *Automazione*: Riduzione significativa dei tempi di collaudo e installazione;
- *Efficienza*: Miglioramento dell'efficienza operativa grazie alla centralizzazione dei dati e alla possibilità di eseguire comandi remoti;
- *Usabilità*: Interfaccia intuitiva che semplifica l'interazione per gli operatori.

8.2 Punti di debolezza

- *Accuratezza della decodifica QR*: Sebbene elevata, potrebbe essere migliorata ulteriormente;
- Dipendenza dalla qualità della connessione di rete per la trasmissione dei dati;
- Necessità di manutenzione continua per aggiornamenti dei sensori e dell'applicazione.

8.3 Contributo rispetto al problema aziendale

Il progetto ha migliorato significativamente l'efficienza del collaudo dei dispositivi IoT, riducendo i tempi di verifica e aumentando l'affidabilità dei dati raccolti. Questo ha portato a una diminuzione dei costi operativi e a un miglioramento complessivo della qualità del servizio offerto ai clienti offrendo un notevole vantaggio competitivo all'azienda.

8.4 Conclusioni

Il progetto ha dimostrato l'importanza dell'automazione e dell'integrazione tecnologica nei processi di collaudo e installazione dei dispositivi. I risultati ottenuti sono stati positivi, con miglioramenti evidenti in termini di efficienza operativa e qualità del servizio. Ulteriori ottimizzazioni, soprattutto nella decodifica dei codici QR e nella gestione degli errori, potrebbero rendere l'applicazione ancora più robusta e affidabile.

Licenza

Questa relazione tecnica è distribuita con Licenza Creative Commons Attribuzione - Non commerciale - Condividi allo stesso modo 4.0 Internazionale.

Attribuzione - Non commerciale - Condividi allo stesso modo 4.0 Internazionale (CC BY-NC-SA 4.0)

Questo lavoro è distribuito con Licenza Creative Commons Attribuzione - Non commerciale - Condividi allo stesso modo 4.0 Internazionale.

Tu sei libero di:

- **Condividere** — riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare questo materiale con qualsiasi mezzo e formato.
- **Modificare** — remixare, trasformare il materiale e basarti su di esso.

Il licenziante non può revocare questi diritti fintanto che tu rispetti i termini della licenza.

Alle seguenti condizioni:

- **Attribuzione** — Devi attribuire adeguatamente l'opera, fornire un link alla licenza e indicare se sono state effettuate delle modifiche. Puoi farlo in qualsiasi modo ragionevole, ma non in alcun modo che suggerisca che il licenziante avalli te o il tuo utilizzo dell'opera.
- **Non commerciale** — Non puoi utilizzare il materiale per scopi commerciali.
- **Condividi allo stesso modo** — Se remixi, trasformi il materiale o ti basi su di esso, devi distribuire i tuoi contributi sotto la stessa licenza dell'originale.

Nota:

Non ci sono ulteriori restrizioni — Non puoi applicare termini legali o misure tecnologiche che limitino legalmente altri dal fare qualsiasi cosa la licenza permetta.

Per una copia della licenza completa, visita <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>.