UNIVERSITÀ DEL SALENTO

Facoltà di Ingegneria

Corso di Laurea in Ingegneria dell'Informazione

Documentazione di Progetto per il corso
Internet of Things

Smart Drive WebApp

Docente

Prof. Luigi Patrono

Tutor

Ing. Teodoro Montanaro

Studenti

Marco Longo

Priamo Tarantino

Sommario

Capitolo 1 – Introduzione e Contesto

La sicurezza stradale e l'efficienza nella guida dei veicoli sono temi di crescente importanza nel contesto delle moderne tecnologie di mobilità. Con l'incremento dell'adozione di sistemi di monitoraggio e telemetria, è diventato possibile raccogliere una quantità significativa di dati riguardanti il comportamento del veicolo e dello stile di guida.

Questo progetto si pone l'obiettivo di sviluppare un sistema integrato per l'acquisizione, l'analisi e la visualizzazione dei dati di guida, al fine di stimare lo stile di guida del conducente e migliorare la sicurezza e l'efficienza.

Capitolo 2 – Analisi dei requisiti

2.1 – Requisiti funzionali

Il sistema deve raccogliere dati in tempo reale dai sensori dello smartphone (accelerometro, giroscopio e GPS) tramite un'app dedicata e memorizzarli su un database MongoDB. Utilizzando un modello di machine learning Random Forest, il sistema classifica gli stili di guida (prudente, normale, sportivo, aggressivo) e stima il comportamento medio di guida per ogni utente. Deve calcolare anche la percentuale di rischio di incidenti e fornire feedback sugli stili di guida. Infine, è necessario implementare un sistema di autenticazione sicuro basato su JWT per garantire l'accesso alle funzionalità dell'applicazione.

2.2 – Requisiti non funzionali

Il sistema deve garantire la sicurezza dei dati in conformità con le normative sulla privacy, gestire efficacemente una grande quantità di dati in tempo reale e assicurare un funzionamento continuo e affidabile. Deve offrire un'interfaccia utente intuitiva, essere scalabile per un numero crescente di utenti e dati, e compatibile con vari dispositivi e sistemi operativi. Inoltre, è essenziale che il sistema mantenga alte prestazioni con latenza minima e preveda una manutenzione regolare per aggiornamenti e miglioramenti.

Capitolo 3 – Stato dell'Arte

Negli ultimi decenni, i sistemi di monitoraggio degli stili di guida sono evoluti notevolmente. Inizialmente, si basavano su sensori di accelerazione per rilevare eventi critici e migliorare la sicurezza stradale. Con il tempo, sono stati introdotti sensori più avanzati e

tecnologie elettroniche, come GPS e giroscopi, che hanno migliorato la precisione nella raccolta dei dati e l'integrazione con i sistemi di assistenza alla guida (ADAS).

Recentemente, l'intelligenza artificiale (AI) e il machine learning (ML) hanno rivoluzionato questi sistemi, permettendo l'analisi approfondita dei dati per identificare pattern complessi e fornire feedback personalizzati. Inoltre, nuove tecnologie come il monitoraggio delle emozioni del conducente e la comunicazione V2X hanno ulteriormente migliorato la rilevazione e la gestione dei comportamenti di guida, offrendo una sicurezza e un'efficienza maggiori.

Capitolo 4 – Tecnologie utilizzate

4.1 - Acquisizione dati: *Sensor Logger*

Sensor Logger [1][2] (Figura 4.1) è un'app mobile che raccoglie dati dai sensori dello smartphone, tra cui quelli d'interesse per il progetto quali: accelerometro, giroscopio e GPS. Questi dati vengono inviati al back-end in formato JSON tramite richieste HTTP POST. *Sensor Logger* permette di configurare la frequenza di acquisizione dei dati e l'indirizzo del server destinatario, rendendola una scelta ideale per il sistema di acquisizione dati. La flessibilità di *Sensor Logger* consente di adattare facilmente il sistema a diverse configurazioni di rete e requisiti di acquisizione.

Figura 4.1 - Logo app *Sensor Logger*.

4.2 - Back-End: Python

Python[3] (Figura 4.2) Python è il linguaggio di programmazione scelto per la realizzazione del back-end grazie alla sua sintassi chiara e alla sua flessibilità, che favoriscono uno sviluppo rapido e una facile manutenzione del codice. Utilizzando il framework Flask per il back-end, Python gestisce efficacemente le richieste HTTP e supporta una vasta gamma di librerie per l'elaborazione dei dati e il machine learning, adattandosi alle esigenze del progetto.

[1] https://play.google.com/store/apps/details?id=com.kelvin.sensorapp&hl=it&pli=1
[2] https://apps.apple.com/us/app/sensor-logger/id1531582925
[3] https://www.python.org/

Figura 4.2 - Logo Python.

4.3 – Database: MongoDB

MongoDB[4] (Figura 4.3) è un database NoSQL scelto per il progetto per la sua flessibilità e scalabilità. Utilizzando un modello basato su documenti JSON, MongoDB gestisce efficacemente dati non strutturati e semi-strutturati, offre potenti funzionalità di query e aggregazione, e consente una scalabilità orizzontale per gestire grandi volumi di dati e adattarsi alla crescita del sistema.

Figura 4.3 - Logo MongoDB.

4.3 – Front-End: Vue.js

Il front-end del sistema è sviluppato con Vue.js[5] (Figura 4.4), un framework JavaScript che consente di creare Single Page Applications (SPA) dinamiche e reattive. Vue.js permette di visualizzare i dati e i risultati dell'analisi in tempo reale, offrendo un'interfaccia utente chiara e intuitiva. Inoltre, supporta l'integrazione con librerie grafiche come *Chart.js*[6] e *D3.js*[7] per la creazione di visualizzazioni interattive dei dati.

Figura 4.4 - Logo Vue.js.

[4] https://www.mongodb.com/
[5] https://vuejs.org/
[6] https://www.chartjs.org/
[7] https://d3js.org/

Capitolo 5 – Progettazione del database

La progettazione del database per il sistema di rilevazione dello stile di guida è strutturata per garantire un'organizzazione efficace ed una gestione efficiente dei dati raccolti tramite smartphone. Il database impiegato è suddiviso in quattro collezioni principali: samples, session, test e user (Figura 5.1).

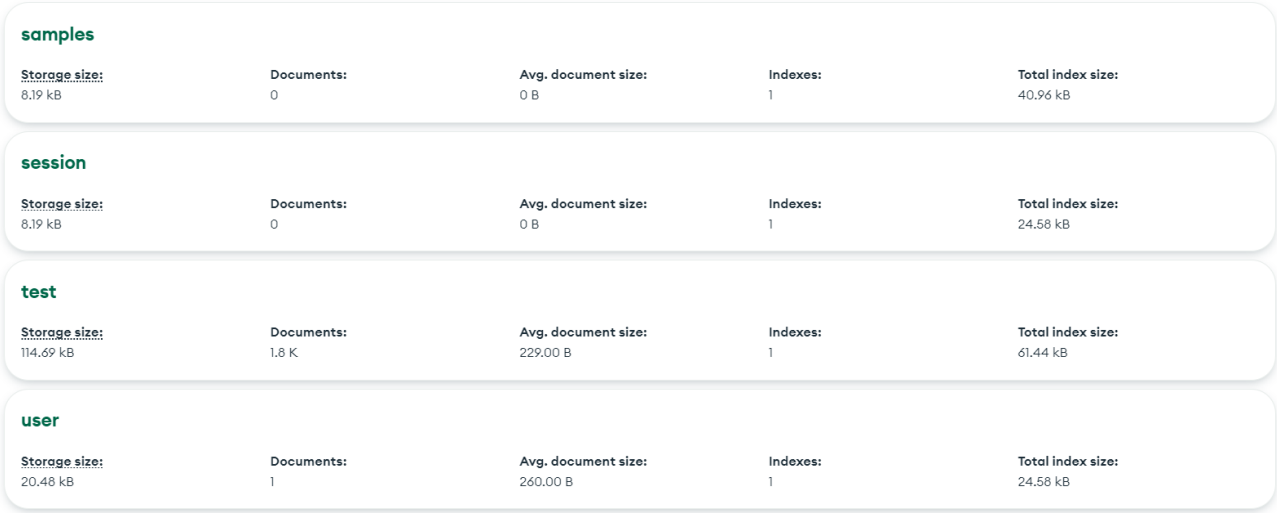

Figura 5.1 - Struttura del database.

5.1 – Struttura del database

Il sistema utilizza diverse collezioni per gestire i dati:

- Collection *Samples*: Contiene i dati sensoriali dei campioni di guida, associati a specifiche sessioni e utenti.
- Collection *Session*: Registra le sessioni di guida, tracciando l'utente e i metadati relativi.
- Collection *Test*: Memorizza i dati etichettati utilizzati per addestrare il modello di machine learning, separati dai dati delle sessioni in corso.
- Collection *User*: Conserva le informazioni di registrazione degli utenti, inclusi nome, email e password cifrata, oltre al Device ID unico del dispositivo.

Questa struttura facilita la gestione e l'analisi dei dati, migliorando la sicurezza e l'efficacia del sistema.

5.2 – Diagramma E-R

Il diagramma Entità-Relazione (Figura 5.2) rappresenta la struttura dati fondamentale del sistema, visualizzando le relazioni tra le entità principali ed i loro attributi. Questo

diagramma fornisce una panoramica chiara e dettagliata della struttura dati, facilitando la progettazione, la comprensione e l'ottimizzazione delle operazioni.

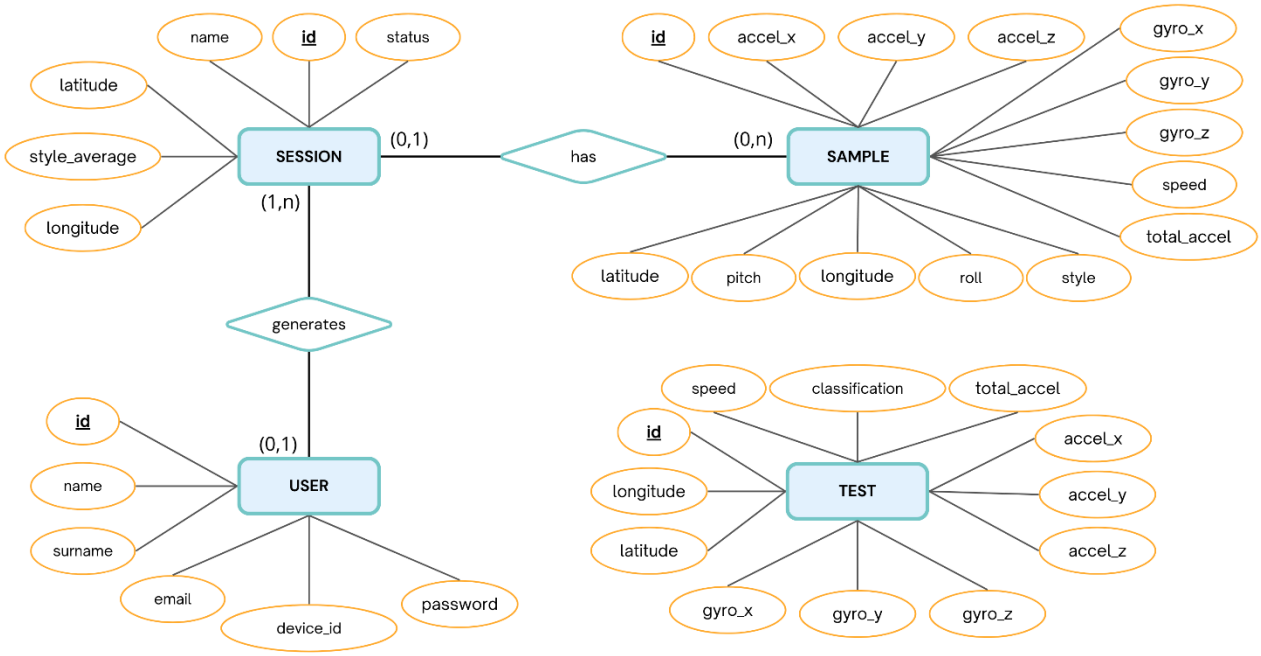

Figura 5.2 - Diagramma E-R.

5.6 – Diagramma Logico

Il diagramma logico (Figura 5.3) è basato sul fondamentale diagramma Entità-Relazione (ER). Questo diagramma traduce le relazioni e le entità identificate nel diagramma ER in una struttura più precisa e implementabile per il nostro ambiente di sviluppo. Ogni tabella nel diagramma logico rappresenta un'entità chiave del sistema, mentre le relazioni tra di esse sono rifinite ed ottimizzate per garantire l'efficienza delle operazioni e la coerenza dei dati.

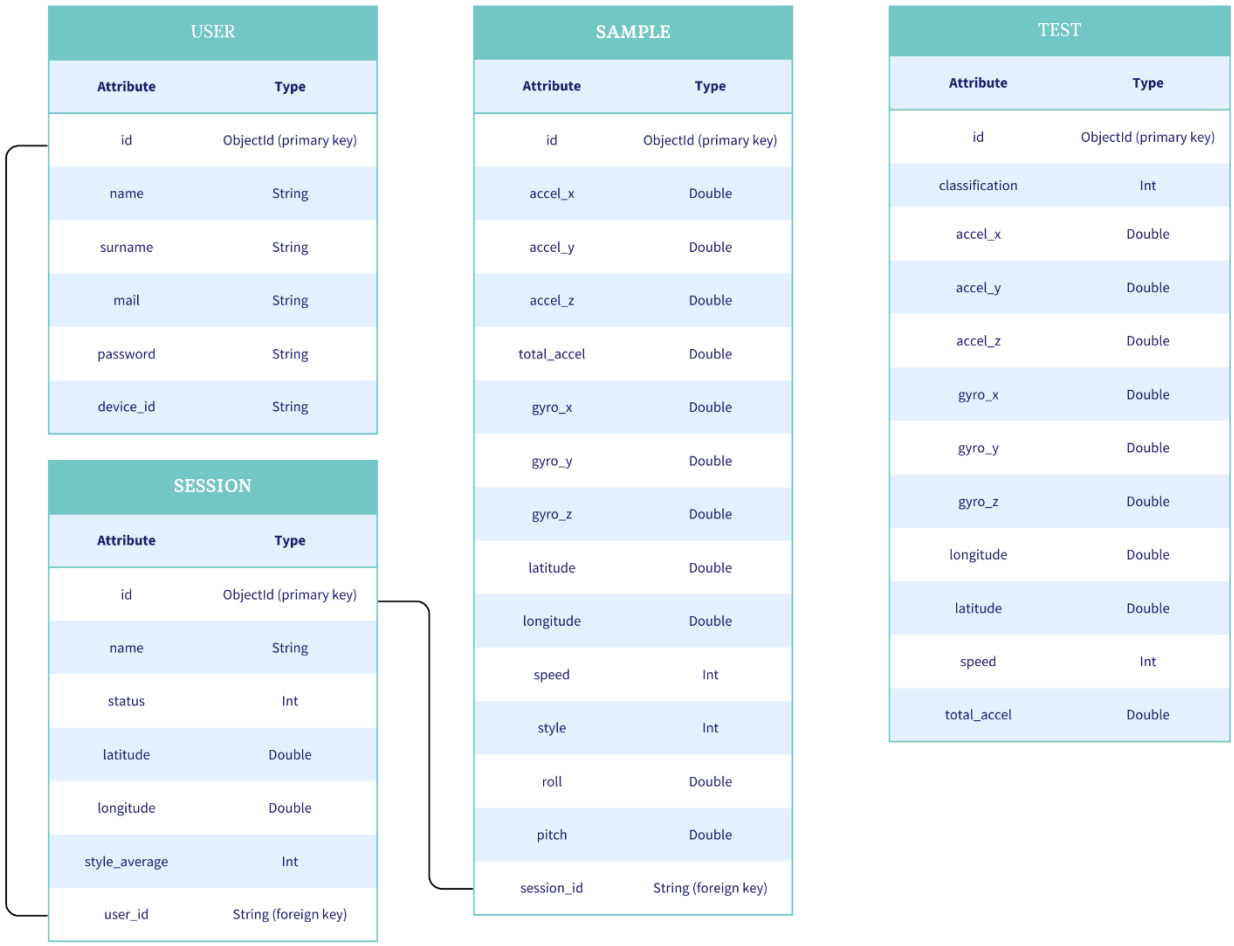

Figura 5.3 - Diagramma Logico.

Capitolo 6 – Soluzione proposta

6.1 – Architettura del sistema

Il processo inizia con la raccolta dei campioni tramite l'app *Sensor Logger*. Questa applicazione invia i campioni al server attraverso la rete. Il server elabora i dati, li ripulisce e li salva nel database MongoDB. Successivamente, il front-end può richiedere vari metodi tramite chiamate API. I dati vengono quindi recuperati dal database, elaborati ed adattati per l'utilizzo sul front-end. Di seguito è riportata l'architettura del sistema (Figura 6.1).

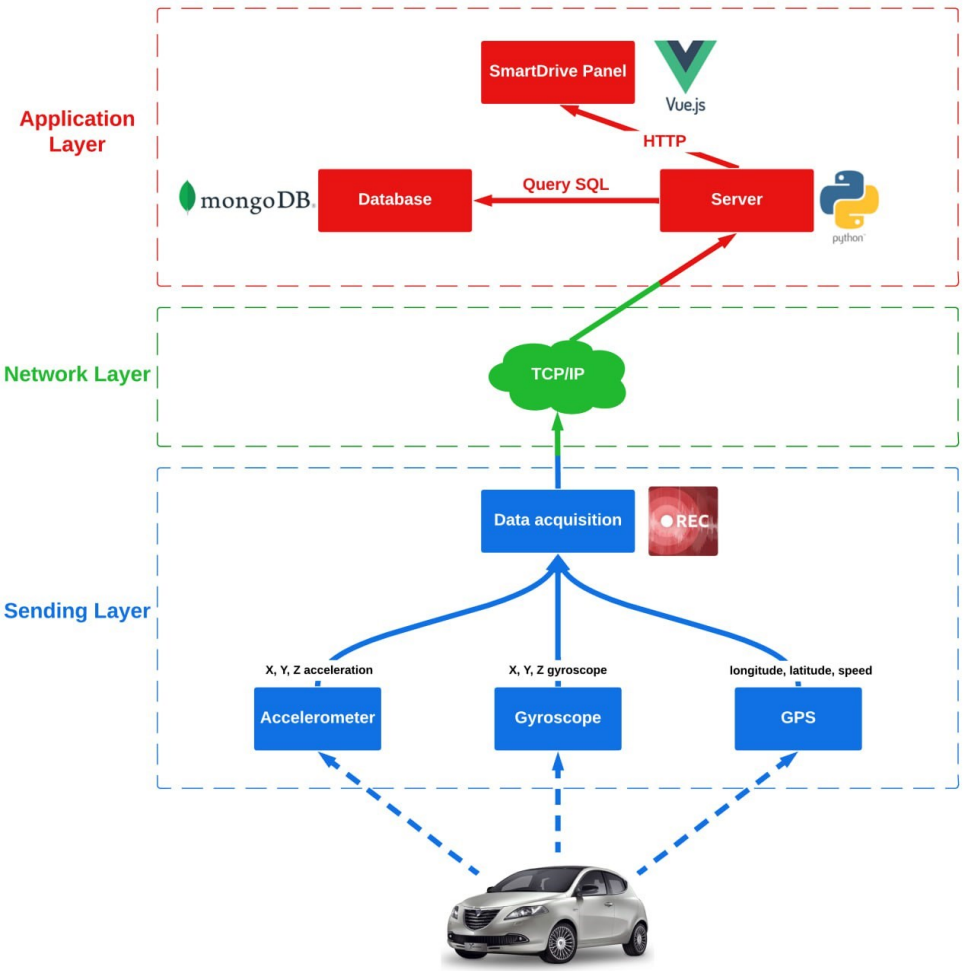

Figura 6.1 - Architettura del sistema.

6.2 - Raccolta dei Dati

Il primo passo cruciale nello sviluppo del sistema è stato l'acquisizione dei dati. Il team ha effettuato una raccolta sistematica di campioni di dati da diverse sessioni di guida, coprendo i vari stili. Questa fase ha richiesto un approccio metodologico per garantire la rappresentatività dei dati raccolti, comprendendo condizioni di guida normali, aggressive, sportive e prudenti.

Durante le sessioni di raccolta dati, l'app *Sensor Logger* ha catturato informazioni dettagliate sull'accelerazione e sulla velocità del veicolo. Questi dati grezzi sono stati successivamente pre-processati per eliminare eventuali anomalie o dati non validi, assicurando la qualità e l'affidabilità del dataset finale utilizzato per l'addestramento del modello.

6.3 – Grandezze osservate

Il rilevamento dello stile di guida si basa su diverse grandezze misurate dai sensori durante la guida. Queste includono:

- *Velocità*: Misurata in metri al secondo (m/s) tramite il delta delle coordinate GPS. Stili di guida aggressivi mostrano variazioni frequenti e rapide della velocità, mentre quelli tranquilli presentano variazioni più stabili.
- *Accelerazione (Modulo)*: Misurata in metri al secondo quadrato (m/s^2) e calcolata come modulo delle accelerazioni lungo gli assi X, Y e Z. Guida aggressiva mostra accelerazioni elevate e rapide, mentre la guida tranquilla ha accelerazioni più costanti e graduali.
- *Giroscopio*: Misura la velocità angolare in radianti al secondo (rad/s) lungo i tre assi principali. Guida aggressiva si caratterizza per elevate velocità angolari durante curve strette e brusche, mentre la guida tranquilla ha velocità angolari minori (Figura 6.2).
- *Rollio*: Angolo di inclinazione attorno all'asse longitudinale, misurato in radianti. Maggiore angolo di rollio indica curve strette e veloci, mentre minore angolo di rollio suggerisce curve più morbide.
- *Beccheggio*: Angolo di inclinazione attorno all'asse laterale, misurato in radianti. Maggiore beccheggio si osserva durante forti accelerazioni e frenate, mentre minore beccheggio si associa a cambiamenti più dolci.
- *Imbardata*: Angolo di rotazione attorno all'asse verticale, misurato in radianti. Maggiore imbardata si verifica durante cambi di direzione rapidi e improvvisi, mentre minore imbardata indica cambiamenti più graduali.

Queste grandezze aiutano a identificare e analizzare gli stili di guida, distinguendo tra comportamenti aggressivi e tranquilli.

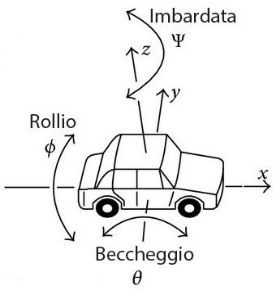

Figura 6.2 – Valori angolari del giroscopio.

6.4 - Preparazione dei Dati

Poiché non erano disponibili dataset preesistenti adatti, il team ha raccolto direttamente oltre 1800 campioni di dati per addestrare e testare il modello di machine learning. La preparazione dei dati ha incluso la pulizia e normalizzazione dei dati grezzi, l'ingegneria delle features per estrarre variabili significative, e la selezione delle features per migliorare l'efficienza e la precisione del modello. I dati sono stati successivamente trasformati per

ottimizzare la loro distribuzione e infine suddivisi in set di addestramento e test per valutare le prestazioni del modello.

6.5 – Modello *Random Forest*

Il modello selezionato per il sistema di rilevazione dello stile di guida è il Random Forest, noto per la sua robustezza e versatilità. Questa tecnica di machine learning combina più alberi decisionali, ciascuno addestrato su un campione casuale del dataset, per migliorare la precisione e la stabilità delle previsioni. L'uso del bagging, che campiona i dati con sostituzione, aiuta a ridurre il rischio di overfitting e a gestire grandi volumi di dati con efficienza. Random Forest si distingue per la sua capacità di generalizzare su nuovi dati e di adattarsi bene ai dati rumorosi, rendendolo ideale per il monitoraggio continuo e in tempo reale degli stili di guida.

6.6 – Addestramento del modello

Il processo di addestramento del modello (Figura 6.3) inizia con la connessione al database MongoDB per estrarre i dati tramite pymongo. I dati vengono poi preparati, includendo le features come accelerazione totale e velocità, e il target che indica la categoria dello stile di guida. Dopo la preparazione, i dati vengono suddivisi in set di addestramento e test. Il modello, un Random Forest Classifier con 100 alberi decisionali, viene addestrato e ottimizzato per garantire un'alta accuratezza e una buona generalizzazione sui dati non visti.

```python
def train_model_mongodb():
    # Connessione a MongoDB
    client = pymongo.MongoClient('mongodb://localhost:27017/')
    db = client['SmartDrive']
    collection = db['test']

    # Estrazione dei dati dal database
    cursor = collection.find({})
    data = list(cursor)

    # Preparazione dei dati
    X = []
    y = []

    for entry in data:
        X.append([entry['total_acceleration'], entry['speed']])  # Features: total_acceleration e speed
        y.append(entry['classification'])  # Target: classification

    # Split dei dati in set di addestramento e test
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Creazione del modello Random Forest
    forest_model = RandomForestClassifier(n_estimators=100, random_state=42)

    # Addestramento del modello
    forest_model.fit(X_train, y_train)

    # Valutazione del modello con cross-validation
    scores = cross_val_score(forest_model, X_train, y_train, cv=5)
    print(f'Cross-Validation Scores: {scores}')
    print(f'Mean Cross-Validation Accuracy: {scores.mean()}')

    # Predizione sui dati di test
    y_pred = forest_model.predict(X_test)

    # Valutazione delle prestazioni del modello sui dati di test
    print(classification_report(y_test, y_pred))

    # Ritorna il modello addestrato
    return forest_model
```

Figura 6.3 - Metodo per l'addestramento del modello.

6.7 – REST API

Il back-end espone delle REST API (Figura 6.3) fondamentali per facilitare l'interazione fluida tra il database sottostante e le varie componenti del sistema. Queste API costituiscono un ponte essenziale che consente alle diverse parti dell'applicazione di comunicare in modo efficiente ed affidabile con il database, garantendo al contempo la sicurezza e l'integrità dei dati gestiti.

```
/api
    /data (POST - new_data)
    /session
        /find_all (GET - getAllSessions)
        /{session_id} (GET - getSession)
        /new_session (POST - newSession)
        /find_by_user (GET - getSessionsByUser)
        /activate/{id_session} (PATCH - startSession)
        /deactivate/{id_session} (PATCH - endSession)
        /style_average/{id_session} (PATCH - calculateStyleAverage)
        /delete/{id_session} (DELETE - deleteSession)
    /samples
        /find_all (GET - getAllSamples)
        /find_by_session/{id_session} (GET - getSamplesByIdSession)
        /find_by_id/{id_sample} (GET - getSampleById)
    /user
        /{id_user} (GET - findById)
        /new_user (POST - newUser)
        /login (POST - login)
        /find_all (GET - findAll)
        /style_average (GET - getStyleAverage)
        /get_session_statistics/{id_session} (GET - getSessionMetrics)
        /get_global_statistics (GET - getGlobalUserStats)
        /modify (PATCH - updateUser)
        /delete (DELETE - deleteUser)
```

Figura 6.4 - REST API.

Capitolo 7 - Descrizione dei casi d'uso

In questa sezione verrà affrontata la descrizione dei casi d'uso che sono stati implementati nel sistema. Di seguito è riportato il diagramma dei casi d'uso (Figura 7.1).

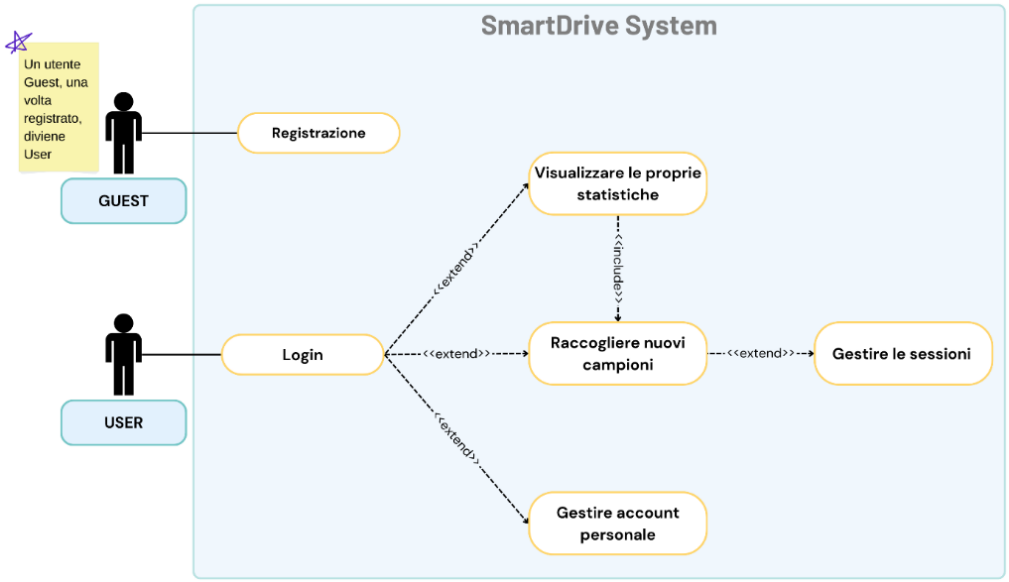

Figura 7.1 - Diagramma dei casi d'uso.

Caso d'uso: <u>Registrazione di un nuovo utente nel Sistema</u>
Il processo di registrazione consente a un utente guest di diventare un utente registrato (User). L'utente guest accede alla pagina di registrazione, inserisce i dati richiesti (nome, cognome, email, password e identificativo del dispositivo), e conferma la richiesta. Il sistema verifica i dati e, se validi, crea un nuovo account e fornisce una conferma di registrazione. In caso di dati incompleti o già esistenti, il sistema richiede correzioni o alternative.

Caso d'uso: <u>Login di uno User nel sistema</u>
Gli utenti registrati accedono al sistema inserendo le proprie credenziali (email e password). Se le credenziali sono corrette, il sistema genera un JSON Web Token (JWT) che consente l'accesso alle funzionalità del sistema. Se le credenziali sono errate o si verifica un errore di sistema, l'utente viene informato e deve riprovare.

Caso d'uso: <u>Avvio di una sessione e raccolta di campioni</u>
Gli utenti possono avviare una sessione di raccolta dati sia manualmente tramite il pannello utente del sistema, sia automaticamente dal proprio smartphone. Una volta avviata la sessione, i campioni raccolti vengono inviati al sistema e associati alla sessione attiva. In caso di problemi con l'identificativo del dispositivo o errori di sistema, il sistema gestisce l'errore e informa l'utente.

Caso d'suo: <u>Visualizzazione delle statistiche dell'utente</u>
Gli utenti autenticati possono visualizzare le statistiche relative allo stile di guida attraverso la Dashboard. Il sistema elabora e visualizza i dati raccolti in forma di grafici e dettagli. Se si verificano errori durante l'elaborazione, l'utente viene avvisato e l'errore viene registrato.

Caso d'suo: <u>Gestione del Profilo Utente</u>
Gli utenti possono gestire il proprio profilo, modificando i dati personali come nome e identificativo del dispositivo, o cancellare il proprio profilo. Le modifiche vengono applicate al database, e in caso di errore, il sistema avvisa l'utente. Se l'utente decide di annullare la cancellazione, il profilo rimane intatto.

Caso d'uso: <u>Gestione delle Sessioni</u>
Gli utenti possono gestire le proprie sessioni, modificando il nome o lo stato di attività, o cancellando una sessione e tutti i campioni associati. Le modifiche vengono applicate al database, e il sistema gestisce eventuali errori durante il processo.

Capitolo 8 – Gestione della sicurezza

Per gestire la sicurezza del sistema sono stati utilizzati due concetti chiave: crittografia e JWT.

8.1 – Crittografia

Per garantire la protezione delle credenziali degli utenti, il sistema adotta misure avanzate di sicurezza per la gestione delle password. Le password non vengono memorizzate in chiaro nel database. Invece, sono cifrate utilizzando l'algoritmo di hashing sha256 (Figura 8.1), che trasforma ogni password in una rappresentazione hash di 256 bit. Questo processo di cifratura impedisce l'accesso diretto alle password originali, anche in caso di compromissione del database, migliorando così la sicurezza delle informazioni sensibili e la privacy degli utenti.

```python
def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()
```

Figura 8.1 - Metodo di cifratura delle password.

8.2 – JSON Web Token (JWT)

Per la gestione delle sessioni e l'autenticazione degli utenti, il sistema utilizza i JSON Web Token (JWT). I JWT offrono un meccanismo standardizzato per la trasmissione sicura delle informazioni tra client e server. Ogni token JWT è composto da tre parti (Figura 8.2): l'header, che specifica l'algoritmo di firma; il payload, che include dati come l'email e l'ID dell'utente; e la signature, che garantisce l'integrità e l'autenticità del token tramite un algoritmo di hashing crittografico. Questo approccio elimina la necessità di certificati complessi e assicura comunicazioni sicure senza richiedere TLS (Transport Layer Security).

Figura 8.2 - Struttura di un JWT.

Capitolo 9 – Validazione funzionale e/o prestazionale

9.1 – Statistiche del codice

Il progetto è stato analizzato utilizzando il plugin Statistic per PyCharm, che ha fornito una panoramica dettagliata delle metriche basilari del codice sorgente (Figura 9.1). Questa analisi ha permesso di ottenere una visione chiara della dimensione e complessità del codice, facilitando la gestione e il miglioramento continuo del progetto. Le statistiche del codice offrono un'indicazione dello stato attuale del progetto e aiutano a identificare le aree che potrebbero necessitare di rifattorizzazione o miglioramento della qualità.

Extension	Count	Size SUM	Size MIN	Size MAX	Size AVG	Lines	Lines MIN	Lines MAX	Lines AVG	Lines CODE
docx (DOCX files)	2x	5.525kB	0kB	5.525kB	2.762kB	42180	1	42179	21090	41512
json (JSON files)	5x	919kB	0kB	680kB	183kB	42280	31	34015	8456	42280
md (MD files)	1x	19kB	19kB	19kB	19kB	732	732	732	732	522
parquet (PARQUET files)	1x	3.141kB	3.141kB	3.141kB	3.141kB	30135	30135	30135	30135	29791
py (PY files)	5x	51kB	1kB	37kB	10kB	1460	46	1084	292	802
txt (Text files)	1x	0kB	0kB	0kB	0kB	7	7	7	7	7
Total:	15x	9.657kB	3.163kB	9.404kB	6.118kB	116794	30952	108152	60712	

Figura 9.1 - Statistiche del back-end.

9.2- Valutazione del modello di machine learning

Durante la fase di addestramento, il modello di machine learning è stato valutato per determinare le sue prestazioni su dati non visti.

```
Cross-Validation Scores: [0.98269896 0.98961938 0.99307958 0.98615917 0.98958333]
Mean Cross-Validation Accuracy: 0.9882280853517876
              precision    recall  f1-score   support

           1       0.99      1.00      0.99       132
           2       0.98      0.98      0.98       112
           3       1.00      0.97      0.99        79
           4       1.00      1.00      1.00        38

    accuracy                           0.99       361
   macro avg       0.99      0.99      0.99       361
weighted avg       0.99      0.99      0.99       361
```

Figura 9.2 - Valutazione del modello di machine learning.

Sono stati considerati i seguenti aspetti (Figura 9.2):

1. **Cross-Validation Scores**: Utilizzando la tecnica di cross-validation, il modello è stato testato su cinque fold del set di addestramento, ottenendo punteggi di accuratezza molto elevati, con valori che variano tra il 98.27% e il 98.96%. Questo dimostra che il modello ha una solida capacità di generalizzazione.

2. **Media dell'Accuratezza di Cross-Validation**: La media dell'accuratezza ottenuta tramite cross-validation è stata di 0.9882, indicando un'eccellente performance del modello nel generalizzare su nuovi dati non visti durante l'addestramento.
3. **Classification Report**: Il report di classificazione fornisce dettagli sulle prestazioni del modello su un set di dati di test separato, includendo metriche come precisione, recall e F1-score, che offrono una panoramica completa della capacità del modello di effettuare predizioni corrette e bilanciare precisione e recall.

Capitolo 10 – Conclusioni e Sviluppi futuri

10.1 – Obiettivo del progetto

Il progetto ha mirato a sviluppare un sistema avanzato per valutare e classificare lo stile di guida degli utenti tramite un modello di machine learning. L'obiettivo principale era fornire un'analisi dettagliata del comportamento alla guida, con la categorizzazione degli stili come prudente, normale, sportivo e aggressivo. Questo è stato realizzato attraverso l'analisi dei dati raccolti durante le sessioni di guida e visualizzati in un pannello informativo integrato, che offre grafici dettagliati e metriche significative.

10.2 – Obiettivi raggiunti

Il progetto ha raggiunto i suoi obiettivi grazie a un processo strutturato di sviluppo e ottimizzazione del modello di machine learning. I test su strada hanno confermato l'efficacia del sistema nel distinguere e classificare correttamente diversi stili di guida. Le prove effettuate hanno dimostrato l'affidabilità delle predizioni del sistema, consolidando la sua validità e precisione.

10.3 – Sviluppi futuri

Per il futuro, il sistema prevede miglioramenti significativi, tra cui l'integrazione di dati aggiuntivi come il consumo di carburante, l'usura dei freni e informazioni ambientali come le condizioni meteorologiche e il traffico. Questi aggiornamenti potrebbero migliorare la precisione delle valutazioni e adattare il modello alle variabili ambientali e alle esigenze specifiche degli utenti.
Sono anche previsti sviluppi per estendere l'applicazione del sistema a contesti professionali, come aziende di trasporto e conducenti di veicoli pesanti, per migliorare la sicurezza e l'efficienza. Inoltre, l'adattamento del modello ai diversi tipi di veicoli e l'adozione di tecniche avanzate di machine learning, come le reti neurali profonde, potrebbero consentire una gestione più personalizzata e dinamica dello stile di guida.