



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	4
1.1 – Contesto ed Obiettivi.....	4
1.2 – Criticità e Soluzioni.....	4
Capitolo 2 – Analisi dei requisiti.....	5
2.1 – Requisiti funzionali.....	5
2.2 – Requisiti non funzionali.....	5
Capitolo 3 – Stato dell’Arte	7
3.1 – Sviluppo storico.....	7
3.1.1 - Le origini dei sistemi di monitoraggio della guida	7
3.1.2 - Avanzamenti nei sensori e nell'elettronica	7
3.1.3 - Intelligenza Artificiale e Machine Learning	8
3.1.4 - Tecnologie emergenti e innovazioni.....	8
Capitolo 4 – Tecnologie utilizzate	10
4.1 - Acquisizione dati: Sensor Logger.....	10
4.2 - Back-End: Python	10
4.3 – Database: MongoDB.....	11
4.3 – Front-End: Vue.js.....	11
Capitolo 5 – Progettazione del database.....	12
5.1 – Collection Samples	12
5.2 – Collection Session.....	12
5.3 – Collection Test.....	13
5.4 – Collection User.....	13
5.5 – Diagramma E-R	13
5.6 – Diagramma Logico	14
Capitolo 6 – Soluzione proposta	15
6.1 – Architettura del sistema	15
6.2 - Raccolta dei Dati	16
6.3 – Grandezze osservate	16
6.3.1 – Velocità.....	16
6.3.2 – Accelerazione (Modulo).....	16
6.3.3 – Giroscopio	17
6.3.4 – Rollio.....	17
6.3.5 – Beccheggio	17

6.3.6 – Imbardata	17
6.4 - Preparazione dei Dati.....	18
6.5 – Modello Random Forest	19
6.6 – Addestramento del modello	19
6.7 – REST API.....	20
Capitolo 7 - Descrizione dei casi d’uso	22
Capitolo 8 – Gestione della sicurezza	28
8.1 – Crittografia	28
8.2 – JSON Web Token (JWT)	28
Capitolo 9 – Validazione funzionale e/o prestazionale	30
9.1 – Statistiche del codice.....	30
9.2- Valutazione del modello di machine learning	30
Capitolo 10 – Conclusioni e Sviluppi futuri	32
10.1 – Obiettivo del progetto.....	32
10.2 – Obiettivi raggiunti	32
10.3 – Sviluppi futuri.....	32
BIBLIOGRAFIA	34

Capitolo 1 – Introduzione

1.1 – Contesto ed Obiettivi

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.

1.2 – Criticità e Soluzioni

La problematica principale risiede nella necessità di un sistema che possa raccogliere dati in tempo reale dai sensori degli smartphone e dei veicoli, analizzare tali dati per determinare lo stile di guida e presentare i risultati in modo chiaro e comprensibile. Attualmente, esistono diverse soluzioni commerciali ed accademiche per il monitoraggio dello stile di guida, ma molte di esse presentano limitazioni in termini di precisione, tempestività e capacità di integrare dati da diverse fonti.

La soluzione proposta prevede l'uso di un'applicazione mobile (disponibile sia per Android che per iOS), *Sensor Logger*, per raccogliere dati dai sensori degli smartphone, tra cui accelerometro, giroscopio e GPS. Questi dati vengono inviati ogni secondo ad un sistema back-end in Python, che li memorizza in un database MongoDB.

Utilizzando un dataset creato internamente dal team, mettendosi alla guida delle proprie auto e simulando ogni stile di guida e ogni possibile scenario, sarà sviluppato un modello di machine learning per riconoscere e stimare lo stile di guida. Il front-end sarà realizzato con una Single Page Application (SPA) in Vue.js per visualizzare i dati ed i risultati dell'analisi.

Capitolo 2 – Analisi dei requisiti

2.1 – Requisiti funzionali

- *Raccolta dei dati da smartphone*: Il sistema deve essere in grado di ricevere dati provenienti dall'accelerometro, giroscopio e GPS di uno smartphone attraverso un'applicazione dedicata scaricabile dallo store.
- *Salvataggio dei dati*: I dati ricevuti devono essere salvati in tempo reale su un database MongoDB per consentire l'accesso e l'analisi successiva.
- *Addestramento del modello di machine learning*: Utilizzare un dataset predefinito per addestrare un modello di machine learning *Random Forest*, che sarà responsabile della classificazione degli stili di guida.
- *Classificazione degli stili di guida*: Dopo aver addestrato il modello, il sistema deve classificare in tempo reale gli stili di guida degli utenti registrati (prudente, normale, sportivo, aggressivo) basandosi sui dati provenienti dallo smartphone.
- *Stima dello stile di guida medio*: Calcolare la media degli stili di guida dei campioni raccolti nel database per ogni utente, al fine di stimare lo stile di guida medio.
- *Calcolo della percentuale di rischio incidenti*: Utilizzare lo stile di guida medio calcolato per determinare la percentuale di rischio di incidenti, fornendo un feedback all'utente sul suo comportamento alla guida.
- *Autenticazione degli utenti*: Implementare un sistema di registrazione, login e autenticazione basato su JWT (JSON Web Token) per garantire l'accesso sicuro alle funzionalità dell'applicazione.

2.2 – Requisiti non funzionali

- *Sicurezza*: Assicurare che i dati raccolti dagli utenti siano protetti in conformità alle normative sulla privacy e che l'accesso alle informazioni sensibili sia limitato ai soli utenti autorizzati.
- *Prestazioni*: Garantire che il sistema sia in grado di gestire una grande mole di dati in tempo reale senza compromettere le prestazioni, sia per la raccolta che per l'elaborazione dei dati.
- *Affidabilità*: Il sistema deve essere robusto e garantire un funzionamento continuo e senza interruzioni, minimizzando il rischio di perdita di dati critici o di malfunzionamenti.
- *Usabilità*: Progettare un'interfaccia utente intuitiva e user-friendly per consentire agli utenti di navigare facilmente tra le varie funzionalità dell'applicazione.
- *Scalabilità*: Prevedere la possibilità di espandere il sistema per gestire un numero crescente di utenti e di dati senza compromettere le prestazioni o la sicurezza.

- *Compatibilità*: Assicurarsi che l'applicazione sia compatibile con una varietà di dispositivi smartphone e sistemi operativi per massimizzare la sua accessibilità e utilità per gli utenti.
- *Manutenzione*: Implementare una strategia di manutenzione regolare per garantire che il sistema rimanga aggiornato e funzionante nel tempo, con eventuali correzioni di bug e miglioramenti delle funzionalità.
- *Performance*: Il sistema deve rispondere rapidamente alle richieste del front-end, mantenendo una latenza minima per assicurare che i dati siano visualizzati in tempo reale.

Capitolo 3 – Stato dell'Arte

3.1 – Sviluppo storico

Negli ultimi decenni, i sistemi di rilevazione degli stili di guida si sono evoluti notevolmente, influenzando la sicurezza stradale e l'efficienza dei veicoli. Questi sistemi combinano sensori, algoritmi di analisi dei dati e intelligenza artificiale per monitorare e valutare il comportamento del conducente, fornendo informazioni utili per migliorare la guida e la gestione dei veicoli.

3.1.1 - Le origini dei sistemi di monitoraggio della guida

Negli anni '90, i primi sistemi di monitoraggio degli stili di guida si concentravano principalmente sulla sicurezza stradale utilizzando sensori di accelerazione e decelerazione. Questi sensori, come gli accelerometri, registravano rapidi cambiamenti di accelerazione lungo gli assi X, Y e Z del veicolo. La loro funzione principale era rilevare eventi critici come frenate improvvise o accelerazioni violente, segnalando potenziali situazioni di pericolo al conducente o attivando sistemi di assistenza alla guida. Questi primi tentativi rappresentarono un passo significativo verso l'integrazione di tecnologie sensoriali nel contesto automobilistico per migliorare la sicurezza e la reattività dei veicoli su strada[1][2].

3.1.2 - Avanzamenti nei sensori e nell'elettronica

Negli anni 2000, l'evoluzione dei sistemi di monitoraggio degli stili di guida è stata influenzata dall'introduzione di sensori più avanzati e dall'integrazione di elettronica sofisticata nei veicoli. Questo periodo ha segnato un passo importante verso una comprensione più dettagliata del comportamento del conducente e delle dinamiche della guida.

L'adozione di sensori di posizione ha consentito ai sistemi di monitorare con precisione la posizione e l'orientamento del veicolo rispetto all'ambiente circostante. Questo tipo di sensori, spesso basati su GPS, fornisce informazioni cruciali sulla velocità, l'accelerazione e la direzione del veicolo. In combinazione con giroscopi, questi sensori possono rilevare anche piccoli movimenti e variazioni nell'angolo di sterzata, migliorando la capacità dei sistemi di analizzare il comportamento di guida [1] [2].

L'introduzione di unità di controllo elettroniche (ECU) più sofisticate ha permesso una maggiore integrazione e gestione dei dati raccolti dai vari sensori del veicolo. Questo ha facilitato lo sviluppo di sistemi di assistenza alla guida avanzata (ADAS) che possono fornire feedback in tempo reale e avvisi al conducente, migliorando ulteriormente la sicurezza e l'efficienza della guida [1].

3.1.3 - Intelligenza Artificiale e Machine Learning

Recentemente, l'integrazione dell'intelligenza artificiale (AI) e del machine learning (ML) ha apportato cambiamenti radicali nei sistemi di rilevamento degli stili di guida a bordo delle automobili. Questa evoluzione ha permesso ai sistemi di non solo monitorare il comportamento del conducente e le condizioni della strada, ma anche di comprendere e anticipare pattern complessi di guida attraverso l'analisi avanzata dei dati.

L'AI si è dimostrata particolarmente efficace nel trattare grandi volumi di dati provenienti da sensori di vario tipo, come telecamere, sensori di posizione e giroscopi. Utilizzando algoritmi di machine learning, questi sistemi sono stati addestrati su dataset ricchi e diversificati, che comprendono informazioni su diversi stili di guida, condizioni stradali e variabili ambientali [1][3].

Uno dei principali vantaggi dell'intelligenza artificiale nei sistemi di rilevamento degli stili di guida è la capacità di identificare pattern nascosti e correlazioni non ovvie tra dati apparentemente disconnessi. Ad esempio, gli algoritmi possono apprendere e rilevare comportamenti di guida rischiosi come accelerazioni brusche, curve troppo strette o distanze di sicurezza non mantenute, combinando informazioni da diversi sensori per una valutazione più accurata del rischio [2].

Inoltre, l'intelligenza artificiale consente la personalizzazione e l'adattamento dei sistemi di monitoraggio in base al comportamento individuale del conducente. Gli algoritmi possono essere configurati per riconoscere abitudini di guida specifiche di ciascun utente e fornire feedback personalizzati per migliorare la sicurezza e l'efficienza. Recenti studi hanno mostrato come questi sistemi possano adattarsi dinamicamente ai cambiamenti nel comportamento del conducente, migliorando così l'efficacia degli ADAS (Advanced Driver Assistance Systems) [3].

3.1.4 - Tecnologie emergenti e innovazioni

Negli ultimi anni, sono emerse nuove tecnologie che hanno ulteriormente migliorato i sistemi di rilevazione degli stili di guida. Ad esempio, la fusione dei dati provenienti da diverse fonti sensoriali e l'uso di tecniche avanzate di elaborazione dei segnali hanno permesso una maggiore precisione nella rilevazione dei comportamenti di guida.

Rilevazione delle Emozioni del Conducente: Recentemente, è stato introdotto il monitoraggio delle emozioni del conducente attraverso sensori biometrici e tecniche di riconoscimento facciale. Questa tecnologia può rilevare stati emotivi come la stanchezza o lo stress, permettendo al sistema di adattare gli avvisi e gli interventi in tempo reale per prevenire incidenti causati da distrazioni o affaticamento [4].

Tecnologia V2X (Vehicle-to-Everything): La comunicazione V2X rappresenta un'altra importante innovazione. Questa tecnologia consente ai veicoli di comunicare tra loro e con le infrastrutture circostanti, migliorando la consapevolezza situazionale e permettendo una risposta più rapida agli eventi stradali. I sistemi di rilevazione degli stili

di guida possono sfruttare queste informazioni per prevedere comportamenti di guida pericolosi e suggerire azioni preventive [5].

Reti Neurali e Deep Learning: L'uso di reti neurali profonde (DNN) ha permesso lo sviluppo di modelli più complessi e accurati per l'analisi dei dati di guida. Questi modelli possono apprendere direttamente dai dati grezzi dei sensori, migliorando continuamente le loro capacità di rilevazione e predizione con l'aumento dei dati disponibili [6] .

Capitolo 4 – Tecnologie utilizzate

4.1 - Acquisizione dati: *Sensor Logger*

*Sensor Logger*¹² (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

Python³ (Figura 4.2) è il linguaggio di programmazione utilizzato nel progetto per lo sviluppo dell'applicazione e la gestione dei dati. Grazie alla sua sintassi chiara e leggibile, Python favorisce lo sviluppo rapido e la manutenzione del codice, supportando paradigmi di programmazione come l'orientamento agli oggetti e la programmazione funzionale. La vasta libreria standard ed una comunità attiva di sviluppatori ampliano le capacità del linguaggio, permettendo l'implementazione di funzionalità complesse e l'automazione dei processi attraverso il suo potente sistema di scripting.



Figura 4.2 - Logo Python.

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/>

Nel back-end del sistema, sviluppato interamente in Python, è utilizzato il framework Flask per gestire in modo efficiente le richieste HTTP. Python consente di sfruttare una vasta gamma di librerie per l'elaborazione dati e l'implementazione di algoritmi di machine learning, migliorando la capacità del sistema di adattarsi dinamicamente alle esigenze ed alle sfide del progetto.

4.3 – Database: MongoDB

MongoDB⁴ è un database NoSQL flessibile e scalabile, utilizzato nel progetto per gestire dati non strutturati e semi-strutturati. Contrariamente ai database relazionali tradizionali, MongoDB utilizza un modello di dati basato su documenti JSON, il che rende più semplice la gestione di dati complessi e la scalabilità orizzontale. Utilizzando MongoDB, è possibile memorizzare e interrogare dati in modo efficace, sfruttando le sue potenti funzionalità di query ed aggregazione. La sua architettura distribuita consente di gestire volumi elevati di dati e di scalare il sistema in modo flessibile secondo le esigenze di crescita.



Figura 4.3 - Logo MongoDB.

4.3 – Front-End: Vue.js

Il front-end del sistema è sviluppato con Vue.js⁵ (Figura 4.4), un framework JavaScript per la creazione di Single Page Applications (SPA). Vue.js offre un'architettura reattiva e componenti modulari, permettendo di costruire interfacce utente dinamiche e responsive. Utilizziamo Vue.js per visualizzare i dati raccolti ed i risultati dell'analisi in tempo reale, offrendo agli utenti una visualizzazione chiara ed intuitiva del loro stile di guida. Vue.js supporta inoltre l'integrazione con librerie grafiche come *Chart.js*⁶ e *D3.js*⁷, che permettono di creare visualizzazioni interattive e informative dei dati raccolti.



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).

samples				
Storage size: 8.19 kB	Documents: 0	Avg. document size: 0 B	Indexes: 1	Total index size: 40.96 kB
session				
Storage size: 8.19 kB	Documents: 0	Avg. document size: 0 B	Indexes: 1	Total index size: 24.58 kB
test				
Storage size: 114.69 kB	Documents: 1.8 K	Avg. document size: 229.00 B	Indexes: 1	Total index size: 61.44 kB
user				
Storage size: 20.48 kB	Documents: 1	Avg. document size: 260.00 B	Indexes: 1	Total index size: 24.58 kB

Figura 5.1 - Struttura del database.

5.1 – Collection *Samples*

La collection *samples* contiene tutti i dati provenienti dagli smartphone relativi ai campioni di guida. Ogni documento in questa collezione include informazioni dettagliate sui dati sensoriali raccolti durante una sessione di guida. I campioni sono associati ad una specifica sessione che permette di identificare l'utente. Questo garantisce che i dati siano sempre correlati all'utente corretto ed alla sessione appropriata.

5.2 – Collection *Session*

La collection *session* registra tutte le sessioni di guida avviate dagli utenti. Ogni documento in questa collezione contiene informazioni sulla sessione, l'utente che ha avviato la sessione ed altri metadati pertinenti. Questa struttura consente di tracciare facilmente tutte le sessioni di guida e di associare correttamente i campioni raccolti alle rispettive sessioni.

5.3 – Collection Test

La collection *test* è dedicata ai campioni utilizzati per addestrare il modello di machine learning. Qui sono memorizzati i dati di guida raccolti in precedenza che sono stati etichettati ed utilizzati per migliorare l'accuratezza del sistema di riconoscimento dello stile di guida. La separazione di questi dati in una collezione distinta facilita la gestione e l'aggiornamento del modello di machine learning senza interferire con i dati di sessione attuali.

5.4 – Collection User

La collection *user* contiene tutti i dati di registrazione degli utenti del servizio. Nome, cognome ed indirizzo email sono salvati in chiaro, mentre la password è cifrata per garantire una maggiore sicurezza oltre che per le procedure di registrazione e login. In fase di registrazione, viene richiesto all'utente di fornire anche il proprio *Device ID* (32 caratteri esadecimali divisi in 5 gruppi) che è un codice univoco associato al dispositivo.

5.5 – 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.

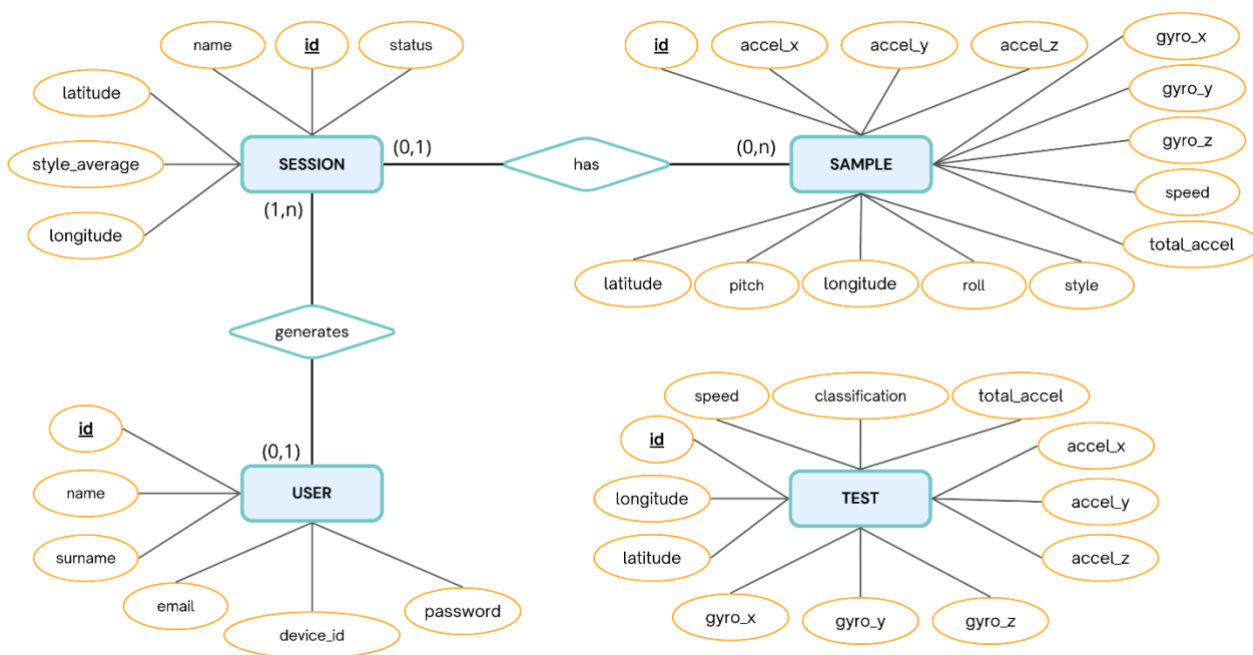


Figura 5.2 - Diagramma E-R.

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.

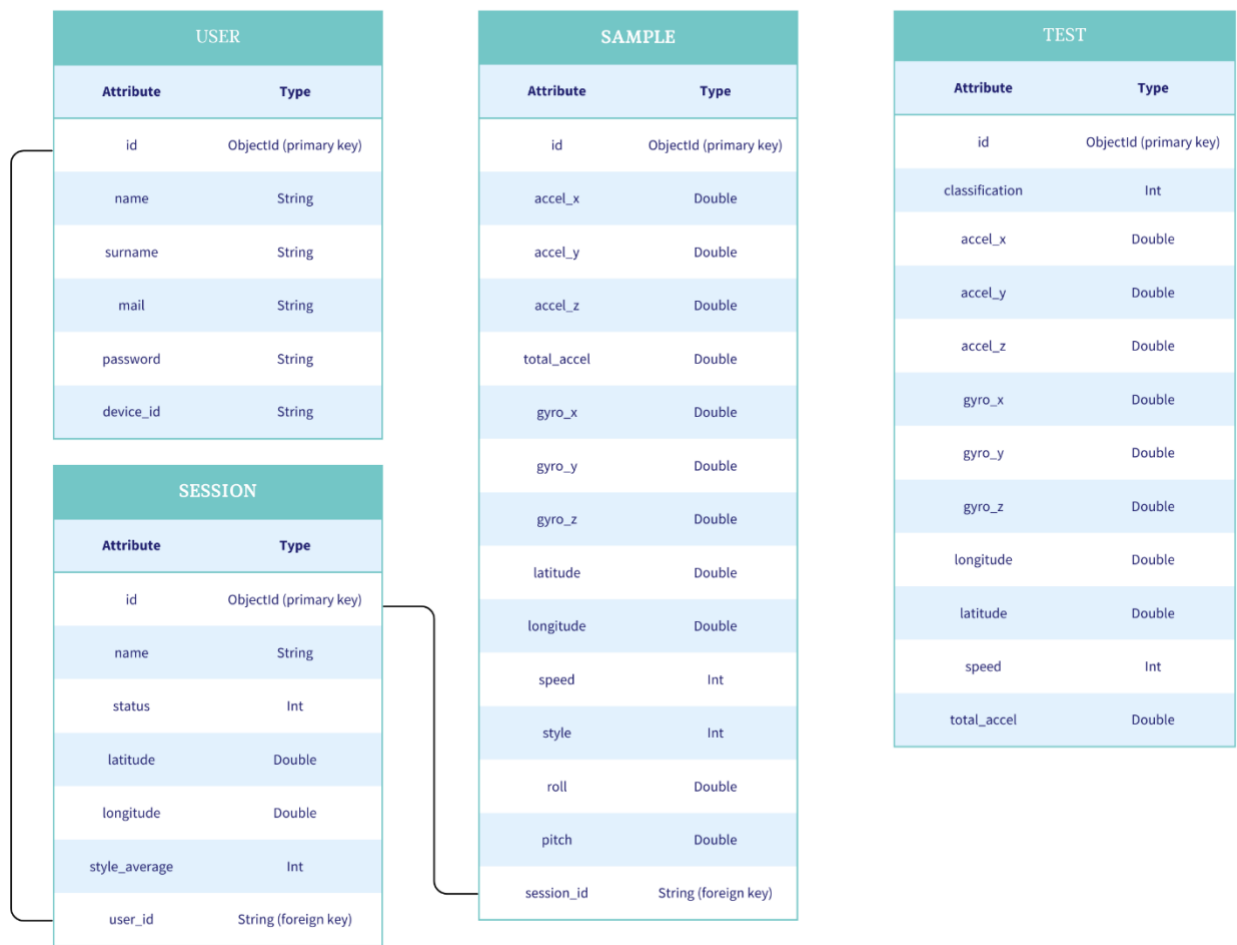


Figura 5.3 - Diagramma Logico.

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).

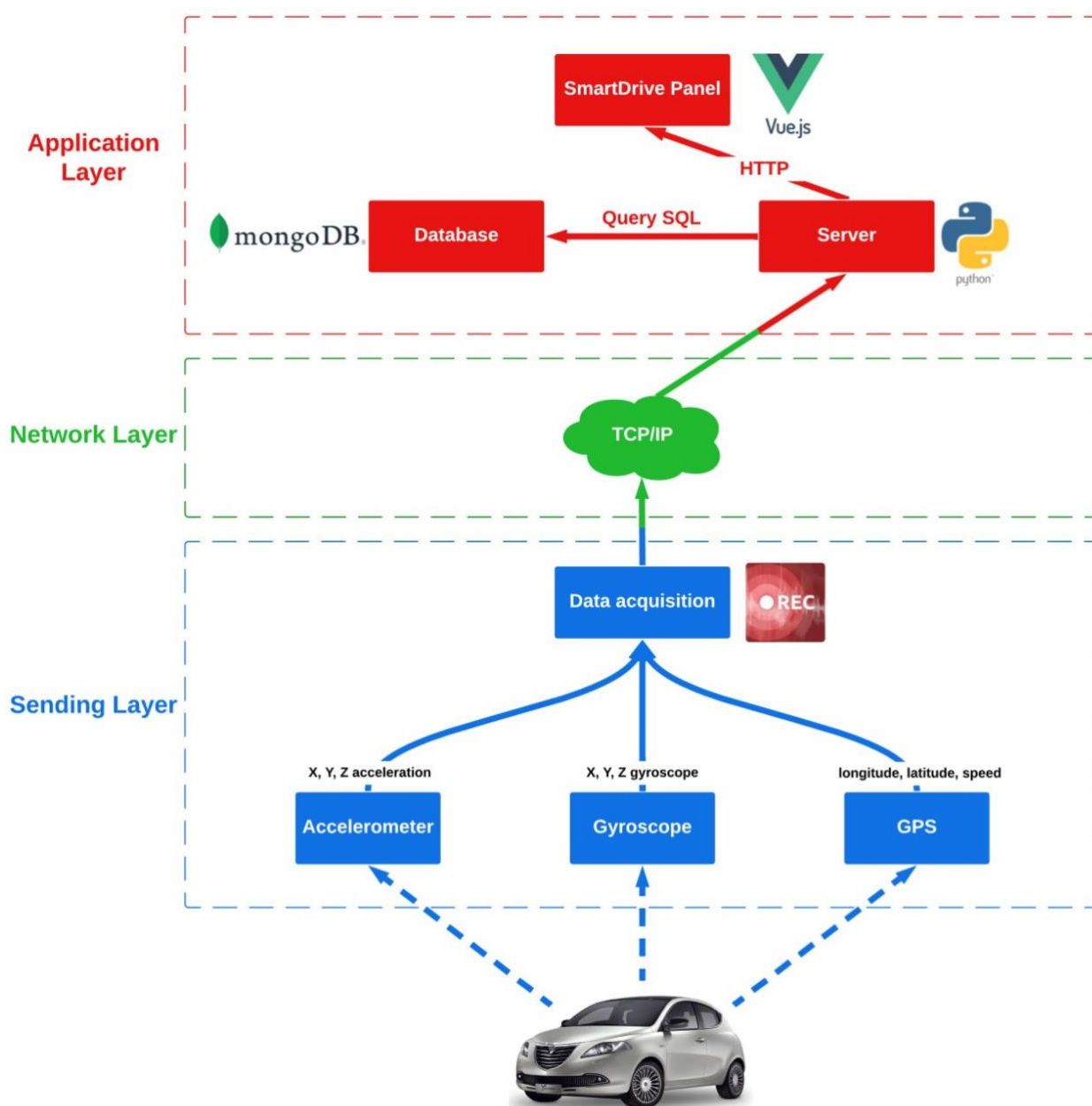


Figura 6.1 - Architettura del sistema.

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 è funzione di diverse grandezze rilevate durante la guida dai rispettivi sensori. Durante una sessione di guida, è importante tracciare velocità ed accelerazione, sia medie che istantanee. Stili di guida sportivi o spericolati registreranno picchi considerevoli oltre ad una maggiore varianza dei singoli valori rilevati. Sono altresì d'interesse le rilevazioni del giroscopio che tracciano le variazioni angolari del veicolo lungo i 3 assi (longitudinale, laterale, verticale) noti anche rispettivamente come *rollio*, *beccheggio* ed *imbardata*. Anche in questo caso, picchi elevati e variazioni repentine di tali valori angolari saranno indice di curve effettuate ad alta velocità, sprint energici o repentine frenate. Analizziamo nel dettaglio le grandezze in gioco, le relative unità di misura ed i valori tipici in stili di guida opposti.

6.3.1 – Velocità

La velocità misura quanto velocemente il veicolo si sposta su una distanza in un determinato tempo. Nel caso di interesse, è ricavata dal delta delle coordinate GPS (latitudine e longitudine) rispetto al tempo.

Unità di misura: metri al secondo (m/s).

Guida aggressiva: Velocità variabile con frequenti aumenti e diminuzioni rapide.

Guida tranquilla: Velocità più stabile e gradualità variazioni.

6.3.2 – Accelerazione (Modulo)

L'accelerazione misura il tasso di variazione della velocità del veicolo. Viene calcolata come modulo risultante delle accelerazioni lungo gli assi X, Y e Z (principalmente

l'accelerazione longitudinale nel caso di una vettura).

Unità di misura: metri al secondo quadrato (m/s^2).

Guida aggressiva: Accelerazioni elevate con frequenti e rapidi cambiamenti di velocità.

Guida tranquilla: Accelerazioni più basse e costanti con cambiamenti di velocità gradualmente.

6.3.3 – Giroscopio

Il giroscopio misura la velocità angolare attorno ai tre assi principali del veicolo (X, Y e Z).

Unità di misura: radianti al secondo (rad/s).

Guida aggressiva: Elevate velocità angolari durante curve strette e brusche, cambiamenti di corsia rapidi e frenate/accelerazioni improvvise.

Guida tranquilla: Velocità angolari minori con rotazioni più gradualmente e controllate.

6.3.4 – Rollio

Il rollio misura l'angolo di inclinazione del veicolo attorno all'asse longitudinale. Indica quanto il veicolo si inclina da un lato all'altro (Figura 6.2).

Unità di misura: radianti.

Guida aggressiva: Maggiore angolo di rollio durante curve strette e veloci.

Guida tranquilla: Minore angolo di rollio con curve più morbide.

6.3.5 – Beccheggio

Il beccheggio misura l'angolo di inclinazione del veicolo attorno all'asse laterale. Indica quanto il veicolo si inclina in avanti o indietro (Figura 6.2).

Unità di misura: radianti.

Guida aggressiva: Maggiore beccheggio durante forti accelerazioni e frenate.

Guida tranquilla: Minore beccheggio con accelerazioni e frenate più dolci.

6.3.6 – Imbardata

L'imbardata misura l'angolo di rotazione del veicolo attorno all'asse verticale. Indica quanto il veicolo ruota a destra o sinistra (Figura 6.2).

Unità di misura: radianti.

Guida aggressiva: Maggiore imbardata durante cambi di direzione rapidi e improvvisi.

Guida tranquilla: Minore imbardata con cambi di direzione più gradual.

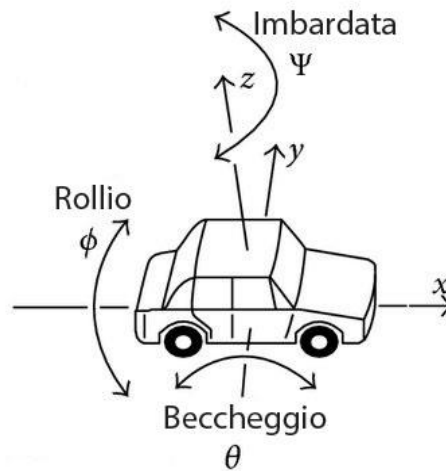


Figura 6.2 – Valori angolari del giroscopio.

6.4 - Preparazione dei Dati

La fase iniziale della preparazione dei dati ha coinvolto la pulizia e la normalizzazione dei dati grezzi acquisiti dall'app *Sensor Logger*. Questo processo è stato essenziale per garantire che i dati fossero completi, coerenti e privi di errori o anomalie che potrebbero compromettere l'accuratezza del modello di machine learning. Le operazioni di pulizia hanno incluso la gestione dei valori mancanti, la rimozione di dati duplicati e la correzione di eventuali inconsistenze nei formati dei dati.

Successivamente, è stata eseguita l'ingegneria delle *features* per estrarre e selezionare le variabili più rilevanti dai dati grezzi. Questo processo ha incluso la creazione di nuove *features* che potessero catturare le caratteristiche distintive degli stili di guida, come la velocità media, l'accelerazione massima, l'accelerazione media ed altre metriche correlate al comportamento del conducente.

Una volta ingegnerizzate, le *features* sono state selezionate attentamente per evitare sovra-apprendimento e migliorare l'efficienza computazionale del modello. Tecniche come l'analisi delle componenti principali o la selezione basata su importanza delle *features* sono state utilizzate per identificare quelle più predittive e ridurre la dimensionalità del dataset senza compromettere la capacità predittiva del modello.

In alcuni casi, è stata applicata la trasformazione dei dati per migliorare la distribuzione o la normalità delle *features*, rendendo i dati più adatti per i modelli di machine learning. Questo può includere la normalizzazione delle *features* per garantire che abbiano scale comparabili o la trasformazione di *features* non lineari per catturare relazioni più complesse tra le variabili.

Infine, il dataset è stato suddiviso in set di addestramento e set di test. Il set di addestramento è stato utilizzato per addestrare il modello di machine learning, mentre il set di test è stato riservato per valutare le prestazioni del modello su dati non visti in

precedenza. Questa pratica è fondamentale per valutare l'effettiva capacità predittiva del modello e per identificare eventuali problemi di generalizzazione.

6.5 – Modello *Random Forest*

Il modello scelto per l'addestramento è un *Random Forest*, una tecnica di machine learning estremamente potente e versatile, ampiamente utilizzata per problemi di classificazione e regressione. È una tecnica che combina diversi alberi decisionali indipendenti tra loro per migliorare la capacità predittiva e la generalizzazione del modello complessivo.

Gli alberi decisionali sono strumenti di machine learning che operano con una serie di domande condizionali su un insieme di *features*, suddividendo iterativamente i dati in gruppi più piccoli ed omogenei. Ogni nodo dell'albero rappresenta una domanda ed ogni foglia rappresenta una classificazione o una previsione.

Il *Random Forest* combina diversi alberi decisionali, ognuno dei quali è addestrato su un sottoinsieme casuale dei dati di addestramento e delle *features*. Questo processo di campionamento casuale ed addestramento indipendente riduce il rischio di *overfitting* e migliora la robustezza del modello complessivo.

Random Forest utilizza il *bagging* per la creazione dei diversi alberi. Il *bagging* consiste nel campionare casualmente più volte il dataset di addestramento con sostituzione (*bootstrap*), addestrando un modello su ciascun campione e combinando le loro previsioni per ottenere una previsione finale più stabile ed accurata.

Il modello di *Random Forest* è stato scelto per la sua robustezza e capacità di gestire grandi volumi di dati, ma anche per le sue prestazioni e la scalabilità. Questo approccio combina efficacemente la resistenza al rumore nei dati, la capacità di generalizzare su nuovi dati e l'efficienza nel trattare i numerosi dati raccolti dalle sessioni di guida tramite l'app *Sensor Logger*. La sua scalabilità e le performance consentono di mantenere tempi di risposta rapidi e di gestire in tempo reale i flussi di dati, essenziali per il monitoraggio continuo e l'aggiornamento del sistema.

6.6 – Addestramento del modello

Il processo di addestramento (Figura 6.2) del modello per il sistema di rilevazione dello stile di guida inizia con la connessione a MongoDB per recuperare i dati necessari. Attraverso l'uso di *pymongo*, è possibile stabilire una connessione al database locale SmartDrive ed accedere alla collezione *test*, dove sono memorizzati i dati raccolti.

Una volta estratti i dati, è necessario prepararli per l'addestramento del modello. Questo include la creazione delle *features* (X) e del *target* (y). Nel caso specifico, le *features* sono composte da *total_acceleration* e *speed*, mentre il *target* è rappresentato da *classification*, che indica la categoria dello stile di guida.

Successivamente, i dati vengono suddivisi in set di addestramento e test utilizzando la funzione *train_test_split* di *scikit-learn*. Questo passaggio è cruciale per valutare l'efficacia del modello su dati non visti precedentemente. Il modello scelto per l'addestramento è un

Random Forest Classifier con 100 alberi decisionali, configurato per massimizzare l'accuratezza e la generalizzazione.

```
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).

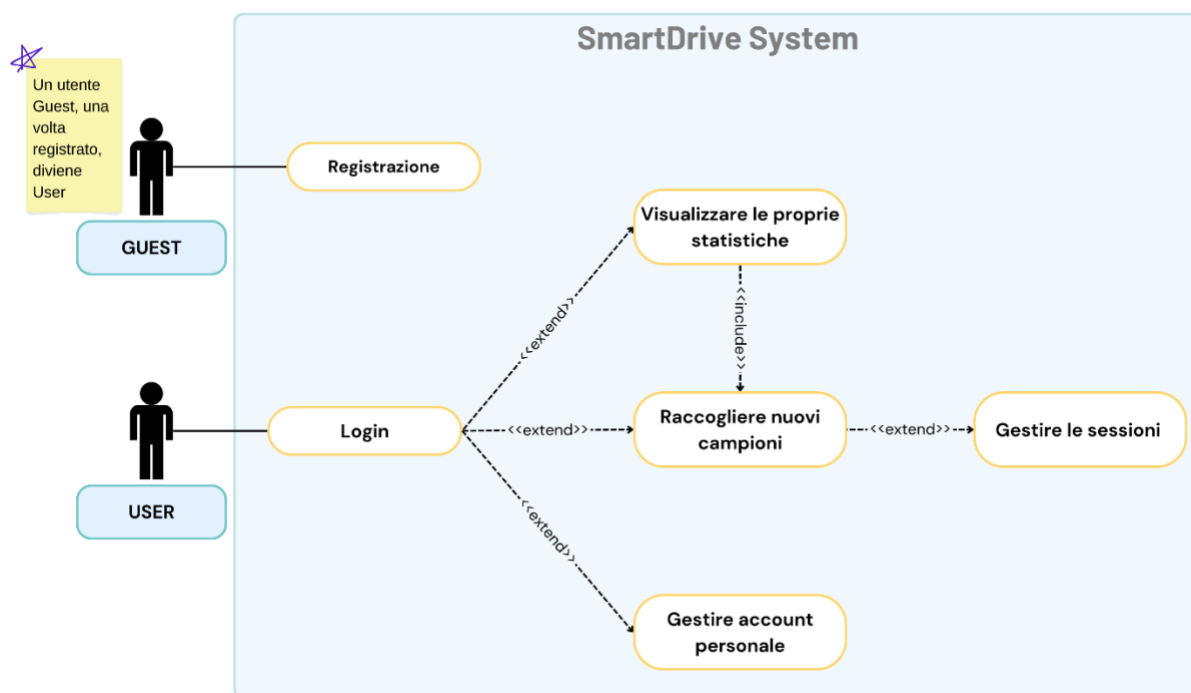


Figura 7.1 - Diagramma dei casi d'uso.

Caso d'uso: Registrazione di un nuovo utente nel Sistema

Descrizione: Questo caso d'uso descrive il processo attraverso il quale un utente guest si registra nel sistema per ottenere accesso alle funzionalità offerte e diviene uno User.

Attori:

- Utente Guest

Pre-condizioni:

- Il sistema è operativo e accessibile.
- L'utente guest ha accesso alla pagina di registrazione nel sistema.

Post-condizioni:

- L'utente guest ha completato la registrazione e diviene un User con accesso alle funzionalità del sistema.

Scenario principale:

1. L'utente guest accede alla pagina di registrazione nel sistema.
2. L'utente inserisce i dati richiesti per la registrazione, che includono:
 - Nome

- Cognome
 - Indirizzo email
 - Password
 - Device_id (identificativo del proprio smartphone)
3. L'utente conferma i dati inseriti e invia la richiesta di registrazione al sistema.
 4. Il sistema verifica i dati inseriti per assicurarsi che siano completi e corretti.
 5. Se i dati sono validi, il sistema crea un nuovo account utente nel database del sistema.
 6. L'utente riceve una conferma di registrazione e può accedere al sistema utilizzando le credenziali fornite.

Scenari alternativi:

- *Dati incompleti o errati:* Se alcuni dati inseriti dall'utente guest non sono completi o corretti, il sistema avvisa l'utente e richiede la correzione o l'integrazione dei dati mancanti.
- *Email già esistente:* Se l'indirizzo email fornito dall'utente guest è già in uso nel sistema, il sistema richiede di utilizzare un altro indirizzo email.
- *Errore di sistema:* Se si verificano errori durante il processo di registrazione (ad esempio, problemi di connessione o errori di sistema), il sistema registra l'evento e informa l'utente guest.

Caso d'uso: Login di uno User nel sistema

Descrizione: Questo caso d'uso descrive il processo attraverso il quale un utente registrato accede al sistema utilizzando le proprie credenziali.

Attori:

- User

Pre-condizioni:

- Il sistema è operativo e accessibile.
- L'utente è già registrato nel sistema.

Post-condizioni:

- L'utente registrato ha completato l'autenticazione e ha ottenuto un JWT valido per l'autorizzazione alle funzionalità del sistema.

Scenario principale:

1. L'utente accede alla pagina di login del sistema.
2. L'utente inserisce le proprie credenziali di accesso, che includono:
 - Indirizzo email
 - Password
3. L'utente conferma i dati inseriti e invia la richiesta di login al sistema.
4. Il sistema verifica le credenziali inserite per assicurarsi che siano corrette.
5. Se le credenziali sono valide, il sistema genera un JWT (JSON Web Token) per l'utente.
6. L'utente riceve il JWT e può accedere alle funzionalità del sistema utilizzando il token per l'autorizzazione.

Scenari alternativi:

- *Credenziali errate*: Se l'indirizzo email o la password inseriti dall'utente non sono corretti, il sistema informa l'utente dell'errore e richiede di reinserire le credenziali.
- *Errore di sistema*: Se si verificano errori durante il processo di login (ad esempio, problemi di connessione o errori di sistema), il sistema registra l'evento e informa l'utente.

Caso d'uso: Avvio di una sessione e raccolta di campioni

Descrizione: Questo caso d'uso descrive il processo attraverso il quale uno User può avviare una sessione di raccolta dati e inviare i campioni raccolti durante la sua sessione di guida dal proprio smartphone al sistema. I campioni vengono registrati nel sistema e associati alla sessione attiva. Lo User può avviare la sessione accedendo al sistema o automaticamente tramite lo smartphone.

Attori:

- User

Pre-condizioni:

- Il sistema è operativo e accessibile.
- Se lo User desidera avviare la sessione manualmente, deve aver effettuato il login ed essere autenticato.
- Lo User ha il proprio smartphone connesso a Internet.

Post-condizioni:

- Lo User ha avviato una sessione di raccolta dati.
- I campioni raccolti dallo smartphone dello User vengono inviati e registrati nel sistema, associati alla sessione attiva.

Scenario uno: Avvio manuale della Sessione

1. Lo User accede al pannello utente del sistema.
2. Lo User avvia una nuova sessione di raccolta dati dal pannello utente del sistema.
3. Il sistema crea una nuova sessione e la associa all'utente.
4. Lo User inizia la raccolta dei campioni di dati dal proprio smartphone durante la sessione di guida.
5. Lo smartphone invia i campioni raccolti al sistema.
6. Il sistema riceve i campioni e verifica il device_id per identificare lo User associato.
7. Il sistema registra i campioni associandoli alla sessione attiva.

Scenario due: Avvio automatico della Sessione

1. Lo User avvia la raccolta dei campioni di dati direttamente dal proprio smartphone senza accedere al sistema.
2. Lo smartphone invia i campioni raccolti al sistema.
3. Il sistema riceve i campioni e verifica il device_id per identificare lo User associato.
4. Il sistema crea automaticamente una nuova sessione e la associa allo User.
5. Il sistema registra i campioni associandoli alla nuova sessione creata.

Scenari alternativi:

- **Device_id non riconosciuto**: Se il device_id inviato non è riconosciuto dal sistema, il sistema non registra i campioni.

- **Errore di sistema:** Se si verificano errori durante il processo di avvio della sessione o di invio dei campioni (ad esempio, problemi di connessione o errori di sistema), il sistema registra l'evento e informa lo User dell'errore.
- **Sessione già attiva:** Se lo User cerca di avviare una nuova sessione mentre una sessione è già attiva, il sistema informa lo User che deve prima terminare la sessione corrente.

Caso d'uso: Visualizzazione delle statistiche dell'utente

Descrizione: Questo caso d'uso descrive il processo attraverso il quale uno User può visualizzare le proprie statistiche relative allo stile di guida e ad altri dati raccolti durante le sessioni nel sistema. Le informazioni sono presentate nella Dashboard dell'utente con grafici e dettagli per fornire una visione completa e dettagliata.

Attori:

- User

Pre-condizioni:

- Lo User è autenticato nel sistema.
- Il sistema ha raccolto e registrato dati relativi alle sessioni di guida dello User.

Post-condizioni:

- Lo User ha visualizzato le proprie statistiche aggiornate nella Dashboard.

Scenario principale:

1. Lo User accede alla Dashboard del sistema.
2. Lo User naviga verso la sezione dedicata alle statistiche e all'analisi dello stile di guida.
3. Il sistema recupera i dati relativi alle sessioni di guida dello User.
4. Il sistema elabora i dati per generare grafici e statistiche dettagliate sullo stile di guida dell'utente.
5. Il sistema visualizza le statistiche aggiornate nella Dashboard dell'utente.

Scenari alternativi:

- **Errore di sistema:** Se si verificano errori durante il recupero o l'elaborazione dei dati (ad esempio, problemi di connessione o errori di sistema), il sistema registra l'evento e informa lo User dell'errore.

Caso d'uso: Gestione del profilo utente

Descrizione: Questo caso d'uso descrive il processo attraverso il quale uno User gestisce il proprio profilo nel sistema. Le operazioni disponibili includono la modifica dei dati personali (nome, cognome, device_id) e la cancellazione del proprio profilo, con la conseguente rimozione di tutti i dati associati nel sistema.

Attori:

- User

Pre-condizioni:

- Lo User è autenticato nel sistema.
- Il sistema contiene i dati del profilo dell'utente.

Post-condizioni:

- Le modifiche al profilo dell'utente sono state applicate o il profilo è stato cancellato con successo dal sistema.

Scenario uno: Modifica del profilo

1. Lo User accede alla sezione di gestione del profilo nel sistema.
2. Lo User visualizza e modifica i dati del proprio profilo, che possono includere:
 - Modifica del nome e cognome.
 - Modifica del device_id (identificativo del dispositivo).
3. Lo User conferma le modifiche apportate al profilo.
4. Il sistema verifica e applica le modifiche al profilo dell'utente nel database.

Scenario due: Cancellazione del profilo:

1. Lo User accede alla sezione di gestione del profilo nel sistema.
2. Lo User seleziona l'opzione per cancellare il proprio profilo.
3. Il sistema richiede conferma da parte dell'utente per procedere con la cancellazione.
4. Lo User conferma la cancellazione del profilo.
5. Il sistema rimuove completamente il profilo dell'utente e tutti i dati associati dal database.

Scenari Alternativi:

- **Modifica dati non riuscita:** Se la modifica dei dati del profilo non riesce (ad esempio, a causa di dati non validi o di errori di sistema), il sistema avvisa lo User dell'errore e richiede di riprovare.
- **Cancellazione annullata:** Se lo User decide di annullare la cancellazione del profilo durante il processo di conferma, il sistema interrompe l'operazione e mantiene il profilo e i dati associati intatti.
- **Errore di sistema:** Se si verificano errori durante il processo di gestione del profilo (ad esempio, problemi di connessione o errori di sistema), il sistema registra l'evento e informa lo User dell'errore.

Caso d'uso: Gestione delle Sessioni

Descrizione: Questo caso d'uso descrive il processo attraverso il quale uno User gestisce le proprie sessioni nel sistema. Le operazioni disponibili includono la modifica del nome della sessione, la modifica dello stato di attività della sessione e la cancellazione della sessione, con la rimozione di tutti i campioni associati.

Attori:

- User

Pre-condizioni:

- Lo User è autenticato nel sistema.
- Il sistema contiene le sessioni create dall'utente e i relativi campioni.

Post-condizioni:

- Le modifiche alla sessione sono state applicate o la sessione è stata cancellata con successo dal sistema, insieme alla rimozione dei campioni associati.

Scenario principale:

1. Lo User accede alla sezione di gestione delle sessioni nel sistema.
2. Lo User visualizza l'elenco delle sessioni create.
3. Lo User seleziona una sessione e sceglie una delle seguenti operazioni:
 - **Modifica del nome della Sessione:** Lo User modifica il nome della sessione e conferma le modifiche.
 - **Modifica dello stato di attività della Sessione:** Lo User cambia lo stato di attività della sessione (attiva o inattiva) e conferma le modifiche.
 - **Cancellazione della Sessione:** Lo User conferma l'eliminazione della sessione.
4. Il sistema verifica e applica le modifiche alla sessione nel database o procede con la cancellazione.

Scenari alternativi:

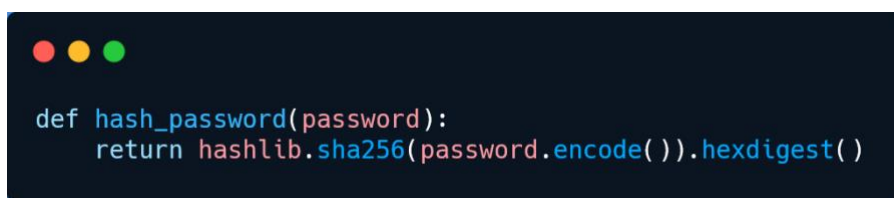
- **Eliminazione dei campioni:** Quando una sessione viene cancellata, il sistema rimuove automaticamente tutti i campioni di dati associati alla sessione dal database.
- **Modifica non riuscita:** Se la modifica del nome della sessione o dello stato di attività non riesce (ad esempio, a causa di errori di sistema), il sistema avvisa lo User dell'errore e richiede di riprovare.
- **Cancellazione annullata:** Se lo User decide di annullare la cancellazione della sessione durante il processo di conferma, il sistema interrompe l'operazione e mantiene la sessione e i campioni associati intatti.
- **Errore di sistema:** Se si verificano errori durante il processo di gestione delle sessioni (ad esempio, problemi di connessione o errori di sistema), il sistema registra l'evento e informa lo User dell'errore.

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 un elevato livello di sicurezza nel sistema, le password degli utenti non vengono memorizzate come testo in chiaro. Invece, prima di essere salvate nel database, le password vengono cifrate utilizzando un algoritmo crittografico. Questo processo di cifratura è fondamentale per proteggere le informazioni sensibili degli utenti da accessi non autorizzati. L'algoritmo è stato implementato nel metodo `hash_password()` (Figura 8.1) utilizzato per cifrare le password nel sistema.

A screenshot of a code editor with a dark background. At the top left, there are three colored circles (red, yellow, green) representing a terminal window. The code is written in a light blue font and shows a Python function definition for `hash_password`.

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

Figura 8.1 - Metodo di cifratura delle password.

In questo processo, la password inserita dall'utente viene convertita in una rappresentazione hash tramite la funzione `hashlib.sha256`. Questo algoritmo di hash prende la stringa della password come input e produce un hash di lunghezza fissa (256 bit in questo caso) in formato esadecimale.

Il valore hash risultante è quindi quello che viene memorizzato nel database per rappresentare la password dell'utente. A differenza del testo in chiaro, l'hash è difficile da invertire, il che significa che anche se il database dovesse essere compromesso, i criminali informatici avrebbero difficoltà ad ottenere le password originali a partire dall'hash memorizzato. Questo approccio garantisce una maggiore sicurezza per gli utenti, proteggendo le loro credenziali da attacchi informatici. Inoltre, assicura che anche gli amministratori di sistema non abbiano accesso diretto alle password degli utenti, migliorando la privacy e la sicurezza complessiva del sistema.

8.2 – JSON Web Token (JWT)

Una volta autenticato nel sistema, all'utente viene fornito un *JSON Web Token* (JWT) che rappresenta un meccanismo standard per la creazione di token sicuri. Questo standard,

definito dal RFC 7519, consente di trasportare in modo sicuro payload cifrati su connessioni non sicure, senza necessità di utilizzare TLS (*Transport Layer Security*).

L'uso di JWT offre diversi vantaggi, specialmente nel contesto del *Web of Things* (WoT). Per esempio, elimina la necessità di gestire certificati complessi, consentendo interazioni sicure tra applicazioni e dispositivi.



Figura 8.2 - Struttura di un JWT.

Un token JWT è strutturato in tre parti separate (Figura 8.2):

- *Header*: Contiene informazioni sull'algoritmo di firma utilizzato ed il tipo di token.
- *Payload*: Contiene le informazioni utili che in questo caso includono dati come l'email e l'ID dell'utente (*user_id*). Queste informazioni consentono di identificare l'utente ed accedere facilmente a dati specifici.
- *Signature*: Utilizza un algoritmo di hashing crittografico (HMACSHA256) per firmare in modo sicuro l'intestazione ed il payload, garantendo l'integrità del token e la sua autenticità.

Il processo di firma include la codifica in base64 dell'intestazione e del payload, seguita dalla generazione della firma tramite l'algoritmo specificato. Questo assicura che il token JWT sia resistente alla manipolazione e possa essere verificato per garantire la sicurezza durante le comunicazioni e le operazioni all'interno del sistema.

Capitolo 9 – Validazione funzionale e/o prestazionale

9.1 – Statistiche del codice

Il progetto è stato analizzato utilizzando il plugin *Statistic* per *PyCharm*, che fornisce una panoramica delle metriche basilari del codice sorgente (Figura 9.1). Questo strumento consente di ottenere una visione chiara ed immediata delle dimensioni e della complessità del codice, facilitando la gestione ed il miglioramento continuo del progetto.

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.

Queste metriche offrono una visione generale dello stato del progetto, aiutando a monitorare l'evoluzione del codice nel tempo e a identificare eventuali aree che potrebbero beneficiare di una rifattorizzazione o di un miglioramento della qualità.

9.2- Valutazione del modello di machine learning

Durante la fase di addestramento del modello di machine learning, è pratica comune valutare le sue prestazioni. Questo aiuta a valutare quanto bene il modello generalizza su dati non visti durante l'addestramento.

```
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.

Nel test sono stati valutati i seguenti aspetti:

- 1) **Cross-Validation Scores:** Durante la fase di addestramento del modello di machine learning, è pratica comune utilizzare la tecnica di cross-validation per valutare le sue prestazioni. Questo metodo aiuta a valutare quanto bene il modello generalizza su dati non visti durante l'addestramento. I "*Cross-Validation Scores*" indicano l'accuratezza del modello su diverse parti del set di addestramento, utilizzate in cinque iterazioni di *cross-validation*. I punteggi riportati rappresentano l'accuratezza del modello su ciascuna delle cinque parti (*fold*) del set di addestramento utilizzate durante la *cross-validation*. Ad esempio, il *fold* 1 ha un'accuratezza del 98.27%, il *fold* 2 del 98.96%, e così via.
- 2) **Media dell'Accuratezza di Cross-Validation:** La "Media dell'Accuratezza di Cross-Validation" è il valore medio dei punteggi di *cross-validation* ottenuti dai cinque *fold*. In questo caso, la media è 0.9882. Questo valore rappresenta l'accuratezza media del modello su tutte le cinque parti del set di addestramento durante la *cross-validation*. Indica quanto bene il modello generalizza su nuovi dati, fornendo una stima dell'accuratezza che ci si può aspettare quando il modello viene applicato a nuovi dati non visti durante l'addestramento.
- 3) **Classification Report:** Il "*Classification Report*" fornisce metriche dettagliate sulle prestazioni del modello su un set di dati di test separato. Le metriche includono:
 - *Precision*: La precisione è la proporzione di istanze positive predette correttamente rispetto a tutte le istanze positive predette (vero positivi diviso per il totale predetto positivo).
 - *Recall*: Il recall (o *sensitivity*) è la proporzione di istanze positive predette correttamente rispetto a tutte le istanze positive reali (vero positivi diviso per il totale reale positivo).
 - *F1-score*: L'*F1-score* è la media armonica di *precision* e *recall*. È una misura complessiva della performance del modello, bilanciando precisione e *recall*.
 - *Support*: Il supporto è il numero di campioni di ogni classe presente nel set di dati di test.

Capitolo 10 – Conclusioni e Sviluppi futuri

10.1 – Obiettivo del progetto

Il progetto mira a implementare un sistema avanzato per valutare e categorizzare lo stile di guida degli utenti attraverso l'utilizzo di un modello di machine learning. L'obiettivo principale è di fornire agli utenti un'analisi approfondita e precisa del loro comportamento alla guida, assegnando loro uno stile di guida tra prudente, normale, sportivo e aggressivo. Questa valutazione avviene tramite l'analisi dei campioni raccolti durante la sessione di guida e resi disponibili a una consultazione attraverso un pannello informativo integrato nel sistema, il quale offre anche altre informazioni dettagliate sulla guida dell'utente, con grafici dettagliati e metriche significative.

10.2 – Obiettivi raggiunti

Il raggiungimento dell'obiettivo è stato conseguito attraverso un processo strutturato di sviluppo, test ed ottimizzazione del modello di machine learning. I risultati ottenuti durante le fasi di prova hanno ampiamente confermato l'efficacia del sistema nel distinguere e classificare diversi stili di guida. In particolare, la validità del sistema è stata verificata tramite prove su strada, le quali hanno fornito conferma definitiva dell'affidabilità delle predizioni.

Il team ha affrontato il processo con un approccio rigoroso, simulando una vasta gamma di stili di guida per garantire la completezza e l'accuratezza delle rilevazioni. I dati raccolti durante queste sessioni hanno coerentemente rispecchiato le aspettative del team, consolidando ulteriormente la solidità e l'efficacia del sistema nel contesto reale.

Questo approccio ha quindi permesso al sistema di essere validato in modo esaustivo, confermando così il conseguimento pieno dell'obiettivo prefissato dal progetto.

10.3 – Sviluppi futuri

Guardando al futuro, sarà possibile integrare diversi miglioramenti significativi per arricchire ulteriormente il sistema di valutazione e categorizzazione dello stile di guida degli utenti. Una prospettiva promettente è l'integrazione di dati provenienti da sensori avanzati come il consumo di carburante, l'usura dei freni ed informazioni ambientali come le condizioni meteorologiche, il traffico e dettagli sulla strada percorsa. L'inclusione di questi dati potrebbe notevolmente migliorare la precisione e la completezza delle valutazioni dello stile di guida, consentendo al modello di machine learning di adattarsi più efficacemente alle specifiche esigenze degli utenti e alle variabili ambientali.

Inoltre, ampliare l'applicazione del sistema per includere il monitoraggio dello stile di guida in contesti lavorativi, come aziende di trasporto, corrieri o conducenti di veicoli pesanti, rappresenta un'opportunità chiave. Questo consentirebbe di migliorare la sicurezza e l'efficienza anche in ambiti professionali critici, contribuendo a promuovere comportamenti di guida più sicuri e responsabili.

Un'altra strategia potenziale è l'adattamento del modello di machine learning per considerare le specificità di diversi tipi di veicoli. Questo approccio potrebbe migliorare ulteriormente l'accuratezza delle valutazioni, consentendo al sistema di classificare lo stile di guida in modo più dettagliato e contestualizzato rispetto al tipo di veicolo utilizzato.

Infine, l'adozione di approcci avanzati di machine learning, come le reti neurali profonde o gli algoritmi di apprendimento federato, potrebbe consentire una gestione più dinamica e personalizzata dello stile di guida. Questi metodi permetterebbero al sistema di adattarsi in tempo reale ai cambiamenti nel comportamento di guida degli utenti, fornendo feedback immediati e incoraggiando un miglioramento continuo delle abitudini alla guida.

In conclusione, il progetto ha già dimostrato di essere una risorsa preziosa per migliorare la consapevolezza e la gestione dello stile di guida degli utenti. Con ulteriori sviluppi mirati, l'integrazione di nuove tecnologie e un approccio innovativo alla raccolta e all'analisi dei dati, il sistema potrebbe continuare a crescere e ad adattarsi, offrendo benefici sempre più significativi sia agli utenti che agli sviluppatori.

BIBLIOGRAFIA

- [1] <<Driving behavior analysis and classification by vehicle OBD data using machine learning>>, Raman Kumar, Anuj Jain
<https://link.springer.com/article/10.1007/s11227-023-05364-3>
- [2] <<A Review of Intelligent Driving Style Analysis Systems and Related Artificial Intelligence Algorithms>>, Gys Albertus Marthinus Meiring, Hermanus Carel Myburgh
<https://www.mdpi.com/1424-8220/15/12/29822>
- [3]<<Recognition of the Driving Style in Vehicle Drivers >>, Jorge Cordero, Jose Aguilar, Kristell Aguilar, Danilo Chavez, Eduard Puerto
<https://www.mdpi.com/1424-8220/20/9/2597>
- [4]<< Driver emotion recognition based on attentional convolutional network>>, Xing Luan, Quan Wen, Bo Hang
<https://www.frontiersin.org/journals/physics/articles/10.3389/fphy.2024.1387338/full>
- [5]<<Vehicle-to-Everything (V2X) Communication for Intelligent Transportation>>, Chen Chen, Kai Liu, Lei Liu, Qingqi Pei, Dapeng Lan
https://www.mdpi.com/journal/sensors/special_issues/V2X_Intelligent_Transportation
- [6]<< Drivers' Comprehensive Emotion Recognition Based on HAM>>, Dongmei Zhou, Yongjian Cheng, Luhan Wen, Hao Luo, Ying Liu
<https://www.mdpi.com/1424-8220/23/19/8293>