

**Università Del Salento**

**Facoltà di Ingegneria**

Corso di Laurea in Ingegneria dell’Informazione

Documentazione di Progetto

per il corso

*Internet of Things*

**Smart Drive Panel**

Docente

*Prof. Luigi Patrono*

Tutor

Studenti

*Marco Longo*

*Priamo Tarantino*

*Ing. Teodoro Montanaro*

**Indice**

[*Premessa (da eliminare)* 1](#_heading=h.gjdgxs)

[Capitolo 1 – Introduzione 2](#_heading=h.30j0zll)

[Capitolo 2 – Analisi dei requisiti 3](#_heading=h.1fob9te)

[Capitolo 3 – Stato dell’Arte 4](#_heading=h.3znysh7)

[Capitolo 4 – Tecnologie utilizzate 5](#_heading=h.2et92p0)

[2.1 Tecnologia A 5](#_heading=h.tyjcwt)

[2.2 Tecnologia B 5](#_heading=h.3dy6vkm)

[Capitolo 5 – Soluzione proposta 6](#_heading=h.1t3h5sf)

[3.1 Paragrafo A 6](#_heading=h.4d34og8)

[3.2 Paragrafo B 6](#_heading=h.2s8eyo1)

[3.3 Altri paragrafi 6](#_heading=h.17dp8vu)

[Capitolo 6 – Validazione funzionale e/o prestazionale 7](#_heading=h.3rdcrjn)

[Capitolo 7 – Conclusioni e Sviluppi futuri 8](#_heading=h.26in1rg)

***Premessa (da eliminare)***

*Questo documento fornisce il formato e le principali linee guida per la realizzazione di una documentazione di buona qualità, che faciliti lo studente nella scrittura e il professore e il tutor nella lettura. Se ci sono delle variazioni importanti di questa struttura, discutere con il proprio tutor eventuali modifiche*

***Indicazioni Generali****:*

1. *Nella scrittura del documento utilizzare sempre il modo impersonale (es. “Per implementare l’applicazione è stato utilizzato Android..” e non “Per implementare l’applicazione ho utilizzato Android…”)*
2. *Curare la forma del testo, evitando frasi troppo lunghe e complesse, utilizzando opportunamente la punteggiatura e rispettando le regole dell’italiano scritto*
3. *Ad ogni immagine deve essere associata una didascalia esplicativa del contenuto, numerata secondo la convenzione “Figura Num\_capitolo.numero” (es. Figura 5.1 per la prima figura del capitolo 5)*
4. *Le immagini devono essere referenziate nel testo*
5. *I paragrafi vanno numerati secondo la numerazione del capitolo, evitando di lasciare titoli di sezioni senza numerazione*
6. *La documentazione deve avere una lunghezza media dalle 25 alle 40 pagine, in relazione all’entità del progetto*
7. *La documentazione finale in formato cartaceo va consegnata al professore una settimana prima della data dell’appello*
8. *È consigliabile inviare la documentazione in formato elettronico al proprio tutor qualche giorno prima della consegna ufficiale al professore, per una pre-validazione dei contenuti*

# 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 e 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 e i risultati dell'analisi in tempo reale.

# 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

## Nel corso degli ultimi decenni, i sistemi di rilevazione degli stili di guida si sono evoluti significativamente, influenzando sia la sicurezza stradale che l'efficienza dei veicoli. Questi sistemi utilizzano una combinazione di sensori, algoritmi di analisi 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 attraverso l'utilizzo di sensori di accelerazione e decelerazione. Questi sensori, come gli accelerometri, erano capaci di registrare rapidi cambiamenti di accelerazione lungo gli assi X, Y e Z del veicolo. La loro funzione principale era quella di rilevare eventi critici come frenate improvvisi 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.

### 3.1.2 - Avanzamenti nei sensori e nell'elettronica

Negli anni 2000, l'evoluzione dei sistemi di monitoraggio degli stili di guida è stata significativamente 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 al suo 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.

### 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'intelligenza artificiale 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.

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 a 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.

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. (aggiungere gli aggiornamenti introdotti dal 7 luglio)

# Capitolo 4 – Tecnologie utilizzate

## 4.1 - Acquisizione dati: Sensor Logger

Sensor Logger 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 via 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.

## 4.2 - Back-End: Python

Python 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 e 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.

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 e alle sfide del progetto.

## 4.3 – Database: MongoDB

MongoDB è un database NoSQL flessibile e scalabile che 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 e 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.

## 4.3 – Front-End: Vue.js

Il front-end del sistema è sviluppato con Vue.js, 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* e *D3.js*, che permettono di creare visualizzazioni interattive e informative dei dati raccolti.

# 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 e una gestione efficiente dei dati raccolti tramite smartphone. Il database MongoDB è suddiviso in quattro sezioni principali: samples, session, test e user.

**Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente**

Figura 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 a una specifica sessione che permette di identificare l'utente. Questo garantisce che i dati siano sempre correlati all'utente corretto e 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 e altre 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 e 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 – Diagramma E-R

Il diagramma Entità-Relazione rappresenta la struttura dati fondamentale del sistema, visualizzando le relazioni tra le entità principali e i loro attributi. Questo diagramma fornisce una panoramica chiara e dettagliata della struttura dati, facilitando la progettazione, la comprensione e l'ottimizzazione delle operazioni.

Immagine che contiene diagramma, linea, testo, modello

Descrizione generata automaticamente

Figura 2. Diagramma E-R.

# Capitolo 6 – Soluzione proposta

## 6.1 - 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 di guida e scenari stradali. 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.

## 6.2 - 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 feature per estrarre e selezionare le variabili più rilevanti dai dati grezzi. Questo processo ha incluso la creazione di nuove feature che potessero catturare le caratteristiche distintive degli stili di guida, come la velocità media, l'accelerazione massima, l'accelerazione media e altre metriche correlate al comportamento del conducente.

Una volta ingegnerizzate, le feature sono state selezionate attentamente per evitare sovrapprendimento e migliorare l'efficienza computazionale del modello. Tecniche come l'analisi delle componenti principali o la selezione basata su importanza delle feature sono state utilizzate per identificare le feature 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 feature, rendendo i dati più adatti per i modelli di machine learning. Questo può includere la normalizzazione delle feature per garantire che abbiano scale comparabili o la trasformazione di feature 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.

## 6.3 – Modello Random Forest

Il modello scelto per l'addestramento è un Random Forest, il quale è 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 feature, suddividendo iterativamente i dati in gruppi più piccoli e omogenei. Ogni nodo dell'albero rappresenta una domanda, e 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 feature. Questo processo di campionamento casuale e 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 e 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.

## 6.4 – Addestramento del modello

Il processo di addestramento 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 e 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 feature (*X*) e del *target* (*y*). Nel caso specifico, le feature 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.

Dopo l'addestramento, è essenziale valutare le prestazioni del modello per garantire che sia in grado di generalizzare bene su dati non utilizzati durante l'addestramento. Questo viene eseguito mediante l'utilizzo di cross-validation per misurare l'accuratezza media del modello su diverse parti del set di addestramento.

Infine, il modello addestrato viene testato utilizzando i dati di test separati in precedenza. Le prestazioni del modello vengono valutate attraverso metriche come la precisione, il richiamo e la f1-score, fornite dal metodo *classification\_report*.



Figura 3. Metodo per l'addestramento del modello.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Figura 4. Metriche di test del modello.

**Capitolo 6 – Validazione funzionale e/o prestazionale**

Spiegare come è stata validata, dal punto di vista funzionale e/o prestazionale, l’applicazione realizzata. Per fare questo, si può descrivere un caso d’uso reale, mostrare l’eventuale hardware utilizzato e descrivere come viene utilizzato in tale scenario, corredando la descrizione con opportuni screenshot. Numerare e fornire didascalia a tutte le immagini inserite.

**Capitolo 7 – Conclusioni e Sviluppi futuri**

Riassumere l’obiettivo, evidenziando come questo sia stato effettivamente raggiunto, e accennare ad eventuali sviluppi che possono migliorare il presente lavoro.