Progetto di Ingegneria del Software 2025/26

Università Ca' Foscari Venezia

Piano di progetto

versione 1.2

Colombo

26/01/2026

Document Informations

Green Drive	GD	
Deliverable	Piano di progetto D2	
Data di Consegna	21/10/2025	
Team Leader	Raffaele Cuniolo	903407@stud.unive.it
Team members	Alberto Barison	901146@stud.unive.it
	Riccardo Brollo	902485@stud.unive.it
	Gabriele Bute	895898@stud.unive.it
	Alessandro Derevytskyy	899454@stud.unive.it

Document History

Version	Issue Date	Stage	Changes	Contributors
1.0	21/10/2025	Draft	Presentazione iniziale	Team intero
1.1	28/10/2025	Draft	Aggiunta di ulteriori - rischi - diagrammi di pianificazione	Alberto Barison, Alessandro Derevytskyy
1.2	26/01/2026	Final	- aggiunta analisi dei costi -allineate le date stimate a date effettive	Gabriele Bute

Indice

1. Introduzione

1.1. Sintesi

Il progetto ha come obiettivo principale la realizzazione di un'applicazione mobile per la promozione di una guida sostenibile, basata sull'acquisizione e l'analisi in tempo reale dei dati provenienti dal veicolo tramite interfaccia OBD-II e modulo ELM327.

Attraverso l'elaborazione dei dati relativi a velocità, consumo di carburante e giri motore, l'app mira a:

- Sensibilizzare gli automobilisti sull'impatto ambientale del proprio stile di guida.
- Ridurre i consumi di carburante e le emissioni di CO_2 fornendo feedback personalizzati e suggerimenti di miglioramento.
- Fornire uno strumento di monitoraggio accurato e accessibile.
- Favorire la collaborazione tra cittadini e pubbliche amministrazioni attraverso un sistema di Eco-Score che possa essere integrato in programmi di incentivazione comunale (es. sconti parcheggi, accesso a ZTL, premi simbolici).

Il progetto si inserisce nel più ampio obiettivo della decarbonizzazione del settore dei trasporti, in linea con le direttive europee sulla riduzione delle emissioni e la promozione di una mobilità a zero impatto ambientale.

1.2. Deliverables

I vari deliverables verranno consegnati sotto forma di file di testo e pdf compressi (ZIP) tramite la piattaforma moodle e sono:

2. Piano di testing (18/11/2025)
3. Documento di progettazione (30/11/2025)
4. Versione 1.0 del codice sorgente (22/12/2025)
5. Versione 1.1 del codice sorgente e allineamento documentazione (26/01/2025)

5.1. Evoluzione del progetto

Il progetto seguirà un evoluzione incrementale in più fasi che si ipotizza siano:

- Implementazione del monitoraggio OBD e visualizzazione base dei dati.
- Introduzione dell'Eco-Score e statistiche personalizzate.
- Connessione con piattaforme esterne (es. enti comunali) per incentivi.

Eventuali aggiornamenti futuri potranno includere l'integrazione con veicoli elettrici o plugin per Android Auto.

5.2. Materiali di riferimento

Per la redazione del piano di progetto sono stati analizzati diversi materiali di riferimento, comprendenti fonti dell'inquinamento atmosferico, documentazione tecnica sugli standard OBD-II, manuali relativi ai moduli di interfaccia con i veicoli e i documenti PDF forniti dal docente del corso.

Fonti informative:

- Emissioni di CO2 delle auto: i numeri e i dati. Infografica | Tematiche | Parlamento europeo
- CO2 emissions performance of new passenger cars in Europe | Indicators | European Environment Agency (EEA)
- Air quality - Consilium
- The Impact of Vehicle Technology, Size Class, and Driving Style on the GHG and Pollutant Emissions of Passenger Cars
- The Effect of Aggressive Driving on Vehicle Parameters

Materiali tecnici:

- OBD-II PIDs - Wikipedia
- On-board diagnostics - Wikipedia
- ELM327 AT Commands - Sparkfun
- https://cdn.sparkfun.com/assets/learn_tutorials/8/3/ELM327DS.pdf
- Driving behavior analysis and classification by vehicle OBD data using machine learning - PMC

5.3. Definizioni e abbreviazioni

Sigla / Termine	Significato
OBD-II	On-Board Diagnostics – standard per il monitoraggio dei parametri veicolari
ELM327	Microcontroller per l'interfaccia OBD–Bluetooth/WiFi
Eco-Score	Indicatore sintetico di efficienza e sostenibilità della guida
VPS	Virtual Private Server

2. Organizzazione del Progetto

2.1. Modello del processo

2.1.1. Motivare la risposta in relazione al prodotto da sviluppare

Per la gestione del ciclo di vita del progetto, si è deciso di adottare la metodologia **Waterfall.** L'adozione di tale modello è motivata da fattori cruciali specifici del progetto, che richiedono un approccio sequenziale e rigoroso: **la necessità di requisiti stabili per l'algoritmo, l'esigenza di una forte documentazione preliminare** e la **presenza di vincoli di tempo (scadenze)** che richiedono una pianificazione fissa.

Il progetto riguarda lo sviluppo di un sistema di monitoraggio dati OBD e la creazione di un indice di guida sostenibile. L'approccio **Waterfall** offre il vantaggio di:

- **Definizione Rigida dei Requisiti:** Il modello impone che l'intera analisi (inclusa la validazione del set di dati OBD e la definizione dell'algoritmo di sostenibilità) sia completata e approvata formalmente prima di procedere. Questo **riduce drasticamente il rischio di modifiche sostanziali** durante le fasi di sviluppo e test.
- **Gestione Lineare del Processo:** Lo sviluppo procede in **fasi sequenziali e distinte.** Questo **semplifica la gestione del team,** poiché ogni membro sa esattamente in quale fase si trova e quali sono i deliverable attesi prima di passare al lavoro successivo.
- **Allineamento con le Scadenze Fisse:** Il modello si allinea al vincolo delle scadenze fisse (deadlines), basandosi su un piano di progetto dettagliato.
- **Controllo e Prevedibilità:** Il piano, stabilito all'inizio, offre la **massima prevedibilità sul rispetto dei tempi** e la **gestione delle risorse necessarie** per il completamento di ogni fase.
- **Documentazione Formalizzata:** Ogni fase produce una **documentazione completa e finale** che verrà utilizzata come **base di riferimento** per la fase successiva.

2.2. Struttura organizzativa

La struttura organizzativa del progetto è stata definita per bilanciare l'esigenza di una chiara autorità di progetto con la necessità di flessibilità e autonomia nell'esecuzione tecnica. A tal fine, si adotta un **Modello Ibrido** che combina la filosofia del modello **Waterfall** (per la pianificazione formale) con una struttura a **Matrice** (per l'allocazione delle risorse).

Si opta per una Struttura a Matrice per **rispondere al contesto universitario.** Questa scelta strategica è necessaria in quanto i membri del team non possono dedicarsi al 100% alla realizzazione del progetto.

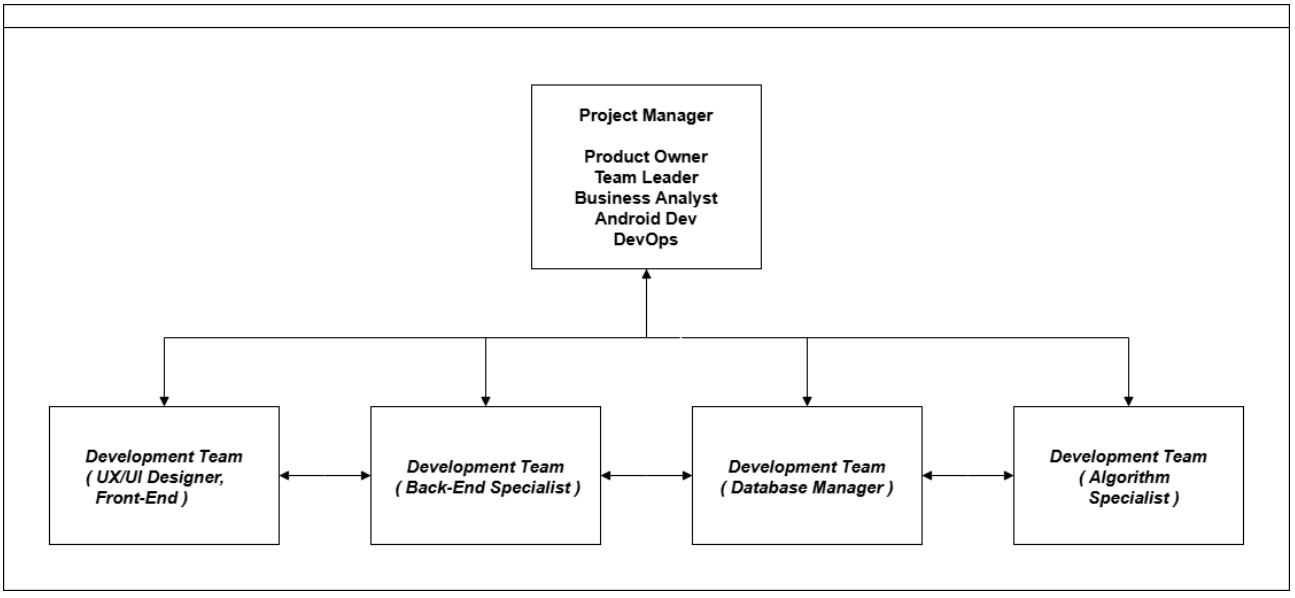
La **Matrice** riconosce che:

- L'autorità primaria sulla disponibilità delle risorse (il tempo dei membri del team) risiede nelle loro **responsabilità funzionali esterne** (gli impegni accademici).
- Il **Project Manager** mantiene l'autorità primaria sul progetto e sui requisiti.
- **L'operatività interna,** pur dovendo rispettare le scadenze e la sequenzialità della pianificazione Waterfall, si basa su principi di **controllo decentralizzato,** garantendo l'efficienza.

La struttura è **gerarchica nel riporto formale** (necessario per la sequenzialità Waterfall), ma **funzionalmente piatta** (essenziale per l'efficienza esecutiva):

- **Ruolo Ibrido del Project Manager:** Il **Project Manager** assume un ruolo ibrido di **massima autorità** per la visione del prodotto, fungendo da **Product Owner (PO)** e **Team Leader** per la direzione dei requisiti. A questo si aggiunge la **partecipazione critica allo sviluppo tecnico** (Sviluppo App Flutter/Networking e gestione della pipeline DevOps). Questa triplice responsabilità (Gestione, Sviluppo e Rilascio) rafforza la comprensione tecnica e l'efficacia decisionale, eliminando ritardi comunicativi.
- **Autonomia del Development Team:** I quattro Sviluppatori, pur riportando formalmente al PM, godono di **piena autonomia sul come implementare** la propria area di specializzazione (Backend, Database, Frontend, Algoritmi). Il **controllo decentralizzato** sulle attività esecutive e la comunicazione interfunzionale bidirezionale sono essenziali per la risoluzione rapida dei problemi. Tale autonomia è vitale per mantenere il ritmo necessario e supportare la rigida sequenzialità delle fasi Waterfall, garantendo l'efficienza nelle consegne tecniche.

2.2.1. Stesura dell'organigramma



2.3. Interfacce organizzative

Questa sezione definisce i punti di contatto formali e i **canali di comunicazione** tra il **Team di Progetto** e gli **Stakeholder esterni.** È essenziale per mantenere il Development Team focalizzato sul lavoro tecnico, convogliando tutte le comunicazioni esterne attraverso **un'unica figura di gestione.**

2.3.1. Definire che si relaziona con committenti/utenti finali/finanziatori

Interfaccia/Stakeholder	Responsabile della Relazione	Ruolo nel Progetto (Giustificazione Ibrida)
Committenti/Utenti Finali	**Project Manager (PO / Business Analyst)**	È l'unica figura autorizzata alla **raccolta e Accettazione Formale dei Requisiti**. Traduce le esigenze in specifiche e gestisce le aspettative e le approvazioni dei deliverables di fase.
Finanziatori/Risorse Esterne	**Project Manager (Team Leader)**	Responsabile della **Reportistica Ufficiale** (es. stato di avanzamento delle fasi) e della gestione dei vincoli del modello a **Matrice**. Comunica esternamente i rischi, le necessità del team e le deviazioni dal piano **Waterfall.**
Team di Sviluppo (Interno)	**Project Manager (Team Leader)**	**Filtro e Direzione:** Agisce come **filtro** per convogliare le **informazioni esterne in modo controllato.** Facilita la comunicazione all'interno del team, **rimuove gli ostacoli** e **protegge gli sviluppatori** dalle interruzioni esterne, garantendo il rispetto della sequenzialità Waterfall.

2.4. Responsabilità di Progetto

2.4.1. Identificazione dei Ruoli

I ruoli all'interno dell'organizzazione di progetto sono stati definiti per garantire una **copertura completa delle competenze tecniche e gestionali**, essenziale per la riuscita delle fasi sequenziali del modello Waterfall e per supportare l'efficienza esecutiva interna.

Si identificano due **macro-ruoli**, con la figura di gestione che è **ibrida** e assume l'autorità formale necessaria al controllo del progetto.

Ruolo di Gestione e Tecnica	Funzioni e Responsabilità Chiave	Ruoli Formali Assunti
Project Manager (PM)	**Gestione/Visione:** Responsabile della **Visione del Prodotto** e della **priorità dei requisiti** (**Product Owner - PO**). Funge da unica interfaccia con gli Stakeholder (**Business Analyst**). **Sviluppo App:** Sviluppo primario della componente Android/Flutter (**Android Dev**). **Rilascio/Pipeline:** Responsabilità della pipeline di Integrazione e Distribuzione Continua (**DevOps**).	**Product Owner (PO)**, **Team Leader**, **Business Analyst**, **Android Dev**, **DevOps**.
Development Team (Sviluppatori)	**Esecuzione Specialistica:** Responsabile dell'implementazione delle funzionalità Backend, Architettura, Frontend (Web/UI) e Database. **Autonomia:** Il Team opera con autonomia sulle soluzioni tecniche, mantenendo la flessibilità interfunzionale.	**System Architect**, **Backend Dev**, **UX/UI Designer**, **Database Manager**, **Frontend Dev**.

2.4.2. Assegnazione dei ruoli ai membri dell'organizzazione

Membro	Ruolo di Gestione (Primario)	Ruoli di Supporto/Tecnici di Gestione	Responsabilità Tecnica Primaria

Raffaele Cuniolo	**Project Manager (PM)**	**Product Owner (PO), Team Leader, Business Analyst**	**Sviluppo App (Flutter), Networking, DevOps** (Pipeline di Rilascio)
Riccardo Brollo	Development Team	**Backend Specialist**	**Logica Applicativa Backend** (API e Servizi Core)
Alessandro Derevytskyy	Development Team	**Database Manager**	**Gestione e Struttura del Database** (Modellazione e Ottimizzazione)
Gabriele Bute	Development Team	**UX/UI Designer, Frontend Specialist**	**Sviluppo Frontend e UI/UX** (Design Interfaccia)
Alberto Barison	Development Team	**Algorithm Specialist, Testing Lead**	**Sviluppo Algoritmi** (Eco-Score) **e Testing Funzionale**

3. Processi gestionali

Di seguito possiamo apprezzare dall'alto un'analisi del flusso di lavoro a fronte di relativi rischi, elementi condizionanti, limitazioni intrinseche alle scelte tecniche ed organizzative.
Gli elementi descritti spaziano dall'applicativo software, concreto, al semplice confronto tra i componenti del gruppo di lavoro.

3.1. Obiettivi e priorità

Trattandosi di un software è necessario assicurarne la realizzazione in tempi ragionevoli, quindi procedere con metodologia coerente alle competenze dei singoli componenti del gruppo, per rispondere al meglio alle specifiche definite. Altrettanto importante è la scelta degli strumenti applicativi per favorire il workflow generale, sincronizzando tutti sui progressi fatti.

3.1.1. Stabilire la periodicità delle scadenze

Il team intero ha adottato un approccio ibrido nella gestione delle scadenze e nell'osservare le stesse con rigore.
Sono state definite immediatamente due categorie: Le dipendenze *hard* e *soft*. Esse differenziano nella possibilità di essere ritrattate durante il corso della loro realizzazione, quanto quindi possano espandersi o restringersi nel tempo occupabile originariamente assegnato. E' utile notare che, generalmente, le dipendenze soft possono essere armonizzate tra loro, ma solo nei limiti stabiliti di una dipendenza hard. Su un piano gerarchico le prime sono "contenute" nelle seconde e tuttavia resta un margine di manovra e di elasticità spesso necessario, in particolare fintanto che abbiamo a che fare con la realizzazione di applicativi tra loro comunicanti come server o database; non fanno eccezione elementi creativi, nel nostro caso sistemi di algoritmi e strategie di networking che richiederanno test di solidità e di efficienza tipici dei sistemi distribuiti.
Al di là delle distinzioni qui sopra, la realizzazione dell'applicativo implica scadenze bloccanti rispetto ad altre, in un ripetersi continuo.

3.1.2. Determinare i fattori da privilegiare

A fronte degli sforzi previsti per raggiungere gli obiettivi principali e secondari, abbiamo deciso di orientare l'organizzazione delle scadenze alla qualità dei micro-risultati, confidenti che non verrà meno il rispetto delle stesse. La componente efficienza vive nel fatto che le parti dell'applicativo sono fortemente comunicanti e che, quindi, è indispensabile completare tempestivamente.

3.2. Assunzioni, dipendenze, vincoli

3.2.1. Descrizione dei fattori esterni che condizionano il progetto

Trattandosi di un sistema di valutazione, la stessa può essere influenzata da condizioni esterne che vanno al di fuori delle responsabilità dell'utente medio e che, giocoforza, può rendere la valutazione finale poco *fair*.
I fattori sono in parte di natura ambientale, metropolitana e strutturale; in certi casi fortemente meccanica.
Un campione di utenti in una città con altimetrie esuberanti (salite improvvise, irregolarità del terreno,...) porterà a risultati decisamente diversi rispetto a quelli riportati nelle città pianeggianti. Anche le condizioni dell'asfalto o che comunque pregiudicano la prestazione dell'autovettura (caldo, freddo) sono da tenere parimenti in considerazione.
Per quanto riguarda il fattore meccanico, il guidatore è soggetto potenzialmente ad un bias di natura socio economica: La categoria Euro-n del veicolo è essa stessa indice di un grado di ricchezza in una città o in un suo distretto, con conseguente limitazione strutturale dei punteggi ottenibili, anche davanti a condotte virtuose degli utilizzatori dell'app.

Una menzione va anche alle dipendenze normative. Le leggi da applicare sono soggette a cambiamenti che possono riflettersi sulla gestione dei dati e sulle modalità di interazione con l'utente, in particolare con la stipula dell'accordo sulla privacy, la gestione dei cookies…

3.3. Gestione dei rischi

La componente statistica richiede di appoggiarsi ad un dataset, ovvero a osservazioni campionarie soggette a errori trascurabili. Tuttavia, gli standard di guida ideali di riferimento da cui trarre valutazioni possono variare con pubblicazioni scientifiche nuove, con conseguente aggiornamento dell'algoritmo di valutazione.

3.3.1. Identificazione dei rischi

L'applicativo in generale propone dei rischi concerni la gestione dei dati sensibili degli utenti. Tra le informazioni critiche viene immediatamente identificata quella della posizione in tempo reale (GPS), assieme ad una rosa di possibile manomissione, intercettamento o furto di dati. Oltretutto, in un contesto di gestione delegata dell'applicativo a enti terzi (comuni, nuclei amministrativi o service providers), appare urgente assicurarsi che questi non usino a scopo di profitto informazioni sensibili.

Dal lato prettamente tecnico i pericoli sono variegati e rientrano nel mondo del web hacking: CORS, database tampering, data-breach, packet–sniffing, cross-site scripting, SQL injection e molti altri di natura informatica. L'app mobile e l'infrastruttura web a supporto sono il cuore dell'origine di potenziali rischi.

Sul fronte organizzativo, invece, indisposizioni del personale possono rallentare l'efficienza del team (stati influenzali ad esempio).

3.3.2. Tabella dei rischi

Impatto	**2** **Molto Improbabile**	**3** **Improbabile**	**4** **Possibile**	**5** **Probabile**
5 – Grave	Medio (7)	Medio (8)	Alto (9)	Alto (10)
4 – Significativo	Medio (6)	Medio (7)	Medio (8)	Alto (9)
3 – Moderato	Basso (5)	Medio (6)	Medio (7)	Medio (8)
2 – Lieve	Basso (4)	Basso (5)	Medio (6)	Medio (7)

Riportiamo la valutazione corrispondente

ID	Rischio	Probabilità	Impatto	Azione
1	Assenza di SSL o protocolli HTTPS	2	4	Implementazione di certificati SSL/TLS e redirezione forzata su HTTPS
2	Mancanza di hashing dei dati sensibili	2	3	Utilizzo di algoritmi di hashing sicuri (es. bcrypt, Argon2) e salting
3	Autenticazione centralizzata vulnerabile	2	4	Adozione di token JWT decentralizzati e verifica lato server
4	Assenza di controlli anti-CORS o anti-SQL injection	3	5	Validazione input, prepared statements e CORS configurato correttamente
5	Assenza di protezione di rete (reverse proxy)	2	4	Instradamento fisso tramite reverse proxy con filtraggio richieste sospette
6	Server non protetti o con utenti privilegiati	4	4	Creazione di utenti non root, gestione privilegi minimi e autenticazione forte
7	Log di sistema non monitorati	3	3	Implementazione di monitoraggio e alerting dei log con strumenti centralizzati
8	Incompatibilità dell'app con l'autoveicolo	2	5	Adattamento delle API standard a quelle specifiche
9	Mancata protezione contro attacchi DoS/DDoS	3	5	Implementazione di firewall, rate limiting e CDN di protezione
10	Rischio di Data Breach (violazione GDPR)	4	5	Applicazione misure GDPR: crittografia, auditing, segnalazione tempestiva e formazione

11	Stati influenzali/infortunistici	2	5	Vaccinazione sistematica antinfluenzale

3.3.3. Azioni di mitigazione/monitoraggio dei rischi

Alle potenziali fragilità seguono una serie di contromisure tecniche, preventive e non, alcune forti di un mandato legale e decise da organi sovranazionali.

Sul fronte *tecnico* sono fondamentali procedure e configurazioni convenzionali che proteggano diverse parti dell'applicativo o tutto.

- SSL, protocolli HTTPS(secure) per sicurezza di base
- Hashing di informazioni sensibili del database
- Metodi di autenticazione decentralizzati JWT
- Meccanismi anti-CORS e anit-SQL injection
- Protezione e instradamento fisso di rete tramite reverse proxy
- Protezione dei server con sistemi di autenticazione e autorizzazione; creazione di utenti non root con privilegio minimo
- Monitoraggio dei log di sistema
- Sistemi di recovery dei singoli servizi nell'infrastruttura (fault recovery)
- Prevenzione contro attacchi DoS e DDoS

Sul lato *legale* sono preferibili, nel contesto già citato di delegazione del servizio o comunque nel caso si instauri una società con responsabilità legali, un rapporto cliente-azienda che risponda ai rigorosi standard del GDPR; da curarsi sono la custodia, la condivisione responsabile e la raccolta non ambigua dei dati; anche i comportamenti da tenere in caso di emergenza (data breach ecc…) richiedono trasparenza col cliente interessato.

3.4. Meccanismi di controllo

3.4.1. Gestione del registro orario e resoconto di lavoro

I diagrammi di Gantt e in senso lato di Pert sono fondamentali per i riferimenti organizzativi dei singoli componenti del team. In accordo con le scadenze sopra citate, il registro è suscettibile parzialmente a variazioni, ma per task più piccoli di sviluppo non c'è un preciso scheduling. Terminare una task piccola prima del previsto permette incontri tra i componenti del gruppo eccezionali, oltre a quelli già fissati periodicamente per fare un compendio sugli obiettivi raggiunti e non.

3.4.2. Gestione dei flussi informativi interni

Il sistema di repository offerto da GitHub comprende features che facilitano la comunicazione tra i processi. Essendo la repository embedded in un progetto, sono state sfruttate creazioni di deadline con notifica, oltre ad esserci appellati all'occorrenza ai GitHub issues. Questi strumenti permettono una comunicazione degli aggiornamenti real-time ai diretti interessati, principalmente via mail, favorendo il workflow e l'incastro delle componenti applicative interdipendenti, seppur realizzate da persone diverse.
Piattaforme per incontri non frontali, dove non possibili quelli in presenza, vengono adoperate solo quando strettamente necessario.

3.5. Pianificazione dello staff

3.5.1. Analisi delle capacità presenti e richieste

Sono state rilevate diverse capacità di natura tecnico-informatica richieste per la realizzazione dell'applicativo, oltre ad una indispensabile qualità alla collaborazione con i membri.
Stabilite le tecnologie da implementare e la soluzione finale, abbiamo raggiunto un'adeguata ripartizione delle responsabilità, senza gravare nessuno in particolare con carichi di lavoro eccessivi.

3.5.2. Pianificazione della formazione

Risultano le diverse assegnazioni di responsabilità, con possibile soccorso tra le parti qualora necessario:

- App Android Flutter e networking: Raffaele Cuniolo
- Database e ORM: Alessandro Derevytskyy
- Backend e ORM: Riccardo Brollo
- Frontend comuni: Gabriele Bute
- Algoritmo di valutazione e parte backend annessa: Alberto Barison

4. Processi tecnici

4.1. Metodi, strumenti e tecniche

4.1.1. Ambiente di sviluppo

L'**ambiente di sviluppo** è stato organizzato come **monorepo su GitHub** con un modello di branching semplificato (branch di sviluppo "feature", branch principale "main" e branch di integrazione "develop") per la gestione del versioneamento.

Il progetto comprende quattro componenti principali:

- **Client mobile**: sviluppato in **Flutter**, con integrazione di componenti nativi in Java per Android (es. interfaccia Android Auto e driver Bluetooth per ELM327). Questo consente la condivisione del codice della user interface tra Android e una possibile futura app iOS, mantenendo comunque la compatibilità con funzioni platform specific.
- **Pannello gestionale web**: sviluppato in **Angular**, utilizzato dall'ufficio comunale per la gestione dei dati raccolti dai veicoli degli utenti e per la definizione delle "zone di interesse".
- **Backend API**: sviluppata in **Node.js con Express**, gestito su VPS monolitica, con database PostgreSQL interfacciato tramite **Prisma ORM**. Le API sono RESTful HTTP-based e utilizzano protobuf come formato serializzato per le risposte e le richieste.
- **Script** di deploy/configurazione automatizzata.

Come ambiente di hosting (e test) si è scelto di usare una **VPS monolitica** basata su Ubuntu Server, con processi di build, test e deploy automatizzati attraverso GitHub Actions. Il server ospita il database PostgreSQL, la applicazione web e il backend REST, serviti tramite Nginx come reverse proxy. I servizi sono containerizzati e gestiti con **Podman**, per garantire l'isolamento e la facilità di aggiornamento. Questo approccio centralizzato semplifica la gestione dell'infrastruttura e mantiene un unico punto di controllo per deploy, configurazioni e segreti, pur sacrificando la scalabilità orizzontale.

Ogni sviluppatore è libero di utilizzare il software e gli strumenti di sviluppo che preferisce, purché le modifiche vengano integrate correttamente nel repository e rispettino le linee guida di progetto. Non è necessario che tutti usino lo stesso ambiente o editor, perché il focus è sul risultato finale e sulla coerenza del codice nel repository.

4.1.2. Linguaggi di programmazione

- **Dart** per l'app Flutter.
- **Java** per le integrazioni native Android.
- **TypeScript/JavaScript** per backend (Node.js/Express) e frontend web (Angular).
- **SQL** per la definizione e gestione del database PostgreSQL.
- **Bash** per gli script di deploy su VPS.
- GitHub Actions per l'automazione di build, test e deploy, gestita tramite workflow **YAML** nel repository.

4.1.3. Standards e linee guida

- **OBD-II** per la lettura dei dati dal veicolo tramite ELM327.
- **Protobuf** per la definizione dei messaggi API e la serializzazione dei dati tra client e server; può essere utilizzato anche su API REST.
- **REST** come stile architetturale per le API, compatibile sia con il client mobile Flutter sia con il pannello web Angular.

- **SAE J1979** per la standardizzazione dei parametri di diagnostica veicolare (RPM, velocità, posizione acceleratore, ecc.).
- **Linee guida interne di codice**: naming convention coerente, documentazione inline e revisione tramite PR su GitHub.
- **Testing e strumenti di supporto**: Postman per testare e condividere gli endpoint REST, assicurando che tutte le funzionalità siano coerenti tra membri del team.

4.2. Documentazione

La documentazione del progetto viene gestita principalmente tramite **Google Drive**, utilizzando **Google Documenti e Google Fogli** come strumento di scrittura collaborativa. Tutti i membri del team contribuiscono direttamente ai documenti, mantenendo così il **versionamento delle modifiche in tempo reale** e la possibilità di revisione continua. Per le consegne ufficiali delle versioni del documento, ogni aggiornamento viene convertito in PDF dal group leader; questi PDF sono poi caricati su GitHub, in modo da avere una copia stabile e accessibile insieme al codice, mentre la modifica continua resta centralizzata su Google Documenti.

Le API vengono documentate e versionate su **Postman** solo dopo l'effettivo sviluppo e test**.**

All'interno di ogni cartella del repository su **GitHub** viene mantenuto un **README markdown specifico** che descrive il funzionamento del componente contenuto, le modalità di esecuzione e i test associati. In questo modo ogni sviluppatore o revisore può rapidamente comprendere come utilizzare il componente, verificarne il corretto funzionamento e seguire le linee guida per l'integrazione con gli altri moduli del progetto.

4.3. Funzionalità di supporto al progetto

4.3.1. Determinare ruolo e cadenza per la pianificazione della qualità

Per garantire un avanzamento coerente e controllato del progetto, il team adotta un approccio organizzato e collaborativo. Il coordinamento tra i membri avviene principalmente tramite **GitHub Issues**, che permettono di assegnare compiti specifici per bug fixing, monitorare lo stato delle attività e mantenere traccia delle milestones di progetto. A queste si affianca un **gruppo WhatsApp** dedicato, utilizzato per comunicazioni rapide e confronto immediato su problemi o decisioni tecniche. L'intero lavoro è pianificato seguendo un diagramma di **Gantt** e le varie deadline prefissate, in modo da procedere in modo costante senza lasciare parti del progetto indietro.

La **qualità del software** è costantemente **monitorata dal group leader**, che interviene sia rileggendo manualmente il codice caricato sul repository, sia tramite controlli automatici integrati nel flusso di deployment. Questo approccio combinato di supervisione manuale e automatizzata assicura che il codice sia coerente, funzionante e conforme agli standard definiti dal team.

4.3.2. Gestione delle configurazioni (es. GitHub).

- I file .proto rappresentano la single source of truth per la definizione dei messaggi tra client e server, condivisi tra tutti i moduli.
- I segreti, come chiavi SSH o credenziali per il database, sono gestiti tramite GitHub Secrets e le variabili d'ambiente della VPS.
- Il deploy delle applicazioni è automatizzato tramite GitHub Actions, che si occupa della build, del trasferimento degli artifact via SSH, dell'aggiornamento dei container gestiti con Podman e dei controlli automatici di salute post-deploy.

5. Pianificazione di attività e risorse

5.1. WBS e dipendenze

Attività principali

Attività	Nome	Inizio	Fine	Durata (gg)	Predecessori	Successori
1	Avvio progetto	07/10/2025	21/10/2025	14		2
2	Analisi requisiti	17/10/2025	11/11/2025	25	1	3
3	Design	31/10/2025	20/11/2025	20	2	4
4	Sviluppo	20/11/2025	02/01/2026	43	3	7
5	Testing e QA	23/12/2025	15/01/2026	23	4	6
6	Rilascio	12/01/2026	25/01/2026	13	5	7
7	Manutenzione e supporto	17/01/2026	24/01/2026	7	6	

Sottoattività

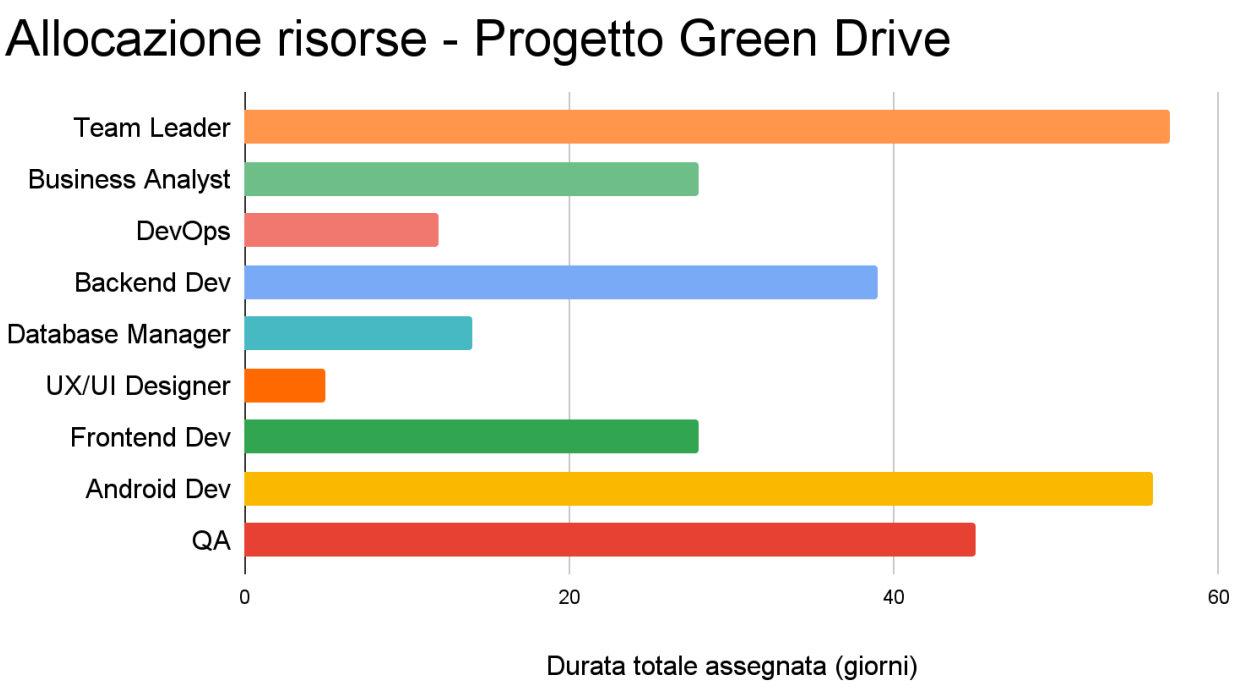
Attività	Nome	Inizio	Fine	Durata (gg)	Predecessori	Successori	Responsabile
1.1	Definizione obiettivi e ambito	07/10/2025	14/10/2025	7		1.2, 1.3, 1.4	Team Leader
1.2	Identificazione stakeholder	14/10/2025	17/10/2025	3	1.1	2.2	Business Analyst
1.3	Pianificazione iniziale	14/10/2025	21/10/2025	7	1.1	2.1, 2.3	Team Leader
1.4	Raccolta requisiti preliminari	14/10/2025	19/10/2025	5	1.1	2.1, 2.3	Business Analyst
2.1	Raccolta requisiti funzionali e non funzionali	21/10/2025	28/10/2025	7	1.3, 1.4	2.4	Business Analyst
2.2	Analisi utenti e	17/10/	20/10/	3	1.2	2.4, 2.5	Business Analyst

	casi d'uso	2025	2025				
2.3	Definizione requisiti tecnici	21/10/2025	31/10/2025	10	1.3, 1.4	3.1, 3.2, 3.4	Team Leader
2.4	Redazione del documento dei requisiti	28/10/2025	11/11/2025	14	2.1, 2.2	3.3	Team Leader
2.5	Validazione con stakeholder	20/10/2025	27/10/2025	7	2.2	3.3	Business Analyst
3.1	Architettura software	31/10/2025	05/11/2025	5	2.3	3.5	Team Leader,DevOps
3.2	Progettazione database	31/10/2025	07/11/2025	7	2.3	3.4, 3.5	Backend Dev,Database Manager
3.3	Progettazione interfaccia utente	11/11/2025	16/11/2025	5	2.4, 2.5	3.5	UX/UI Designer
3.4	Definizione API	07/11/2025	10/11/2025	3	2.3, 3.2	3.5, 4.2	Backend Dev
3.5	Revisione e approvazione design	16/11/2025	20/11/2025	4	3.1, 3.2, 3.3, 3.4	4.1	Team Leader
4.1	Setup ambiente di sviluppo	20/11/2025	25/11/2025	5	3.5	4.2, 4.3, 4.4, 4.5	DevOps
4.2	Implementazione backend	25/11/2025	09/12/2025	14	3.4, 4.1	4.4, 4.7, 5.1	Backend Dev
4.3	Implementazione frontend Android	25/11/2025	23/12/2025	28	4.1	4.7, 5.2, 5.4	Frontend Dev,Android Dev
4.4	Integrazione con API / servizi esterni	09/12/2025	19/12/2025	10	4.1, 4.2	4.7, 5.2	Backend Dev,Android Dev
4.5	Implementazione database locale	25/11/2025	02/12/2025	7	4.1	5.2	Database Manager
4.6	Integrazione sicurezza e autenticazione	09/12/2025	14/12/2025	5	4.2	4.7	Backend Dev
4.7	Code review e refactoring	23/12/2025	02/01/2026	10	4.2, 4.3, 4.4, 4.6	5.3	QA
5.1	Test unitari	23/12/2025	26/12/2025	3	4.2, 4.3	5.6	QA
5.2	Test di integrazione	23/12/2025	26/12/2025	3	4.3, 4.4, 4.5	5.6	QA
5.3	Test funzionali / di sistema	02/01/2026	05/01/2026	3	4.7	5.5	QA
5.4	Test di usabilità	23/12/	26/12	3	4.3	5.6	QA,Android Dev

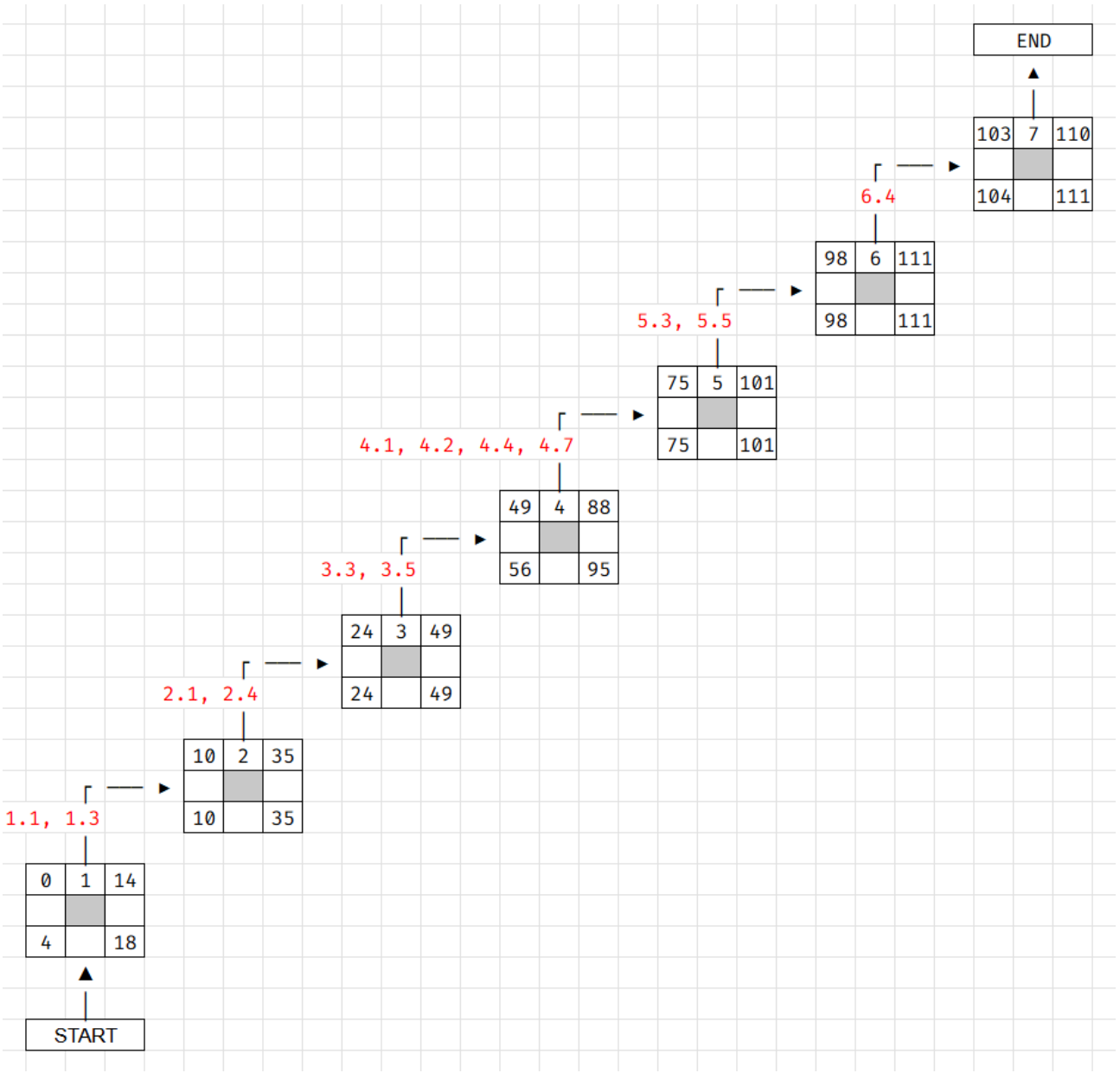
		2025	/2025				
5.5	Bug fixing	05/01/ 2026	12/01 /2026	7	5.3	5.6, 6.1, 6.2	QA
5.6	Validazione finale e accettazione	12/01/ 2026	15/01 /2026	3	5.1, 5.2, 5.4, 5.5	6.3, 6.4	Business Analyst
6.1	Preparazione release	12/01/ 2026	14/01 /2026	2	5.5	6.2	DevOps,Android Dev
6.2	Test di pre-produzione / beta	14/01/ 2026	16/01 /2026	2	5.5, 6.1	6.3	QA
6.3	Deploy su Play Store	16/01/ 2026	17/01 /2026	1	5.6, 6.2	7.1, 7.2, 7.3	Android Dev
6.4	Creazione documentazione	15/01/ 2026	25/01 /2026	10	5.6		Team Leader
7.1	Monitoraggio crash e feedback	17/01/ 2026	24/01 /2026	7	6.3		QA
7.2	Correzione bug post-rilascio	17/01/ 2026	24/01 /2026	7	6.3		QA,Android Dev
7.3	Aggiornamenti e nuove funzionalità	17/01/ 2026	22/01 /2026	5	6.3		Android Dev

5.2. Risorse necessarie

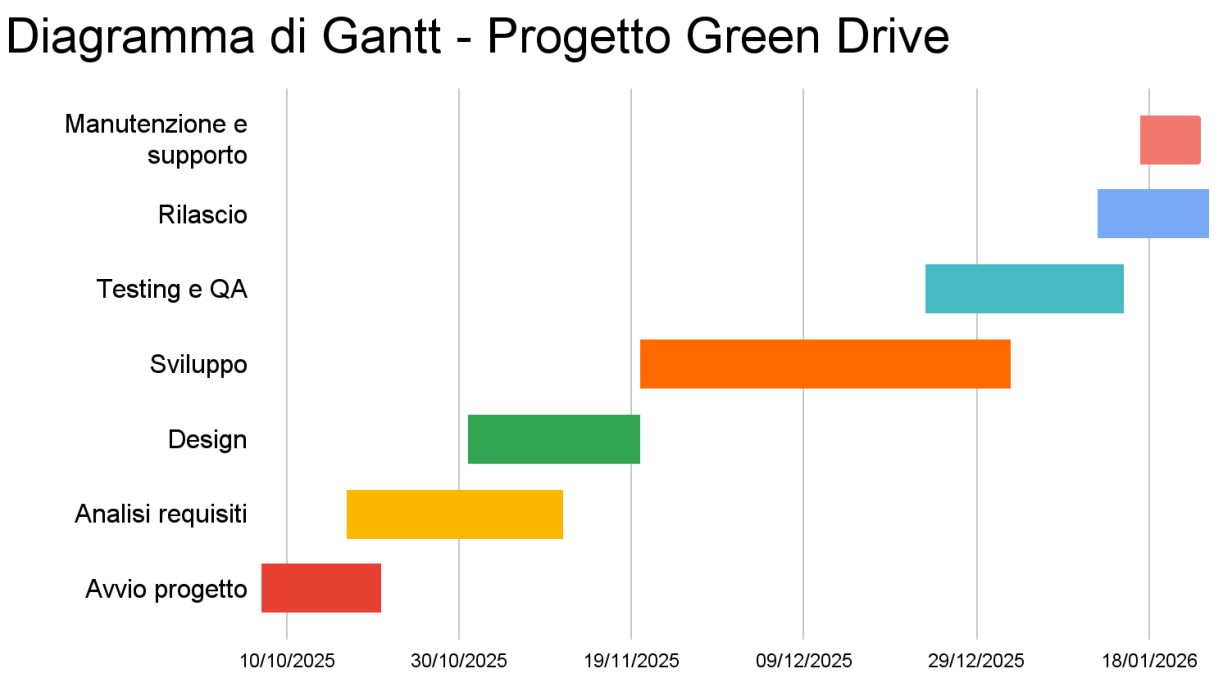
5.2.1. Allocazione complessiva delle risorse

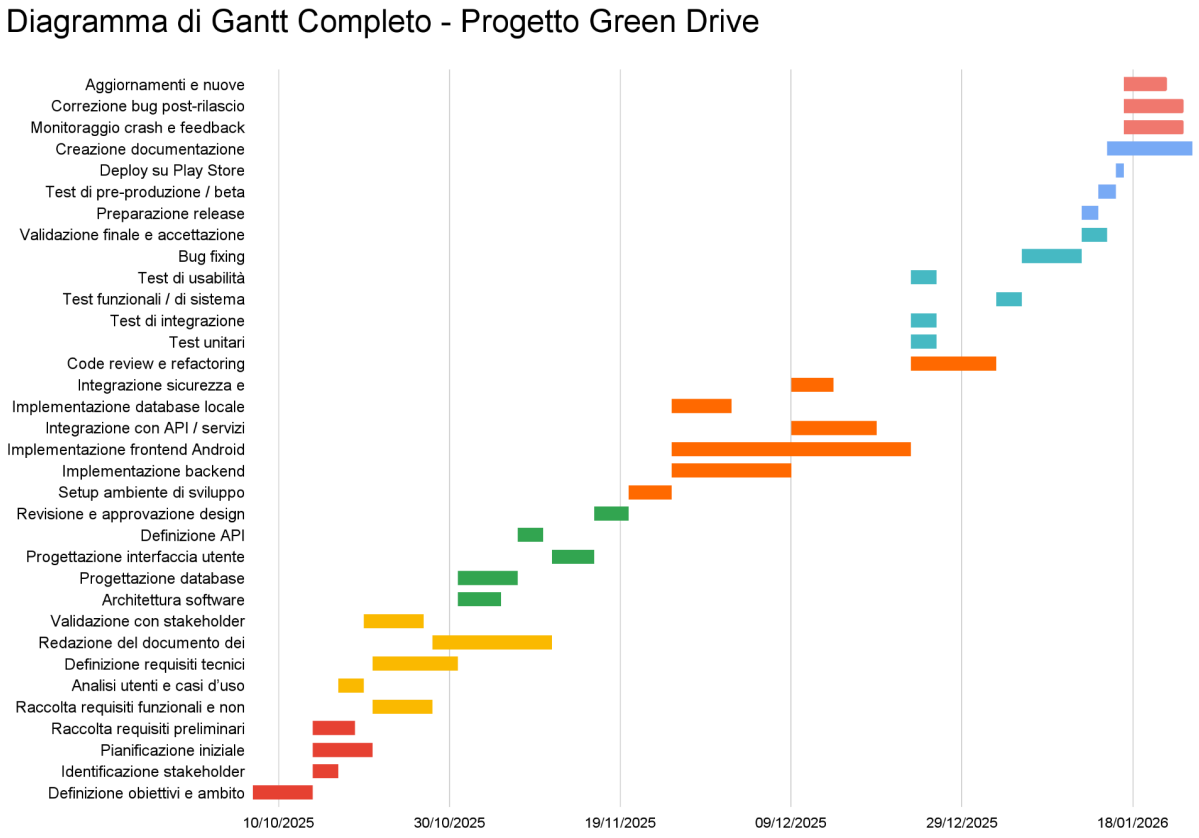


5.2.2. Diagramma di Pert



5.3. Diagramma di Gantt





5.4. Analisi economica

5.4.1. Risorse umane e costo del lavoro

Il valore principale risiede nell'attività intellettuale e tecnica del team.

- **Impegno Orario:** Sono state preventivate in media 150 ore a persona, per un totale di 750 ore di lavoro complessivo per l'intero ciclo di vita del progetto, che va dalla progettazione all'implementazione vera e propria.
- **Tariffa Oraria:** È stata applicata una tariffa media di 30-35 €/h, intesa come costo aziendale lordo. Sebbene non rappresenti un esborso monetario reale, definisce il valore professionale del tempo dedicato, includendo idealmente oneri e costi di gestione che un'azienda sosterrebbe.
- **Valore del Personale:** Il valore stimato del lavoro umano ammonta a circa 24.000 €.

5.4.2. Tecnologie e servizi software

Una scelta strategica fondamentale è stata l'adozione esclusiva di tecnologie Open Source, che ha permesso di azzerare i costi per le licenze software.

- **Sviluppo:** L'utilizzo di framework come Flutter per l'app mobile, Angular per il web e Node.js per il backend non ha comportato costi di licenza.
- **Hosting e Networking:** Per ospitare il backend e il database, è stato utilizzato il tier gratuito di Oracle Cloud, che ha permesso di gestire l'infrastruttura senza costi di canone. La raggiungibilità dei servizi è stata garantita tramite DuckDNS, utilizzato come servizio di DNS dinamico gratuito.
- **Strumenti di Supporto:** La gestione del codice tramite GitHub, la documentazione su Google Docs e l'orchestrazione tramite Podman e Nginx sono state realizzate sfruttando i piani gratuiti per sviluppatori o software liberi, eliminando ogni spesa per servizi esterni.

5.4.3. Infrastruttura hardware e hosting

Le uniche spese vive riguardano le componenti fisiche.

- **Moduli OBD-II:** Per l'interfacciamento fisico con i veicoli, sono stati acquistati due moduli ELM327 a un costo complessivo di 3 €. Nonostante la cifra irrisoria, il valore tecnologico risiede nel driver custom scritto in Java/Dart per l'interpretazione dei dati telemetrici.
- **Asset Preesistenti:** Per lo sviluppo e l'esecuzione dell'app, sono stati utilizzati PC e smartphone già in possesso dei membri del team. Pur non essendo stati acquistati *ad hoc*, la loro disponibilità è stata fondamentale per evitare un investimento iniziale stimabile in circa 4.000 €.
- **Stima dell'Energia Elettrica:** Per le sessioni di sviluppo (750 ore totali), è stato calcolato un consumo medio di 150W per workstation. Considerando il prezzo medio italiano di gennaio 2026 (~0,14 €/kWh), la spesa elettrica stimata per il coding è di circa 15,75 €.

5.4.4. Costi operativi e testing su strada

Per massimizzare l'efficienza e ridurre i rischi e i costi operativi, il team ha adottato una strategia di testing a due fasi:

- **Sviluppo di un Mock di Simulazione:** Per testare la logica dell'Eco-Score e la stabilità dell'app senza gravare sui costi di carburante, è stato realizzato un mock software. Questo strumento ha permesso

di simulare flussi di dati OBD-II complessi in ambiente controllato, garantendo una copertura totale dei test funzionali a costo zero.

- **Validazione su Strada:** La calibrazione finale ha richiesto circa 100 km di prove su strada con un veicolo a benzina. Con un consumo medio riscontrato di 18,5 km/l e un prezzo stimato della benzina di 1,67 €/l, il costo vivo del carburante per i test è di circa 9,03 €.

5.4.5. Riepilogo budget

Il budget totale "teorico" di Green Drive si attesta sui 24.500 €, composto quasi interamente dal valore del lavoro umano (750 ore). Le spese vive reali sono state inferiori ai 30 €, grazie alla combinazione di servizi cloud gratuiti, hardware low-cost e, soprattutto, all'efficienza garantita dal sistema di mock, che ha evitato inutili e costosi test su strada prolungati.