



Progetto di Ingegneria del Software 2025/26

Università Ca' Foscari Venezia

Documento di Progettazione
Versione 1.2

Colombo

26/01/2026



Document Informations

Green Drive		GD
Deliverable		Documento di Progettazione
Data di Consegnna		30/11/2025
Team Leader	Raffaele Cuniolo	903407@stud.unive.it
Team members	Alberto Barison Riccardo Brollo Gabriele Bute Alessandro Derevytskyy	901146@stud.unive.it 902485@stud.unive.it 895898@stud.unive.it 899454@stud.unive.it

Document History

Version	Issue Date	Stage	Changes	Contributors
1.0	30/11/2025	Draft	Presentazione iniziale	Team intero
1.1	25/01/2026	Draft	Aggiunto diagramma di distribuzione	Alessandro Derevytskyy
1.2	26/01/2026	Final	Aggiunto diagrammi di sequenza Stilato paragrafo e aggiunte immagini per interfaccia utente	Alessandro Derevytskyy



Indice

1. Introduzione	4
1.1. Executive Summary	4
1.2. Struttura del documento	4
2. Glossario	4
3. Architettura del sistema	5
3.1. Modello e struttura del sistema	5
3.2. Gestione dei dati	6
4. Modello dei Dati e del Controllo	6
4.1. Gestione dei dati: modello Repository	6
4.2. Modello del controllo	7
5. Modelli UML	8
5.1. Diagramma di distribuzione	8
5.2. Diagramma delle classi	9
5.3. Diagrammi di sequenza	10
5.3.1. Registrazione utente	10
5.3.2. Monitoraggio sessione di guida	10
5.3.3. Definizione zone di interesse	11
6. Progettazione dell'interfaccia utente	11
Schermate cittadino	12
Schermate comune	13



1. Introduzione

1.1. Executive Summary

Il presente documento, ha lo scopo di formalizzare l'architettura tecnica e le specifiche implementative del sistema Green Drive. A valle della fase di analisi dei requisiti, in questo elaborato viene definita la soluzione software adottata per realizzare la piattaforma di monitoraggio della guida sostenibile. Il focus principale risiede nella definizione di un'architettura ibrida che integra l'acquisizione dati IoT (tramite protocollo OBD-II e modulo ELM327) con un'infrastruttura backend centralizzata. Il sistema Green Drive si prefigge di sensibilizzare gli automobilisti sull'impatto ambientale del proprio stile di guida attraverso il calcolo di un *Eco-Score* e di fornire alle amministrazioni comunali strumenti per incentivare comportamenti virtuosi. Questo documento funge da riferimento tecnico per la fase di sviluppo, garantendo che l'implementazione dei moduli (App Mobile Flutter, Dashboard Web Angular e Backend Node.js) proceda secondo principi di modularità, manutenibilità e sicurezza.

1.2. Struttura del documento

L'elaborato è organizzato secondo una progressione logica che parte dalle definizioni generali per arrivare al dettaglio delle interfacce:

- **Sezione 2 - Glossario:** Definisce gli acronimi tecnici e i termini specifici di dominio (es. Eco-Score, OBD-II) utilizzati per evitare ambiguità.
- **Sezione 3 - Architettura del Sistema:** Illustra la struttura ad alto livello, descrivendo l'approccio con VPS monolitica, la containerizzazione dei servizi e la suddivisione tra Client Mobile, Web Dashboard e Backend API .
- **Sezione 4 - Modello dei Dati e del Controllo:** Dettaglia la strategia di gestione dei dati tramite ORM Prisma e PostgreSQL, e il modello di controllo basato su API RESTful e serializzazione Protobuf .
- **Sezione 5 - Modelli UML:** Approfondisce la logica del sistema attraverso diagrammi delle classi e diagrammi di sequenza che mappano le funzionalità critiche come il monitoraggio in tempo reale.
- **Sezione 6 - Progettazione dell'Interfaccia Utente:** Presenta i prototipi delle interfacce per l'app mobile e il portale comunale.
- **Sezione 7 - Riferimenti:** Elenca le fonti bibliografiche e la documentazione tecnica consultata.

2. Glossario

Di seguito sono definiti i termini tecnici e gli acronimi utilizzati all'interno del documento di progettazione.

- **Angular:** Framework open source basato su TypeScript utilizzato per lo sviluppo del Pannello gestionale web rivolto alle amministrazioni comunali.
- **Eco-Score:** Indicatore sintetico calcolato dall'algoritmo proprietario di Green Drive che valuta l'efficienza e la sostenibilità dello stile di guida dell'utente basandosi sui parametri raccolti (RPM, velocità, consumo).



- **ELM327:** Microcontroller standard utilizzato come interfaccia ponte per tradurre i segnali provenienti dalla porta OBD-II del veicolo in un formato trasmissibile via Bluetooth/Wi-Fi allo smartphone.
- **Express:** Framework per applicazioni web per Node.js, utilizzato per strutturare le API REST del backend.
- **Flutter:** Framework UI di Google utilizzato per lo sviluppo dell'applicazione mobile multiplattaforma (Android).
- **OBD-II (On-Board Diagnostics II):** Standard di diagnostica veicolare che permette l'accesso ai dati della centralina (ECU) del veicolo, essenziale per il monitoraggio in tempo reale.
- **Podman:** Motore per la gestione di container (alternativo a Docker) utilizzato nella VPS per isolare e gestire i servizi di backend e database.
- **PostgreSQL:** Sistema di gestione di database relazionale (RDBMS) scelto per la persistenza dei dati degli utenti, delle rilevazioni di guida e delle configurazioni comunali.
- **Prisma:** ORM (Object-Relational Mapping) moderno utilizzato per interfacciare il backend Node.js con il database PostgreSQL, garantendo la tipizzazione sicura delle query.
- **Protobuf (Protocol Buffers):** Metodo di serializzazione dei dati strutturati sviluppato da Google, utilizzato per ottimizzare lo scambio di messaggi tra client e server nelle API.
- **REST (Representational State Transfer):** Stile architettonico utilizzato per la progettazione delle API che gestiscono la comunicazione tra i client (App Android, Web App) e il server.
- **VPS (Virtual Private Server):** Server virtuale basato su Ubuntu che ospita l'intera infrastruttura di backend, database e servizi web in modo centralizzato.
- **Swagger:** Sistema di documentazione e test della rest api.

3. Architettura del sistema

3.1. Modello e struttura del sistema

L'architettura del sistema Green Drive è stata progettata seguendo un approccio client-server centralizzato, supportato da un'infrastruttura di deploy basata su container per garantire isolamento e manutenibilità. La soluzione si articola in tre sottosistemi logici principali che comunicano attraverso interfacce ben definite.

Il sistema è strutturato nei seguenti macro-componenti:

- **Sottosistema Mobile (Client App):** Realizzato con il framework Flutter, questo modulo rappresenta il punto di accesso principale per l'utente automobilista. Oltre alla logica di presentazione (UI), questo sottosistema integra moduli nativi Android (Java) custom per gestire la comunicazione Bluetooth a basso livello con l'hardware esterno.
 - **Funzione principale:** Interfacciamento con il dispositivo ELM327 (OBD-II) per l'acquisizione in tempo reale dei dati telemetrici (RPM, velocità, consumi) e invio asincrono dei dati al backend.
- **Sottosistema Web (Admin Dashboard):** Sviluppato in Angular, fornisce l'interfaccia di gestione per le amministrazioni comunali. Questo client web comunica esclusivamente con le API del backend e non ha accesso diretto al database o ai dispositivi hardware.
 - **Funzione principale:** Configurazione delle zone a traffico sostenibile, visualizzazione delle statistiche aggregate e gestione del sistema di incentivi.
- **Sottosistema Backend (Server):** Costituito da



- Un'applicazione Node.js con framework Express, funge da centro nevralgico del sistema. Espone interfacce API RESTful consumate dai client Mobile e Web.
- Un database PostgreSQL per il salvataggio dei dati degli utenti.
- Un reverse proxy e server web (Nginx) per gestire il traffico verso l'api Express e servire la web app utile alle amministrazioni comunali.
 - o **Gestione dell'Infrastruttura:** L'architettura fisica è ospitata su una VPS monolitica basata su Ubuntu. I servizi (Backend, Database, Web Server) sono containerizzati e orchestrati tramite Podman.

3.2. Gestione dei dati

Per la gestione della persistenza e dello scambio dati, il sistema adotta il modello architettonale a Repository Centralizzato. Questa scelta garantisce che tutti i sottosistemi operino su un'unica fonte di verità, assicurando la consistenza delle informazioni relative agli utenti, allo storico guide e alle configurazioni comunali.

Le tecnologie adottate per l'implementazione di questo modello sono:

- **Database Relazionale (PostgreSQL):** Scelto per la sua robustezza e capacità di gestire dati strutturati complessi. Il database centralizzato memorizza i profili utente, le serie temporali dei dati OBD raccolti e le definizioni geografiche delle zone sostenibili.
- **Object-Relational Mapping (Prisma):** L'interazione tra il backend Node.js e il database è mediata dall'ORM Prisma. Questo livello di astrazione permette di gestire le query in modo type-safe, disaccoppiando la logica applicativa dai dettagli implementativi dell'SQL sottostante.
- **Serializzazione Dati (Protobuf):** A differenza delle tradizionali API REST che utilizzano JSON, il sistema implementa Protocol Buffers (Protobuf) per la serializzazione dei messaggi scambiati tra Client e Server. Questa scelta tecnica ottimizza la banda di rete e riduce i tempi di latenza, un requisito fondamentale per la trasmissione efficiente dei frequenti pacchetti di dati telemetrici provenienti dai veicoli.

4. Modello dei Dati e del Controllo

In questa sezione vengono analizzate le strategie adottate per la gestione della persistenza delle informazioni e per il coordinamento dei flussi di controllo all'interno del sistema Green Drive.

4.1. Gestione dei dati: modello Repository

Per la gestione dei dati persistenti, il sistema adotta il Modello a Repository Centralizzato. Questa scelta architettonale prevede che tutte le componenti del sistema (App Mobile, Dashboard Web, moduli di analisi) facciano riferimento a un'unica base di dati condivisa, garantendo la coerenza e l'integrità delle informazioni.

La realizzazione di questo modello si basa su due pilastri tecnologici:

1. **Repository Fisico (PostgreSQL):** Il database relazionale ospitato sulla VPS agisce come *Single Source of Truth*. Esso centralizza i dati anagrafici degli utenti, le configurazioni delle zone a traffico limitato (ZTL) e, soprattutto, le serie storiche dei dati telemetrici acquisiti dai veicoli.
2. **Livello di Astrazione (Prisma ORM):** L'accesso al repository non avviene tramite query dirette dai client (che comporterebbe un accoppiamento stretto e rischi di sicurezza), ma è mediato dal



backend tramite l'ORM Prisma. Questo garantisce un'astrazione sul modello dei dati, permettendo di evolvere lo schema del database senza impattare pesantemente sulla logica di controllo.

4.2. Modello del controllo

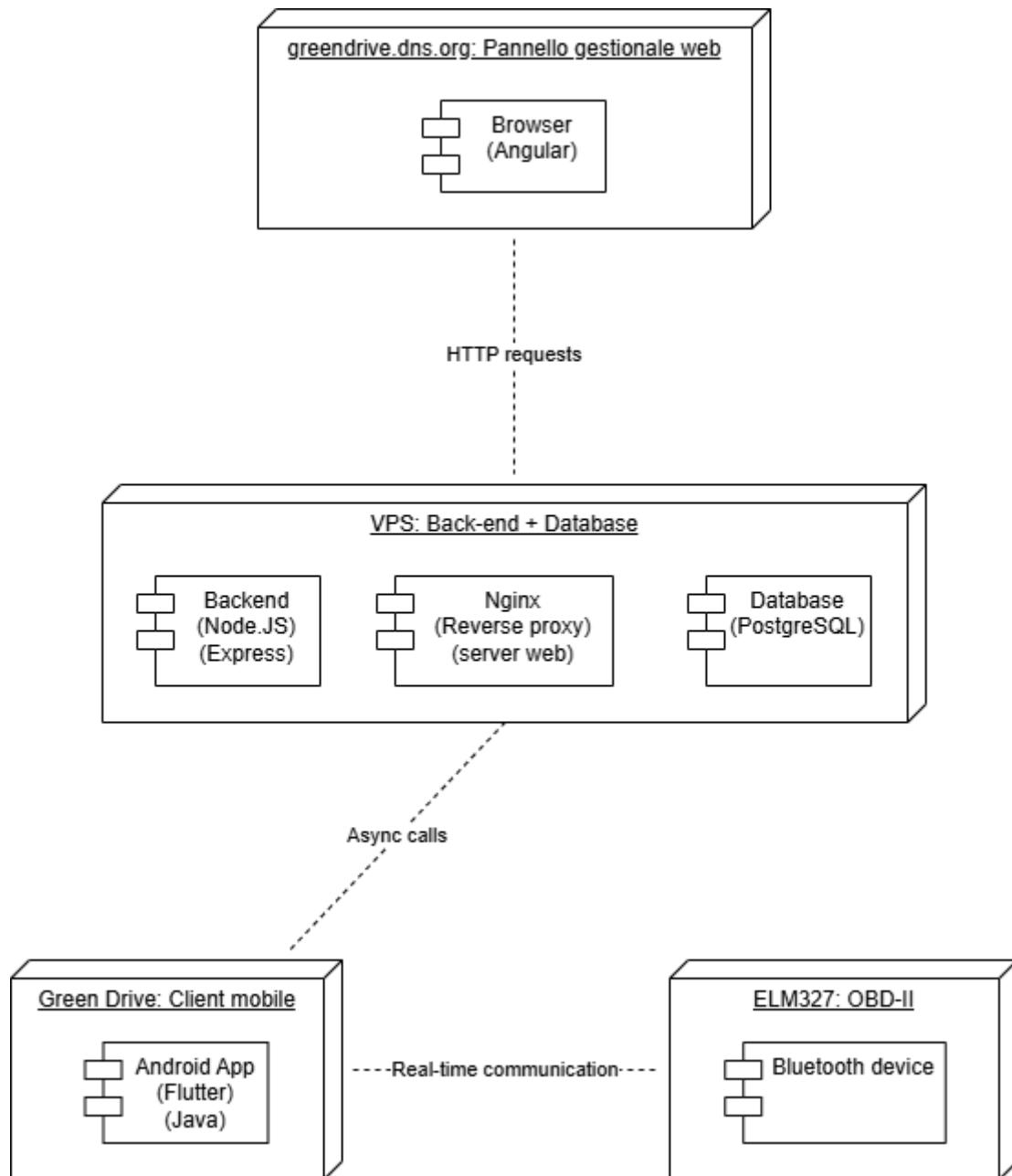
Il sistema Green Drive implementa un modello di controllo ibrido che varia a seconda del sottosistema considerato, per rispondere efficacemente ai requisiti di tempo reale e di consistenza.

- **Controllo Basato sugli Eventi (Sottosistema Mobile/OBD-II):** Per la gestione dell'acquisizione dati dal veicolo, si adotta un modello Event-driven (o guidato da interrupt simulati). L'applicazione Flutter agisce come un *listener* che reagisce al flusso continuo di dati (stream) proveniente dal modulo ELM327 via Bluetooth. Ogni variazione dei parametri (es. cambio RPM, velocità) è un evento che scatena:
 1. L'aggiornamento immediato dell'interfaccia utente.
 2. L'invocazione dell'algoritmo locale per il calcolo istantaneo dell'Eco-Score.
- **Controllo Centralizzato "Manager" (Sottosistema Backend):** Il backend Node.js opera secondo un modello Manager Centralizzato. Esso funge da coordinatore per le operazioni critiche: autenticazione, validazione dei dati inviati dalle app e sincronizzazione con il database. Il server non reagisce passivamente come un semplice archivio, ma orchestra attivamente i processi.



5. Modelli UML

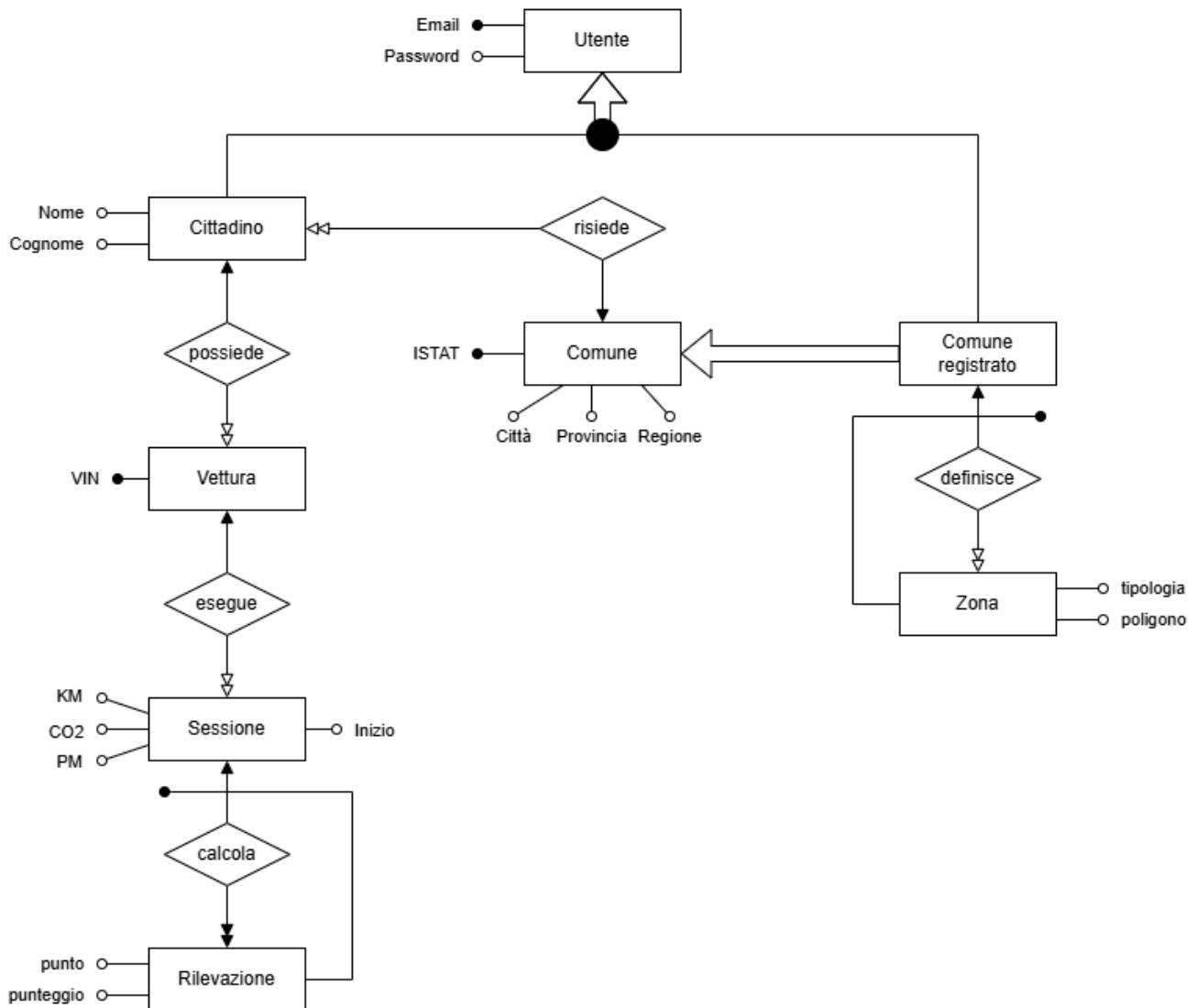
5.1. Diagramma di distribuzione





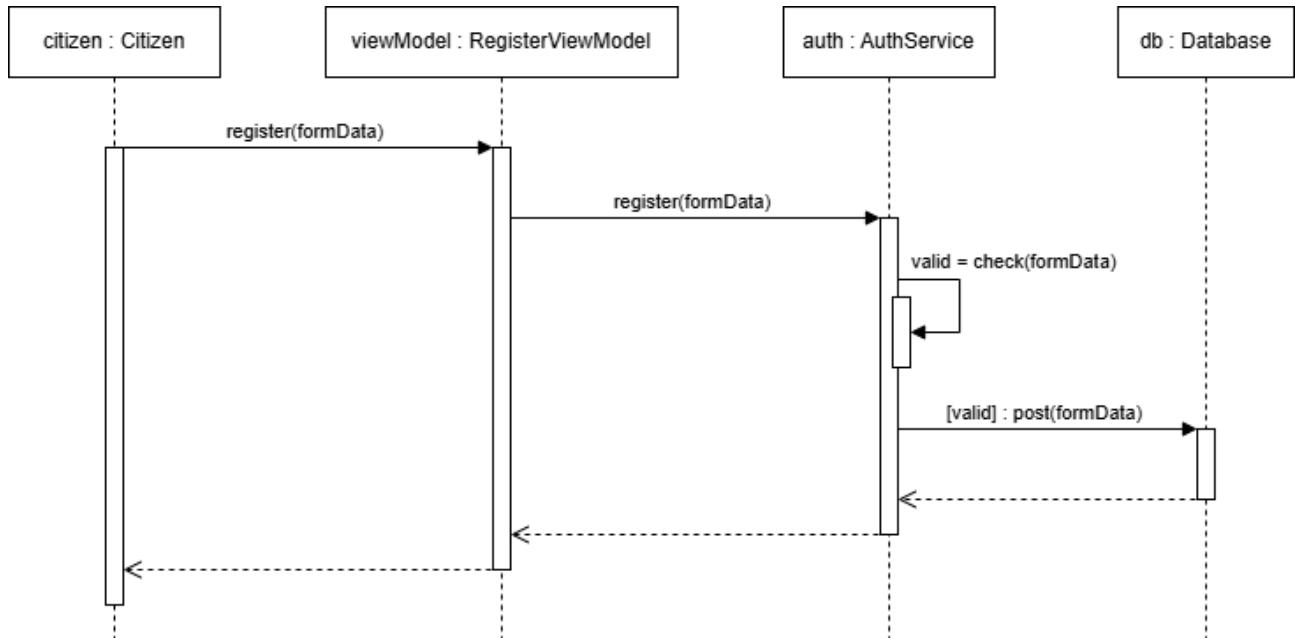
5.2. Diagramma delle classi

Modellazione concettuale

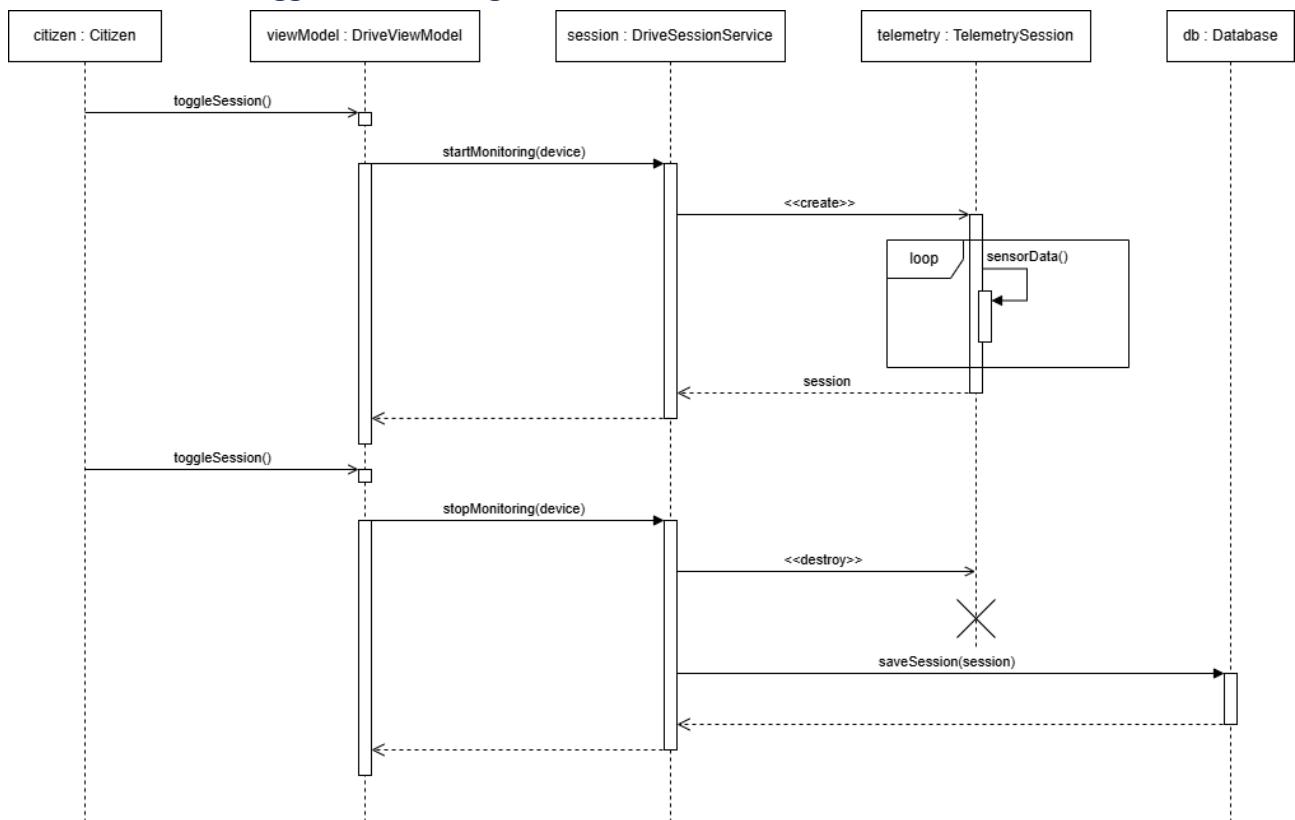


5.3. Diagrammi di sequenza

5.3.1. Registrazione utente

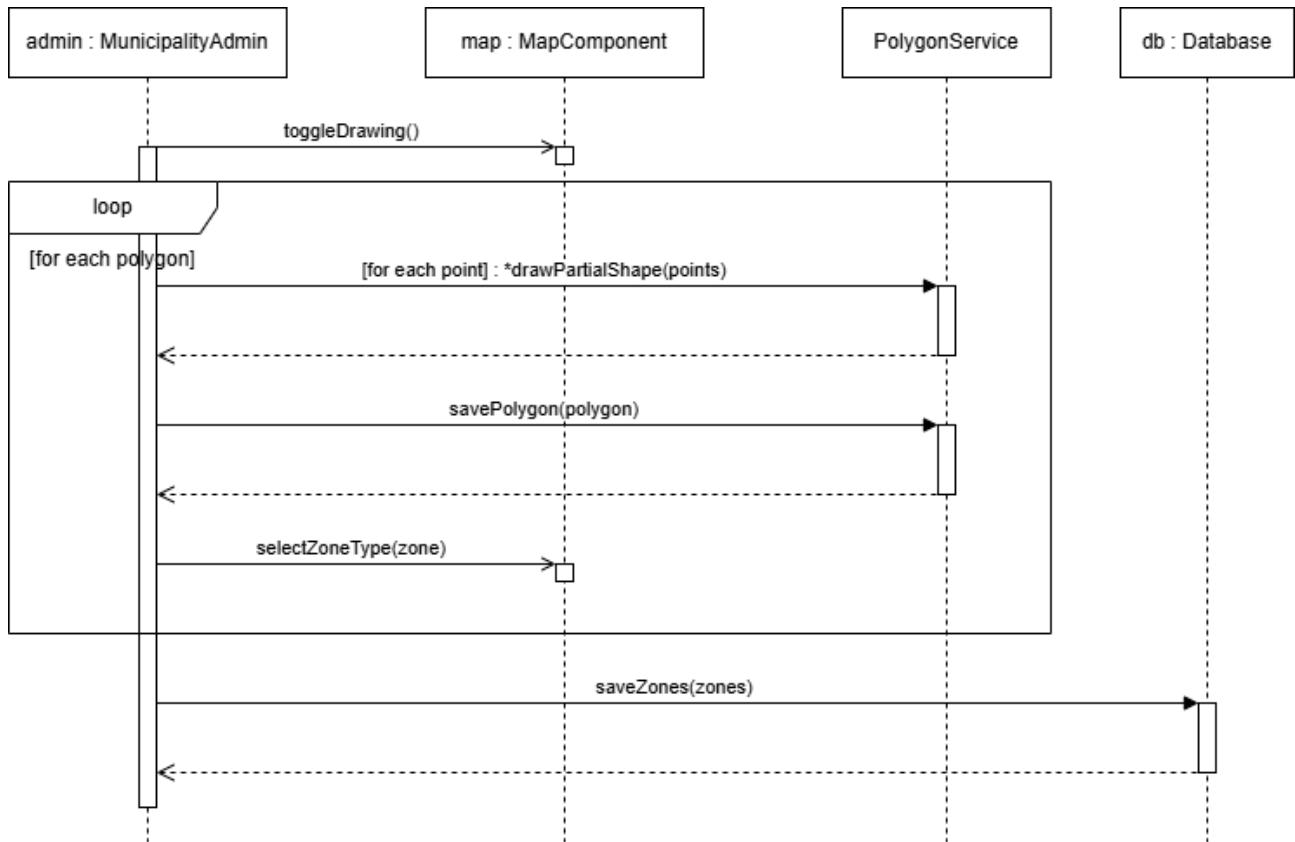


5.3.2. Monitoraggio sessione di guida





5.3.3. Definizione zone di interesse



6. Progettazione dell'interfaccia utente

Prima di procedere con la progettazione delle schermate abbiamo svolto un'indagine per analizzare quali applicazioni fossero state realizzate nell'ambito del monitoraggio delle metriche automobilistiche.

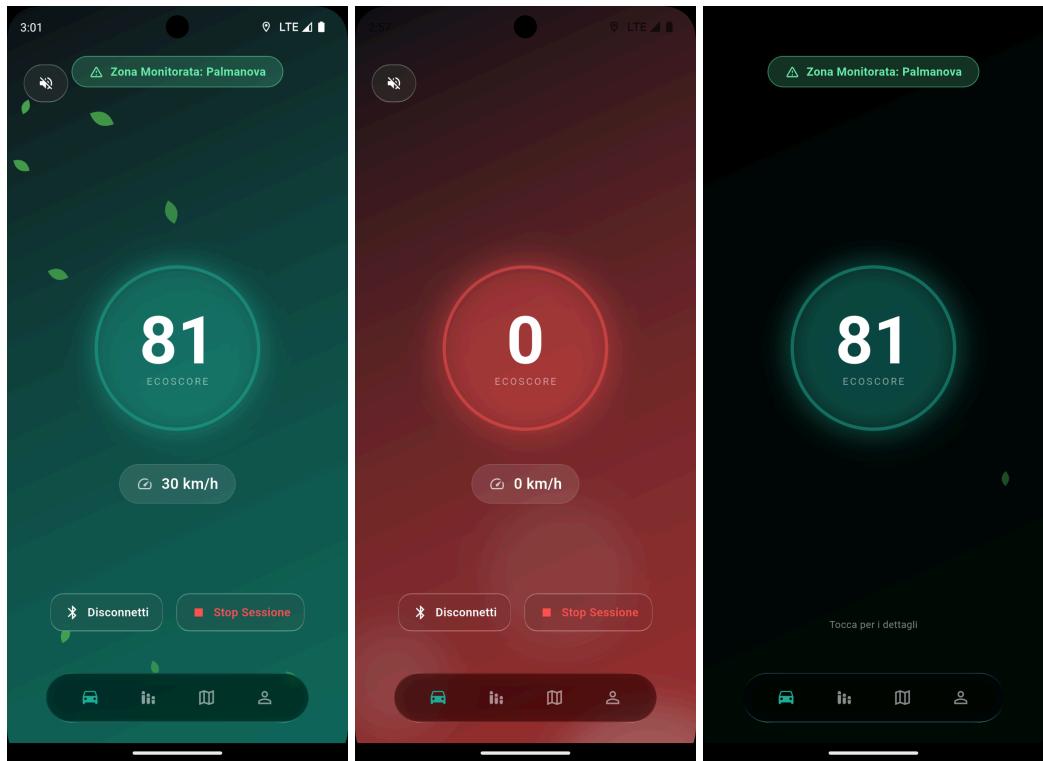
Per focalizzarci sullo sviluppo dell'applicazione siamo poi passati direttamente allo sviluppo delle schermate, suddividendo le funzionalità dei cittadini nell'applicazione realizzata in Flutter, e quelle relative ai comuni in un pannello web creato con la libreria Angular.

Di seguito vengono presentate le schermate per entrambe le categorie:

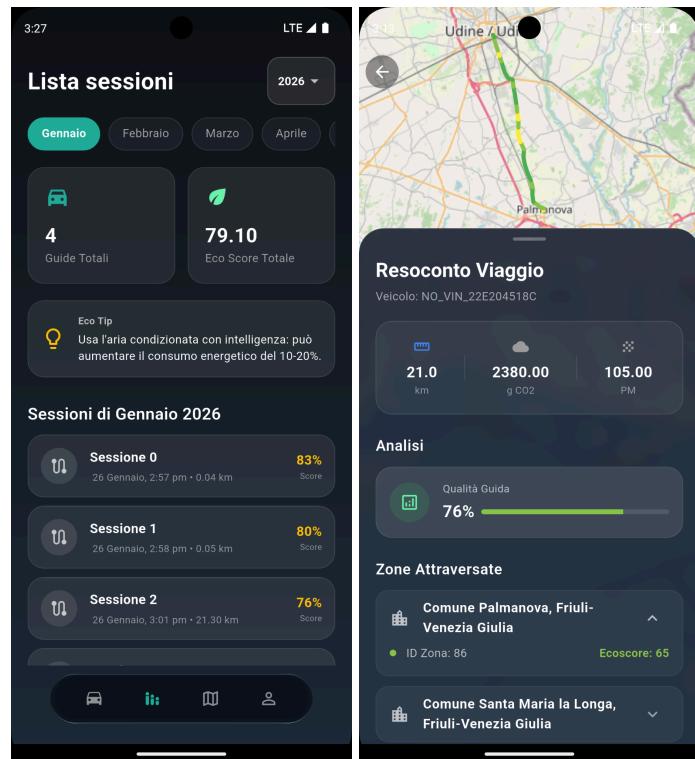


Schermate cittadino

Monitoraggio
sessione di guida



Visualizzazione
storico guide





Consultazione zone a traffico sostenibile



Profilo personale

