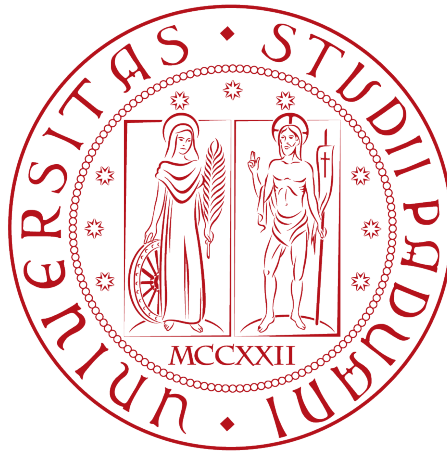


Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

CORSO DI LAUREA IN INFORMATICA



Sviluppo del front end di una applicazione web per il tracciamento di automezzi

Tesi di laurea triennale

Relatore

Prof. Paolo Baldan

Laureando

Mirco Giardina

ANNO ACCADEMICO 2019-2020

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di trecento ore, dal laureando Mirco Giardina presso l'azienda Sync Lab S.r.l.

Il lavoro di stage si inserisce in un progetto, denominato Meterlab, che consiste nello sviluppo di un'applicazione web prototipale in ambito 'Gestione Tracciamento Automezzi', cioè un sistema di geolocalizzazione e trasmissione dati dei mezzi utilizzati per le attività di spazzamento strade.

Gli obiettivi da raggiungere erano molteplici.

In primo luogo era previsto lo studio delle tecnologie necessarie allo sviluppo del [front end](#) del succitato progetto Meterlab. In particolar modo il linguaggio Typescript, il framework Angular e la libreria grafica Material.

In secondo luogo era richiesta la progettazione delle maschere del [front end](#) per l'applicazione web Meterlab, più precisamente per il monitoraggio di veicoli per la pulizia delle strade.

Infine era richiesta l'implementazione di tali maschere e lo sviluppo di un documento tecnico che spiegasse le componenti implementate.

Indice

1	Introduzione	1
1.1	L'azienda	1
1.2	Introduzione al progetto	2
1.3	Architettura del progetto	2
1.4	Problemi riscontrati	3
1.5	Strumenti utilizzati	3
1.6	Prodotto ottenuto	4
1.7	Organizzazione del testo	5
1.7.1	Struttura del documento	5
1.7.2	Convenzioni tipografiche	6
2	Descrizione dello stage	7
2.1	Analisi preventiva dei rischi	7
2.2	Requisiti e obiettivi	8
2.3	Pianificazione	8
3	Analisi dei requisiti	11
3.1	Casi d'uso	11
3.1.1	Attori principali	11
3.1.2	Elenco dei casi d'uso	11
3.2	Tracciamento dei requisiti	32
4	Progettazione e codifica	39
4.1	Tecnologie	39
4.2	Progettazione	40
4.2.1	Architettura di Angular	40
4.2.2	Architettura dell'applicazione sviluppata	41
4.2.3	Progettazione delle maschere	41
4.2.4	Progettazione delle API	42
4.3	Design Pattern utilizzati	42
4.4	Codifica	44
4.4.1	Maschere	44
4.4.2	Componenti	50
4.4.3	Servizi	61
5	Verifica e validazione	65
5.1	Accessibilità	65
5.1.1	Attributi HTML	65
5.1.2	Colori	65

5.1.3	Altri elementi di accessibilità	66
5.2	Verifica	66
5.3	Validazione e collaudo	72
6	Conclusioni	73
6.1	Raggiungimento degli obiettivi	73
6.2	Analisi del lavoro svolto	73
6.3	Valutazione personale	74
	Glossary	75
	Acronyms	77
	Bibliografia	79

Elenco delle figure

1.1	Sedi dell'azienda Sync Lab	1
1.2	Architettura generale del progetto	3
1.3	Informazioni di un veicolo	5
3.1	Scenario principale	12
3.2	UC1: Registrazione	13
3.3	UC4: Autenticazione	16
3.4	UC7: Vis. lista veicoli	17
3.5	UC7.1: Vis. singolo veicolo nella lista	18
3.6	UC7.1.3: Vis. scadenze	19
3.7	UC7.1.4: Vis. parametri	20
3.8	UC7.1.5: Vis. allarmi	23
3.9	UC11: Vis. informazioni account	27
3.10	UC12: Modifica informazioni account	28
3.11	UC12.1: Modifica informazioni generali	29
3.12	UC12.2: Modifica password	30
4.1	Schema logico che rappresenta l'architettura di un applicazione Angular	40
4.2	Mock-up della pagina delle informazioni di un veicolo	41
4.3	Pagina di registrazione	44
4.4	Pagina di login	45
4.5	Pagina di homepage	46
4.6	Pagina dei veicoli	47
4.7	Pagina dei veicoli - Dialogo di conferma eliminazione veicolo	47
4.8	Pagina di impostazioni - Informazioni generali	48
4.9	Pagina di impostazioni - Cambio password	49
4.10	Componente NavComponent - utente non autenticato	50
4.11	Componente NavComponent - utente autenticato	50
4.12	Componente SignupComponent	51
4.13	Componente LoginComponent	52
4.14	Componente VehiclePositionsComponent	53
4.15	Componente VehicleComponent	53
4.16	Funzionalità gestite direttamente da VehicleComponent	54
4.17	Componente SideNavComponent	55
4.18	Componente VehicleMapComponent	56
4.19	Componente vehicleInformationsComponent	56
4.20	Componente vehicleAlarmsComponent	57
4.21	Componente VehicleGraphComponent - Visualizzazione di un solo giorno	58

4.22	Componente VehicleGraphComponent - Visualizzazione di un intervallo di giorni	58
4.23	Componente DeleteDialogComponent	59
4.24	Componente UserSettingsComponent - Informazioni generali	60
4.25	Componente UserSettingsComponent - Cambio password	61
4.26	Messaggio di errore generico	63
5.1	Copertura del codice	72

Elenco delle tabelle

3.1	Tabella del tracciamento dei requisiti funzionali	33
3.2	Tabella del tracciamento dei requisiti qualitativi	36
3.3	Tabella del tracciamento dei requisiti di vincolo	37
5.1	Rapporto contrasti testo-sfondo	65
5.2	Test di unità	66
6.1	Riepilogo dello stato degli obiettivi	73

Capitolo 1

Introduzione

1.1 L'azienda

Sync Lab nasce a Napoli nel 2002 come software house ed è rapidamente cresciuta nel mercato dell'**Information and Communications Technology (ICT)**_G. A seguito di una maturazione delle competenze tecnologiche, metodologiche ed applicative nel dominio del software, l'azienda è riuscita rapidamente a trasformarsi in System Integrator conquistando significative fette di mercato nei settori mobile, videosorveglianza e sicurezza delle infrastrutture informatiche aziendali. Attualmente, Sync Lab ha più di 150 clienti diretti e finali, con un organico aziendale di 200 dipendenti distribuiti tra le 5 sedi dislocate in tutta Italia.

Sync Lab si pone come obiettivo principale quello di supportare il cliente nella realizzazione, messa in opera e governance di soluzione IT, sia dal punto di vista tecnologico, sia nel governo del cambiamento organizzativo.

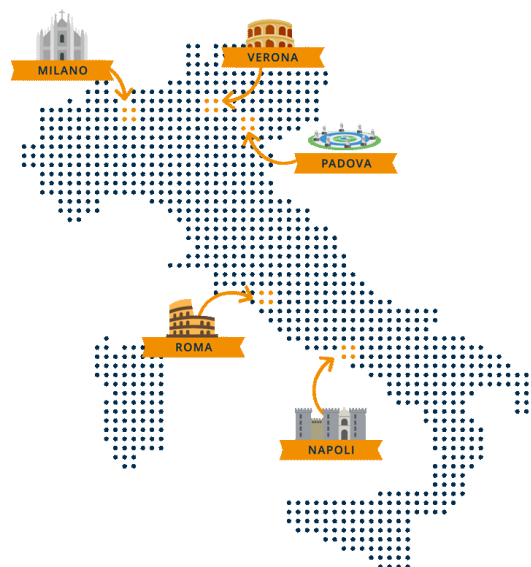


Figura 1.1: Sedi dell'azienda Sync Lab
(Fonte: [7])

1.2 Introduzione al progetto

I sistemi di localizzazione mezzi sono progettati per consentire il controllo dei veicoli aziendali al fine di ricavare informazioni utili alla programmazione e all'organizzazione del lavoro da svolgere.

Questi software permettono di visionare i percorsi stradali effettuati dagli autisti consentendo così allo staff logistico di programmare le tappe da compiere nell'arco della settimana o del mese.

Gli innumerevoli e innovativi software di geolocalizzazione presenti sul mercato vengono scelti proprio per i tanti benefici che questi recano all'attività aziendale. La geolocalizzazione, ad esempio, consente di capire anche in tempo reale, la posizione geografica del mezzo, funzione anch'essa molto utile in caso di furto del veicolo aziendale.

Se da un lato però il controllo dei veicoli aziendali dona efficienza e protezione all'attività imprenditoriale, dall'altro lato c'è la necessità di automatizzare il trasferimento dei dati dal veicolo a un archivio centrale per la loro storicizzazione.

Il progetto di stage proposto da Sync Lab consiste nell'analisi, progettazione e realizzazione di un'applicazione web per il tracciamento e monitoraggio di automezzi.

Tale progetto nasce per mostrare ad un'azienda, che si occupa di pulizia delle strade, un prototipo di quella che potrebbe essere un'applicazione web per monitorare lo stato della flotta dei suoi veicoli.

L'azienda possiede già dei sensori che raccolgono dati sullo stato del veicolo e un dipendente procede ogni sera, al rientro del mezzo, a scaricare i dati su un database usando una chiavetta.

Obiettivo del progetto è quello di automatizzare questo processo e rendere consultabili i dati anche in real time. A tale scopo, i dati devono essere raccolti da un dispositivo che, attraverso la rete mobile, li invia ad un server. L'applicazione web si occupa perciò, consultando i dati sul server, di mostrare all'utente lo stato, sia presente che passato, di ogni veicolo: di quello che è il percorso seguito, le ore di utilizzo di ogni sistema del veicolo e lo stato degli allarmi del motore.

Particolare importanza deve essere data all'archiviazione e storicizzazione dei dati per poter poi essere consultati ed eventualmente usati come prova in caso di contenziosi tra l'azienda di pulizia e il comune appaltatore.

1.3 Architettura del progetto

L'architettura del progetto è composta da quattro elementi principali (figura 1.2):

- **Veicoli:** ogni veicolo raccoglie dati sullo stato dei propri sistemi e li invia usando un raspberry Pi, attraverso la rete mobile, al [back end](#)_G che li storicizza;
- **Server:** sviluppato usando Spring, si occupa di ricevere e gestire le richieste dal [front end](#) e dai veicoli, interfacciandosi con il database;
- **Database:** contiene sia i dati riguardanti i veicoli che i dati riguardanti gli utenti autorizzati ad accedere alle informazioni dei veicoli;
- **Front end:** sviluppato usando Angular, si occupa di richiedere al [back end](#) i dati e di mostrarli all'utente usando un'interfaccia grafica intuitiva.

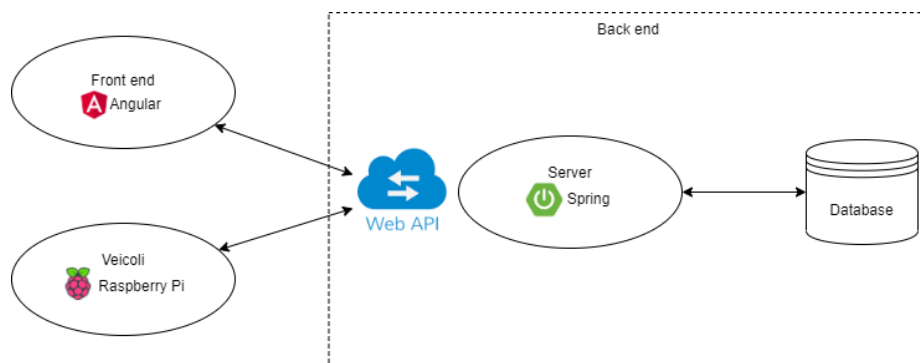


Figura 1.2: Architettura generale del progetto

Al centro dell'architettura c'è quindi il server che, tramite delle [Application Program Interface \(API\)](#)_G, permette da una parte ai veicoli di inviare continuamente i dati aggiornati sul loro stato e dall'altra al [front end](#) di autenticare gli utenti e richiedere le informazioni sui veicoli.

1.4 Problemi riscontrati

Durante lo svolgimento del progetto sono state riscontrate alcune difficoltà che hanno richiesto attenzione da parte del team di sviluppo. In particolare:

- tenendo conto che viene usata una connessione mobile la quantità di dati inviati da ogni veicolo con ogni update è risultata eccessiva. In particolare il problema riguardava il fatto che il veicolo aveva la necessità di aggiornare la sua posizione molto più frequentemente di quanto non dovesse aggiornare lo stato dei suoi sistemi come gli allarmi e le ore di utilizzo degli strumenti. Il problema è stato perciò risolto fornendo due [endpoint](#)_G, uno per aggiornare solo la posizione e uno per inviare sia la posizione che i dati sullo stato del veicolo.
- dopo un'analisi della possibile mole di dati che il [back end](#) avrebbe potuto ricevere da ogni veicolo si è temuto per un eccessivo stress del database. Il problema è stato risolto usando InfluxDB perchè, non dovendo fare operazioni di update e delete sullo storico dei dati, è un database ottimizzato sull'insert di nuovi dati e sulla loro lettura.
- lo sviluppo del [back end](#) ha richiesto più tempo del previsto e perciò non era possibile testare il [front end](#). Il problema è stato risolto usando un server finto che ritornava dei dati prestabiliti e all'occorrenza anche delle risposte di errore.

1.5 Strumenti utilizzati

Diagrams.net

Strumento collaborativo, integrato con Google Drive, per la creazione dei mock-up delle maschere. Utilizzato in fase di progettazione per mostrare al committente come avevamo ideato la grafica dell'applicazione e per ricevere dei feedback su eventuali modifiche prima di sviluppare l'applicazione vera e propria.

Visual Studio Code

Visual studio code è un editor di codice sorgente. Grazie alle numerose estensioni, che è possibile installare, si possono usare una vasta gamma di linguaggi e funzionalità di supporto alla scrittura del codice. Tra le estensioni utilizzate vi sono una per integrare Git, una per migliorare la leggibilità del codice e altre per migliorare l'[intelliSense_G](#) mentre si sviluppa in Typescript e in particolare in Angular.

Git

Git è uno strumento per il controllo di versione. Utilizzato per collaborare con gli altri membri del gruppo e per controllare la versione del codice prodotto così da poter ritornare ad una versione stabile in caso di problemi [4].

Stoplight Studio

Stoplight Studio è uno strumento che permette di progettare le [API](#) utilizzando lo standard OpenAPI. Utilizzato per coordinare il lavoro tra chi sviluppava il [front end](#) e chi sviluppava il [back end](#) e utilizzato in un primo momento come mock-server che rispondeva con esempi preimpostati, di successo o errore, per testare il funzionamento del [front end](#) anche in assenza del [back end](#).

1.6 Prodotto ottenuto

Al termine dello stage l'applicazione web è stata realizzata con successo ed è stata testata ricevendo i dati sulla posizione da una macchina aziendale e il resto delle informazioni da un generatore di dati automatico.

L'applicazione web permette agli utenti di registrarsi, di autenticarsi e di modificare le informazioni del proprio account. Inoltre, permette di visualizzare tutti i dati di un veicolo in un'unica pagina senza scorrimento orizzontale o verticale come richiesto dal committente.

La figura [1.3](#) mostra un esempio di come vengono visualizzate le informazioni di un veicolo.

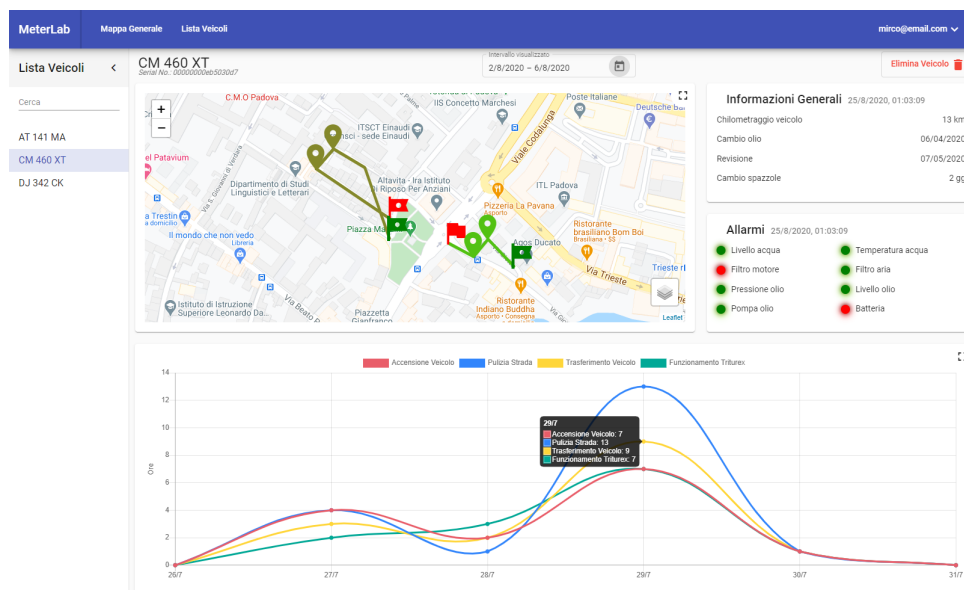


Figura 1.3: Informazioni di un veicolo

Per rendere i dati più facilmente fruibili si è deciso di:

- Mostrare il percorso del veicolo usando una mappa e inserire dei marker cliccabili in ogni punto in cui i dati del veicolo sono cambiati;
- Mostrare lo stato degli allarmi usando dei led verdi o rossi;
- Mostrare in un grafico tutti i dati relativi alle ore di utilizzo dei vari sistemi del veicolo.

Per quanto riguarda la sicurezza sono state implementate due funzionalità:

1. La password dell'utente non viene mai inviata in chiaro sulla rete e viene salvata sul database crittografata. Viene crittografata usando SHA512 sul computer dell'utente e inviata al server che la confronta con la password crittografata salvata sul database.
2. Gli **endpoint** sono stati protetti usando un token che viene generato dal server in fase di autenticazione di un utente. Tale token permette di autenticare le richieste fatte al **back end** e di proteggere le informazioni dei veicoli.

1.7 Organizzazione del testo

1.7.1 Struttura del documento

Il documento è diviso nei seguenti capitoli:

Il secondo capitolo descrive l'analisi preventiva dei rischi, gli obiettivi dello stage e la pianificazione delle ore di lavoro.

Il terzo capitolo approfondisce l'analisi dei requisiti del prodotto.

Il quarto capitolo approfondisce la fase di progettazione e codifica.

Il quinto capitolo approfondisce l'accessibilità e la fase di verifica e validazione.

Nel sesto capitolo contiene un'analisi del lavoro svolto e le conclusioni tratte.

1.7.2 Convenzioni tipografiche

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: parola_G ;

Capitolo 2

Descrizione dello stage

In questo capitolo è presente un'analisi preventiva dei rischi che potevano venire riscontrati durante lo svolgimento dello stage, la lista degli obiettivi da raggiungere e la pianificazione delle ore di lavoro.

2.1 Analisi preventiva dei rischi

Durante la fase di analisi iniziale sono stati individuati alcuni possibili rischi a cui si potrà andare incontro. Si è quindi proceduto a elaborare delle possibili soluzioni per far fronte a tali rischi.

1. Inesperienza tecnologica

Descrizione: alcune delle tecnologie utilizzate nel progetto sono parzialmente o completamente sconosciute.

Soluzione: studio delle tecnologie con corsi forniti dall'azienda e documentazione ufficiale.

2. Impossibilità di recarsi in azienda

Descrizione: per via dell'emergenza covid-19, alcuni giorni potrebbe essere impossibile lavorare in sede per motivi logistici.

Soluzione: predisposizione di strumenti per lo smart working.

3. Monitoraggio scadenze

Descrizione: per via dell'emergenza covid-19, non sarà sempre possibile confrontarsi con il tutor aziendale e potrebbe essere difficile far capire a che punto si è del lavoro.

Soluzione: creazione di un foglio excel condiviso contenente le attività svolte ogni giorno.

4. Problematiche hardware

Descrizione: la strumentazione usata potrebbe essere soggetta a guasti o malfunzionamenti, con conseguenti perdite di dati e/o tempo.

Soluzione: condivisione giornaliera del lavoro svolto su una repository condivisa.

2.2 Requisiti e obiettivi

Notazione

Si farà riferimento ai requisiti secondo le seguenti notazioni:

- *O* per i requisiti obbligatori, vincolanti in quanto obiettivo primario richiesto dal committente;
- *D* per i requisiti desiderabili, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
- *F* per i requisiti facoltativi, rappresentanti valore aggiunto non strettamente competitivo.

Le sigle precedentemente indicate saranno seguite da una coppia sequenziale di numeri, identificativo del requisito.

Obiettivi fissati

Si prevede lo svolgimento dei seguenti obiettivi:

- Obbligatori
 - *O01*: Acquisizione competenze sul framework Angular;
 - *O02*: Capacità di raggiungere gli obiettivi richiesti in autonomia seguendo il crono programma;
 - *O03*: Portare a termine le implementazioni previste con una percentuale di superamento pari al 80%.
- Desiderabili
 - *D01*: Portare a termine le implementazioni previste con una percentuale di superamento pari al 100%.
- Facoltativi
 - *F01*: Dare un contributo importante al gruppo nelle fasi di scouting tecnologico su eventuali librerie grafiche da utilizzare per una consultazione agevole dei dati.

2.3 Pianificazione

Lo stage ha una durata di 8 settimane e prevede lo svolgimento di **300 ore** effettive di lavoro. È diviso in due parti, il primo mese di studio delle tecnologie e il secondo di progettazione e implementazione delle maschere.

La ripartizione settimanale delle attività è la seguente:

- **Prima Settimana - Formazione (40 ore)**
 - (4 ore) Presentazione strumenti di lavoro per la condivisione del materiale di studio e per la gestione dell'avanzamento;
 - (4 ore) Condivisione scaletta di argomenti;

- (8 ore) Ripasso del linguaggio Javascript;
- (24 ore) Inizio studio del linguaggio TypeScript.
- **Seconda Settimana - Formazione (40 ore)**
 - (16 ore) Studio TypeScript;
 - (24 ore) Studio piattaforma NodeJS e AngularCLI.
- **Terza Settimana - Formazione (40 ore)**
 - (40 ore) Studio framework Angular.
- **Quarta Settimana - Formazione (40 ore)**
 - (40 ore) Studio framework Angular.
- **Quinta Settimana - Analisi ed implementazione (40 ore)**
 - (40 ore) Analisi ed implementazione maschere di login/registrazione sul progetto MeterLab.
- **Sesta Settimana - Analisi ed implementazione (40 ore)**
 - (40 ore) Analisi ed implementazione maschere di visualizzazione dati sul progetto MeterLab.
- **Settima Settimana - Analisi ed implementazione (40 ore)**
 - (40 ore) Analisi ed implementazione filtri sulla maschera di visualizzazione dati.
- **Ottava Settimana - Conclusione (20 ore)**
 - (20 ore) Termine implementazione delle maschere e collaudo finale.

Capitolo 3

Analisi dei requisiti

In questo capitolo vengono descritte le funzionalità che il prodotto deve offrire elencando i casi d'uso e i requisiti individuati.

3.1 Casi d'uso

3.1.1 Attori principali

Gli attori individuati sono i seguenti:

- **Utente non autenticato:** attore che indica un utente che non ha ancora effettuato l'autenticazione o la registrazione all'interno dell'applicazione web;
- **Utente autenticato:** attore che indica un utente che ha effettuato l'autenticazione all'interno dell'applicazione web. Ha quindi la possibilità di vedere tutte le informazioni sui veicoli che sarebbero altrimenti inaccessibili.

3.1.2 Elenco dei casi d'uso

Per lo studio dei casi di utilizzo del prodotto sono stati creati dei diagrammi. I diagrammi dei casi d'uso (in inglese *Use Case Diagram*) sono diagrammi di tipo [Unified Modeling Language \(UML\)](#)_G dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso.

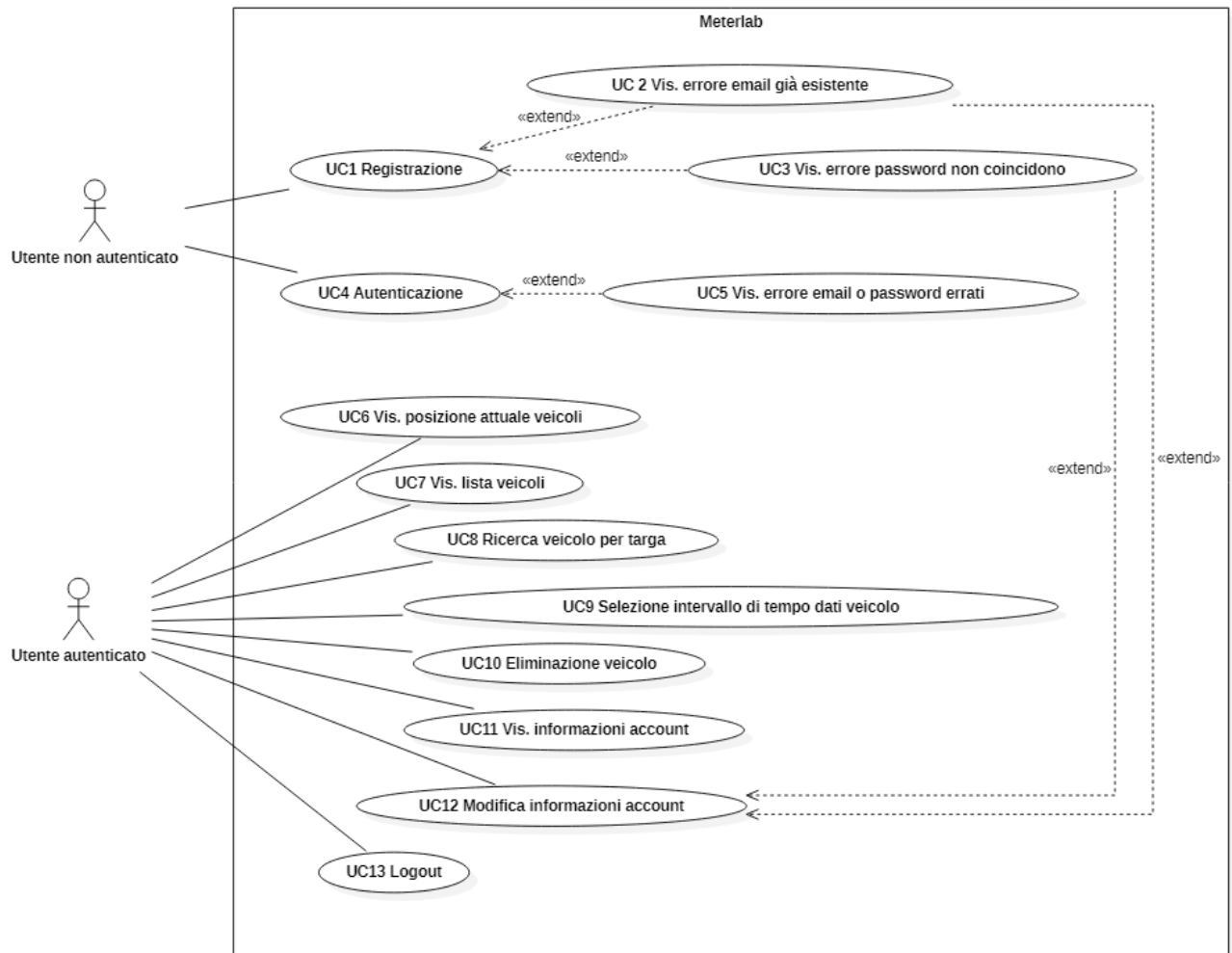


Figura 3.1: Scenario principale

UC1: Registrazione

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.

Descrizione: L'utente vuole registrarsi sull'applicazione web.

Postcondizioni: L'utente è registrato e autenticato sull'applicazione web.

Scenario Principale:

1. L'utente inserisce il suo nome (UC1.1);
2. L'utente inserisce il suo cognome (UC1.2);
3. L'utente inserisce la sua email (UC1.3);
4. L'utente inserisce una password (UC1.4);

5. L'utente inserisce la conferma della password (UC1.5).

Estensioni:

1. Visualizzazione errore email già esistente nel sistema (UC2);
2. Visualizzazione errore password non coincidono (UC3).

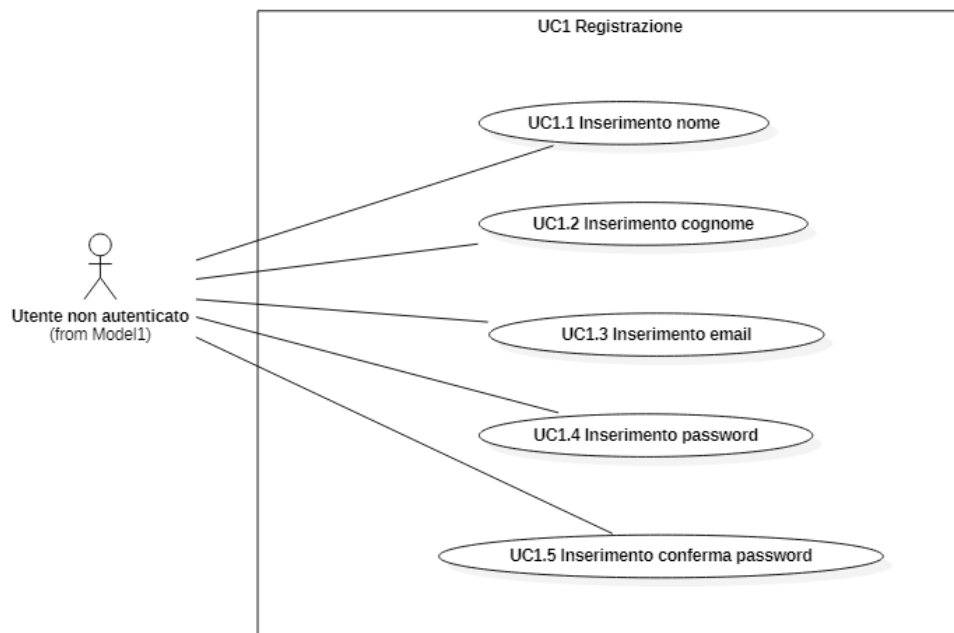


Figura 3.2: UC1: Registrazione

UC1.1: Inserimento nome

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.

Descrizione: L'utente deve inserire il suo nome per registrarsi.

Postcondizioni: L'utente ha inserito il suo nome.

Scenario Principale:

1. L'utente inserisce il suo nome.

UC1.2: Inserimento cognome

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.

Descrizione: L'utente deve inserire il suo cognome per registrarsi.

Postcondizioni: L'utente ha inserito il suo cognome.

Scenario Principale:

1. L'utente inserisce il suo cognome.

UC1.3: Inserimento email

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.

Descrizione: L'utente deve inserire la sua email per registrarsi.

Postcondizioni: L'utente ha inserito la sua email.

Scenario Principale:

1. L'utente inserisce la sua email.

UC1.4: Inserimento password

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.

Descrizione: L'utente deve inserire una password per registrarsi.

Postcondizioni: L'utente ha inserito una password.

Scenario Principale:

1. L'utente inserisce una password.

UC1.5: Inserimento conferma password

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.

Descrizione: L'utente deve inserire la conferma della password per registrarsi.

Postcondizioni: L'utente ha inserito la conferma della password.

Scenario Principale:

1. L'utente inserisce la conferma della password.

UC2: Vis. errore email già esistente

Attori Principali: Utente non autenticato.

Precondizioni: L'utente sta cercando di registrarsi nell'applicazione web.

Descrizione: L'utente vuole registrarsi ma inserisce una email già utilizzata da un altro utente.

Postcondizioni: L'utente visualizza un errore che lo avvisa che l'email è già utilizzata.

Scenario Principale:

1. L'utente cerca di registrarsi con un'email già utilizzata;
2. Il sistema avvisa l'utente che l'email è già utilizzata.

UC3: Vis. errore password non coincidono

Attori Principali: Utente non autenticato.

Precondizioni: L'utente sta cercando di registrarsi nell'applicazione web.

Descrizione: L'utente vuole registrarsi ma inserisce una password di conferma che non coincide con la password già inserita.

Postcondizioni: L'utente visualizza un errore che lo avvisa che le password non coincidono.

Scenario Principale:

1. L'utente cerca di registrarsi inserendo due password diverse;
2. Il sistema avvisa l'utente che le password non coincidono.

UC4: Autenticazione

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.

Descrizione: L'utente vuole autenticarsi sull'applicazione web.

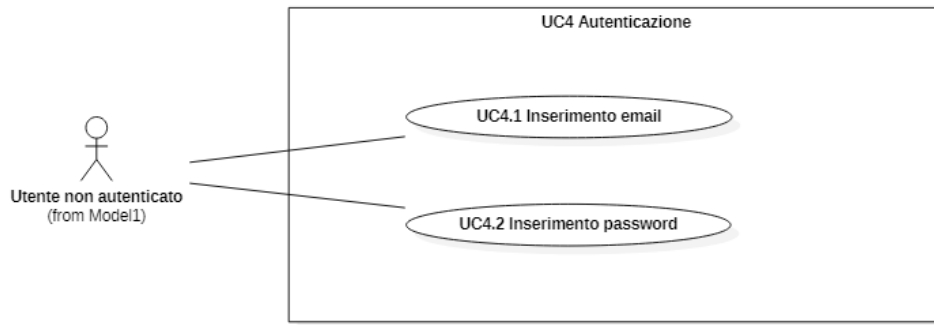
Postcondizioni: L'utente è autenticato sull'applicazione web.

Scenario Principale:

1. L'utente inserisce la sua email (UC4.1);
2. L'utente inserisce la sua password (UC4.2).

Estensioni:

1. Visualizzazione errore email o password errati (UC5);

**Figura 3.3:** UC4: Autenticazione

UC4.1: Inserimento email

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.

Descrizione: L'utente deve inserire la sua email per autenticarsi.

Postcondizioni: L'utente ha inserito la sua email.

Scenario Principale:

1. L'utente inserisce la sua email.

UC4.2: Inserimento password

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha aperto l'applicazione web e non è autenticato.

Descrizione: L'utente deve inserire la sua password per autenticarsi.

Postcondizioni: L'utente ha inserito la sua password.

Scenario Principale:

1. L'utente inserisce la sua password.

UC5: Vis. errore email o password errati

Attori Principali: Utente non autenticato.

Precondizioni: L'utente sta cercando di autenticarsi nell'applicazione web.

Descrizione: L'utente vuole autenticarsi ma inserisce una email e una password che non coincidono con nessun utente registrato.

Postcondizioni: L'utente visualizza un errore che lo avvisa che i dati inseriti sono errati.

Scenario Principale:

1. L'utente cerca di autenticarsi con dati errati;
2. Il sistema avvisa l'utente con un errore che i dati sono errati.

UC6: Vis. posizione attuale veicoli

Attori Principali: Utente autenticato.

Precondizioni: L'utente ha aperto l'applicazione web ed è autenticato.

Descrizione: L'utente vuole visualizzare la posizione attuale di tutti i veicoli monitorati.

Postcondizioni: L'utente ha visualizzato la posizione attuale di tutti i veicoli monitorati.

Scenario Principale:

1. L'utente visualizza la posizione attuale di tutti i veicoli monitorati.

UC7: Vis. lista veicoli

Attori Principali: Utente autenticato.

Precondizioni: L'utente ha aperto l'applicazione web ed è autenticato.

Descrizione: L'utente vuole visualizzare la lista di tutti i veicoli monitorati.

Postcondizioni: L'utente ha visualizzato la lista di tutti i veicoli monitorati.

Scenario Principale:

1. L'utente visualizza la lista dei veicoli.

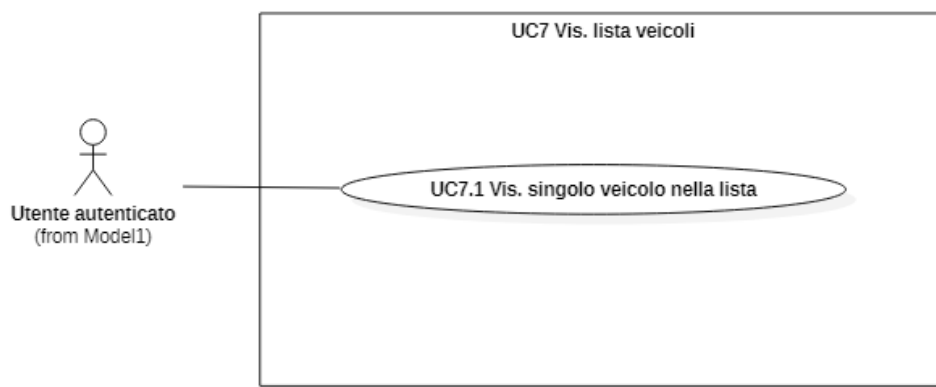


Figura 3.4: UC7: Vis. lista veicoli

UC7.1: Vis. singolo veicolo nella lista

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando la lista dei veicoli.

Descrizione: L'utente vuole visualizzare un veicolo specifico.

Postcondizioni: L'utente ha visualizzato un veicolo specifico.

Scenario Principale:

1. L'utente visualizza un veicolo dalla lista veicoli.

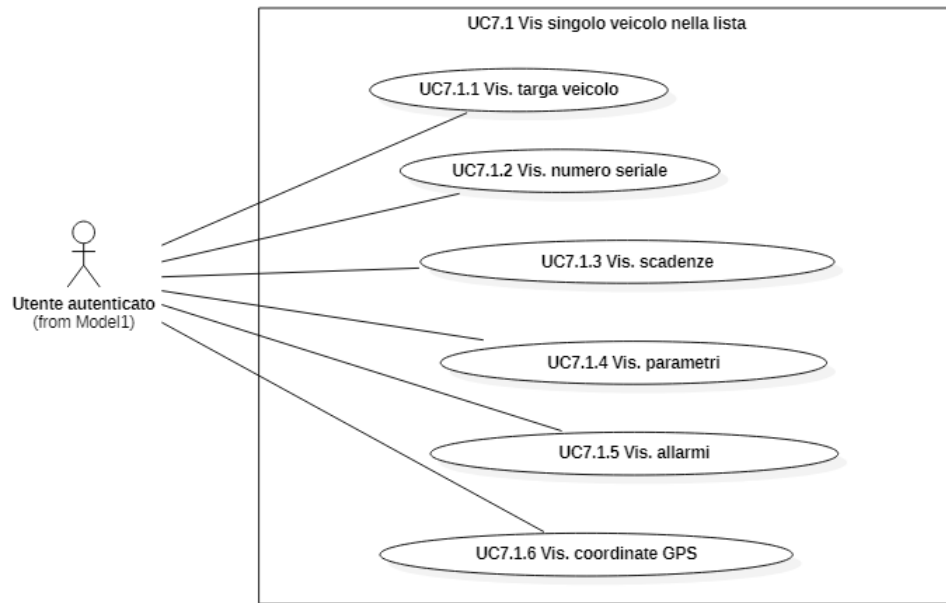


Figura 3.5: UC7.1: Vis. singolo veicolo nella lista

UC7.1.1: Vis. targa veicolo

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando un veicolo dalla lista dei veicoli.

Descrizione: L'utente vuole visualizzare la targa di un veicolo della lista.

Postcondizioni: L'utente ha visualizzato la targa di un veicolo della lista.

Scenario Principale:

1. L'utente visualizza la targa di un veicolo.

UC7.1.2: Vis. numero seriale

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando un veicolo dalla lista dei veicoli.

Descrizione: L'utente vuole visualizzare il numero seriale di un veicolo della lista.

Postcondizioni: L'utente ha visualizzato il numero seriale di un veicolo della lista.

Scenario Principale:

1. L'utente visualizza il numero seriale di un veicolo.

UC7.1.3: Vis. scadenze

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando un veicolo dalla lista dei veicoli.

Descrizione: L'utente vuole visualizzare le scadenze di un veicolo della lista.

Postcondizioni: L'utente ha visualizzato le scadenze di un veicolo della lista.

Scenario Principale:

1. L'utente visualizza la data dell'ultimo cambio dell'olio (UC7.1.3.1);
2. L'utente visualizza la data dell'ultima revisione (UC7.1.3.2).

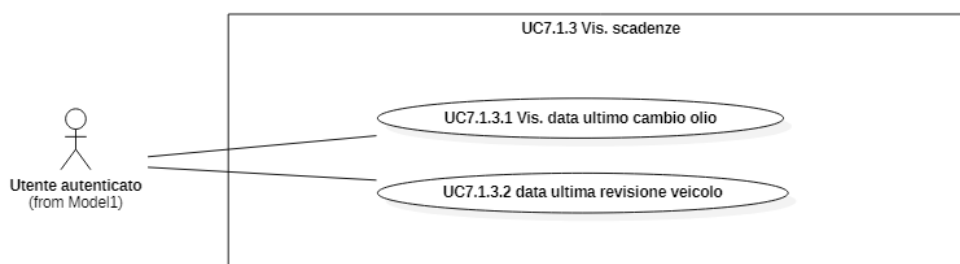


Figura 3.6: UC7.1.3: Vis. scadenze

UC7.1.3.1: Vis. data ultimo cambio olio

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando le scadenze di un veicolo.

Descrizione: L'utente vuole visualizzare la data dell'ultimo cambio dell'olio di un veicolo.

Postcondizioni: L'utente ha visualizzato la data dell'ultimo cambio dell'olio di un veicolo.

Scenario Principale:

1. L'utente visualizza la data dell'ultimo cambio dell'olio di un veicolo.

UC7.1.3.2: Vis. data ultima revisione veicolo

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando le scadenze di un veicolo.

Descrizione: L'utente vuole visualizzare la data dell'ultima revisione di un veicolo.

Postcondizioni: L'utente ha visualizzato la data dell'ultima revisione di un veicolo.

Scenario Principale:

1. L'utente visualizza la data dell'ultima revisione di un veicolo.

UC7.1.4: Vis. parametri

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato, ha selezionato un intervallo di tempo di cui visualizzare i dati e sta visualizzando un veicolo dalla lista dei veicoli.

Descrizione: L'utente vuole visualizzare i parametri delle ore lavorate e altri valori di un veicolo della lista.

Postcondizioni: L'utente ha visualizzato i parametri di un veicolo della lista.

Scenario Principale:

1. L'utente visualizza i giorni passati dall'ultimo cambio delle spazzole (UC7.1.4.1);
2. L'utente visualizza il chilometraggio del veicolo (UC7.1.4.2);
3. L'utente visualizza le ore di accensione del veicolo (UC7.1.4.3);
4. L'utente visualizza le ore di lavorazione del veicolo (UC7.1.4.4);
5. L'utente visualizza le ore di trasferimento del veicolo (UC7.1.4.5);
6. L'utente visualizza le ore di funzionamento del trituratore (UC7.1.4.6).

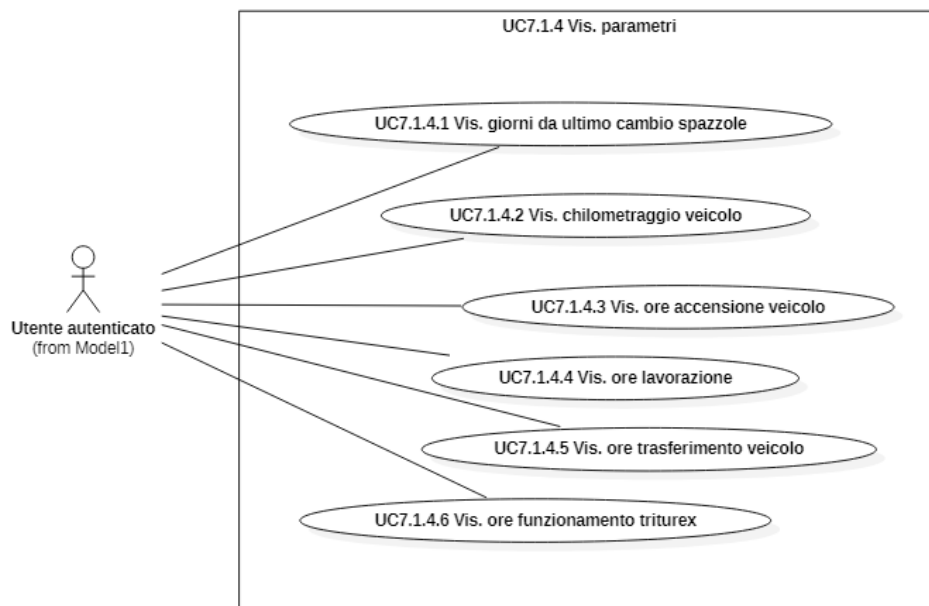


Figura 3.7: UC7.1.4: Vis. parametri

UC7.1.4.1: Vis. giorni da ultimo cambio spazzole

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando i parametri di un veicolo.

Descrizione: L'utente vuole visualizzare i giorni passati dall'ultimo cambio delle spazzole di un veicolo.

Postcondizioni: L'utente ha visualizzato i giorni passati dall'ultimo cambio delle spazzole di un veicolo.

Scenario Principale:

1. L'utente visualizza i giorni passati dall'ultimo cambio delle spazzole di un veicolo.

UC7.1.4.2: Vis. chilometraggio veicolo

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando i parametri di un veicolo.

Descrizione: L'utente vuole visualizzare il chilometraggio di un veicolo.

Postcondizioni: L'utente ha visualizzato il chilometraggio di un veicolo.

Scenario Principale:

1. L'utente visualizza il chilometraggio di un veicolo.

UC7.1.4.3: Vis. ore accensione veicolo

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando i parametri di un veicolo.

Descrizione: L'utente vuole visualizzare le ore di accensione del motore di un veicolo in quel giorno.

Postcondizioni: L'utente ha visualizzato le ore di accensione del motore di un veicolo in quel giorno.

Scenario Principale:

1. L'utente visualizza le ore di accensione del motore di un veicolo in quel giorno.

UC7.1.4.4: Vis. ore lavorazione veicolo

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando i parametri di un veicolo.

Descrizione: L'utente vuole visualizzare le ore di lavorazione di un veicolo in quel giorno. Le ore di lavorazione sono intese come ore di pulizia, ovvero gli strumenti adeguati sono attivi.

Postcondizioni: L'utente ha visualizzato le ore di lavorazione di un veicolo in quel giorno.

Scenario Principale:

1. L'utente visualizza le ore di lavorazione di un veicolo in quel giorno.

UC7.1.4.5: Vis. ore trasferimento veicolo

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando i parametri di un veicolo.

Descrizione: L'utente vuole visualizzare le ore di trasferimento di un veicolo in quel giorno. Le ore di trasferimento sono intese come ore in cui il veicolo si sposta senza pulire la strada.

Postcondizioni: L'utente ha visualizzato le ore di trasferimento di un veicolo in quel giorno.

Scenario Principale:

1. L'utente visualizza le ore di trasferimento di un veicolo in quel giorno.

UC7.1.4.6: Vis. ore funzionamento triturex

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando i parametri di un veicolo.

Descrizione: L'utente vuole visualizzare le ore di funzionamento del triturex di un veicolo in quel giorno..

Postcondizioni: L'utente ha visualizzato le ore di funzionamento del triturex di un veicolo in quel giorno.

Scenario Principale:

1. L'utente visualizza le ore di funzionamento del triturex di un veicolo in quel giorno.

UC7.1.5: Vis. allarmi

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando un veicolo dalla lista dei veicoli.

Descrizione: L'utente vuole visualizzare gli allarmi del motore di un veicolo della lista.

Postcondizioni: L'utente ha visualizzato gli allarmi di un veicolo della lista.

Scenario Principale:

1. L'utente visualizza l'allarme del livello dell'acqua (UC7.1.5.1);
2. L'utente visualizza l'allarme del filtro dell'aria (UC7.1.5.2);
3. L'utente visualizza l'allarme del filtro motore intasato (UC7.1.5.3);
4. L'utente visualizza l'allarme della temperatura dell'acqua (UC7.1.5.4);
5. L'utente visualizza l'allarme della pressione dell'olio (UC7.1.5.5);
6. L'utente visualizza l'allarme del livello dell'olio (UC7.1.5.6);
7. L'utente visualizza l'allarme della pompa dell'olio (UC7.1.5.7);
8. L'utente visualizza l'allarme della batteria (UC7.1.5.8).

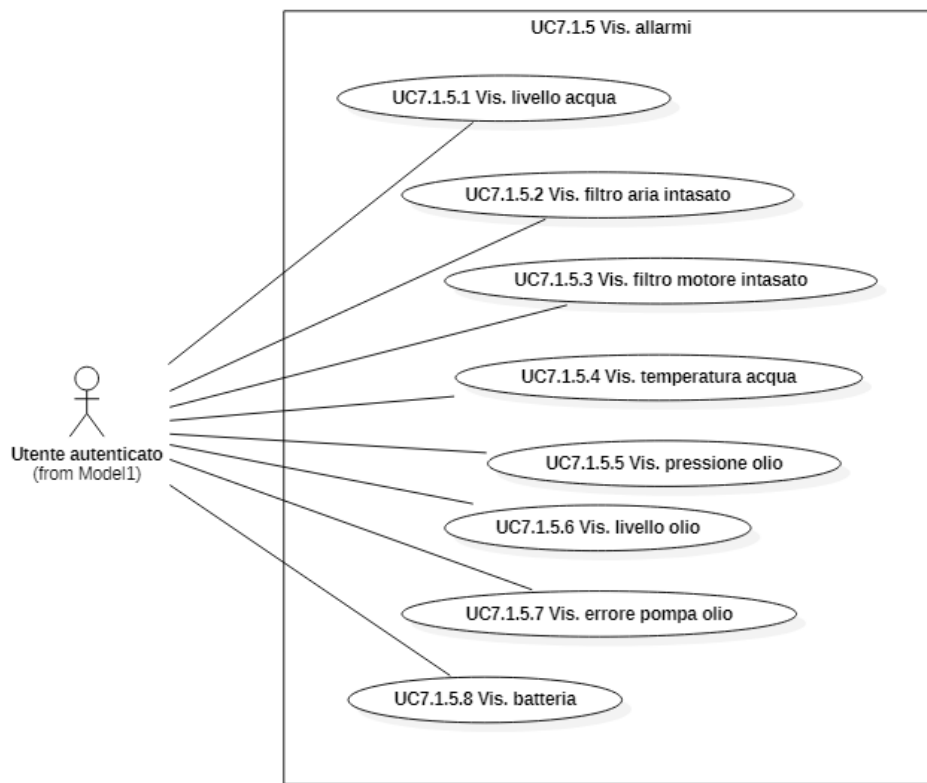


Figura 3.8: UC7.1.5: Vis. allarmi

UC7.1.5.1: Vis. livello acqua

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando gli allarmi di un veicolo.

Descrizione: L'utente vuole visualizzare se l'allarme del livello dell'acqua di un veicolo è al momento attivo..

Postcondizioni: L'utente ha visualizzato se l'allarme del livello dell'acqua di un veicolo è attivo.

Scenario Principale:

1. L'utente visualizza se l'allarme del livello dell'acqua di un veicolo è attivo.

UC7.1.5.2: Vis. filtro aria intasato

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando gli allarmi di un veicolo.

Descrizione: L'utente vuole visualizzare se l'allarme di filtro intasato di un veicolo è al momento attivo..

Postcondizioni: L'utente ha visualizzato se l'allarme di filtro intasato di un veicolo è

attivo.

Scenario Principale:

1. L'utente visualizza se l'allarme di filtro intasato di un veicolo è attivo.

UC7.1.5.3: Vis. filtro motore intasato

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando gli allarmi di un veicolo.

Descrizione: L'utente vuole visualizzare se l'allarme di motore intasato di un veicolo è al momento attivo..

Postcondizioni: L'utente ha visualizzato se l'allarme di motore intasato di un veicolo è attivo.

Scenario Principale:

1. L'utente visualizza se l'allarme di motore intasato di un veicolo è attivo.

UC7.1.5.4: Vis. temperatura acqua

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando gli allarmi di un veicolo.

Descrizione: L'utente vuole visualizzare se l'allarme della temperatura dell'acqua di un veicolo è al momento attivo..

Postcondizioni: L'utente ha visualizzato se l'allarme della temperatura dell'acqua di un veicolo è attivo.

Scenario Principale:

1. L'utente visualizza se l'allarme della temperatura dell'acqua di un veicolo è attivo.

UC7.1.5.5: Vis. pressione olio

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando gli allarmi di un veicolo.

Descrizione: L'utente vuole visualizzare se l'allarme della pressione dell'olio di un veicolo è al momento attivo..

Postcondizioni: L'utente ha visualizzato se l'allarme della pressione dell'olio di un veicolo è attivo.

Scenario Principale:

1. L'utente visualizza se l'allarme della pressione dell'olio di un veicolo è attivo.

UC7.1.5.6: Vis. livello olio

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando gli allarmi di un veicolo.

Descrizione: L'utente vuole visualizzare se l'allarme del livello dell'olio di un veicolo è al momento attivo..

Postcondizioni: L'utente ha visualizzato se l'allarme del livello dell'olio di un veicolo è attivo.

Scenario Principale:

1. L'utente visualizza se l'allarme del livello dell'olio di un veicolo è attivo.

UC7.1.5.7: Vis. errore pompa olio

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando gli allarmi di un veicolo.

Descrizione: L'utente vuole visualizzare se l'allarme di un errore della pompa dell'olio di un veicolo è al momento attivo..

Postcondizioni: L'utente ha visualizzato se l'allarme di un errore della pompa dell'olio di un veicolo è attivo.

Scenario Principale:

1. L'utente visualizza se l'allarme di un errore della pompa dell'olio di un veicolo è attivo.

UC7.1.5.8: Vis. batteria

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando gli allarmi di un veicolo.

Descrizione: L'utente vuole visualizzare se l'allarme della batteria di un veicolo è al momento attivo..

Postcondizioni: L'utente ha visualizzato se l'allarme della batteria di un veicolo è attivo.

Scenario Principale:

1. L'utente visualizza se l'allarme della batteria di un veicolo è attivo.

UC7.1.6: Vis. coordinate GPS

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando un veicolo dalla lista dei veicoli.

Descrizione: L'utente vuole visualizzare le coordinate [Global Positioning System \(GPS\)](#)_G di un veicolo della lista.

Postcondizioni: L'utente ha visualizzato le coordinate [GPS](#) di un veicolo della lista.

Scenario Principale:

1. L'utente visualizza le coordinate [GPS](#) di un veicolo.

UC8: Ricerca veicolo per targa

Attori Principali: Utente autenticato.

Precondizioni: L'utente ha aperto l'applicazione web ed è autenticato.

Descrizione: L'utente vuole cercare un veicolo in base alla sua targa.

Postcondizioni: L'utente ha cercato un veicolo in base alla sua targa.

Scenario Principale:

1. L'utente cerca un veicolo in base alla sua targa.

UC9: Selezione intervallo di tempo dati veicolo

Attori Principali: Utente autenticato.

Precondizioni: L'utente ha aperto l'applicazione web ed è autenticato.

Descrizione: L'utente vuole selezionare un intervallo di date a seconda del quale vengono visualizzati i dati dei veicoli.

Postcondizioni: L'utente ha selezionato un intervallo di tempo.

Scenario Principale:

1. L'utente seleziona un intervallo di tempo.

UC10: Eliminazione veicolo

Attori Principali: Utente autenticato.

Precondizioni: L'utente ha aperto l'applicazione web ed è autenticato.

Descrizione: L'utente vuole eliminare un veicolo.

Postcondizioni: L'utente ha eliminato il veicolo voluto.

Scenario Principale:

1. L'utente elimina un veicolo.

UC11: Vis. informazioni account

Attori Principali: Utente autenticato.

Precondizioni: L'utente ha aperto l'applicazione web ed è autenticato.

Descrizione: L'utente vuole visualizzare le informazioni del proprio account.

Postcondizioni: L'utente ha visualizzato le informazioni del proprio account.

Scenario Principale:

1. L'utente visualizza il proprio nome (UC11.1);
2. L'utente visualizza il proprio cognome (UC11.2);
3. L'utente visualizza la propria email (UC11.3).

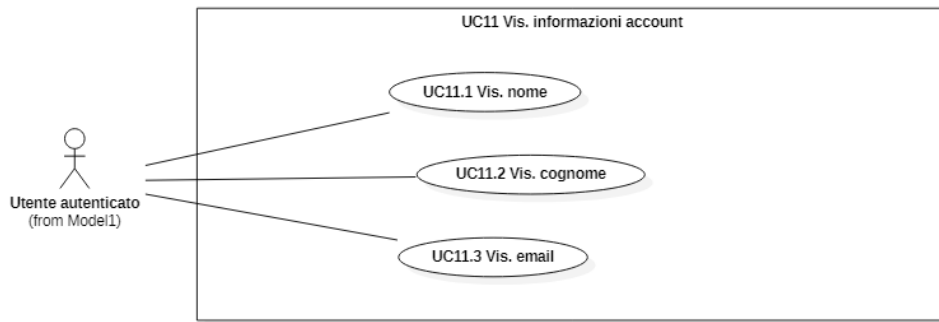


Figura 3.9: UC11: Vis. informazioni account

UC11.1: Vis. nome

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando le informazioni dell'account.

Descrizione: L'utente vuole visualizzare il nome associato al suo account.

Postcondizioni: L'utente ha visualizzato il nome associato al suo account.

Scenario Principale:

1. L'utente visualizza il nome associato al suo account.

UC11.2: Vis. cognome

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando le informazioni dell'account.

Descrizione: L'utente vuole visualizzare il cognome associato al suo account.

Postcondizioni: L'utente ha visualizzato il cognome associato al suo account.

Scenario Principale:

1. L'utente visualizza il cognome associato al suo account.

UC11.3: Vis. email

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando le informazioni dell'account.

Descrizione: L'utente vuole visualizzare l'email associata al suo account.

Postcondizioni: L'utente ha visualizzato l'email associata al suo account.

Scenario Principale:

1. L'utente visualizza l'email associata al suo account.

UC12: Modifica informazioni account

Attori Principali: Utente autenticato.

Precondizioni: L'utente ha aperto l'applicazione web ed è autenticato.

Descrizione: L'utente vuole modificare le informazioni del proprio account.

Postcondizioni: L'utente ha modificato le informazioni del proprio account.

Scenario Principale:

1. L'utente modifica le informazioni del proprio account.

Estensioni:

1. Visualizzazione errore email già esistente nel sistema (UC2);
2. Visualizzazione errore password non coincidono (UC3).

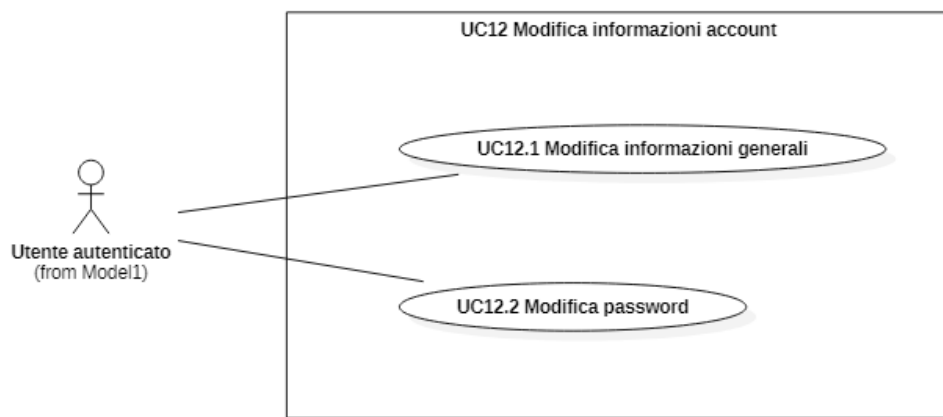


Figura 3.10: UC12: Modifica informazioni account

UC12.1: Modifica informazioni generali

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e vuole modificare le informazioni dell'account.

Descrizione: L'utente vuole modificare le informazioni generali del proprio account come nome e cognome.

Postcondizioni: L'utente ha modificato le informazioni generali del proprio account.

Scenario Principale:

1. L'utente modifica il nome (UC12.1.1);
2. L'utente modifica il cognome (UC12.1.2);
3. L'utente modifica l'email (UC12.1.3).

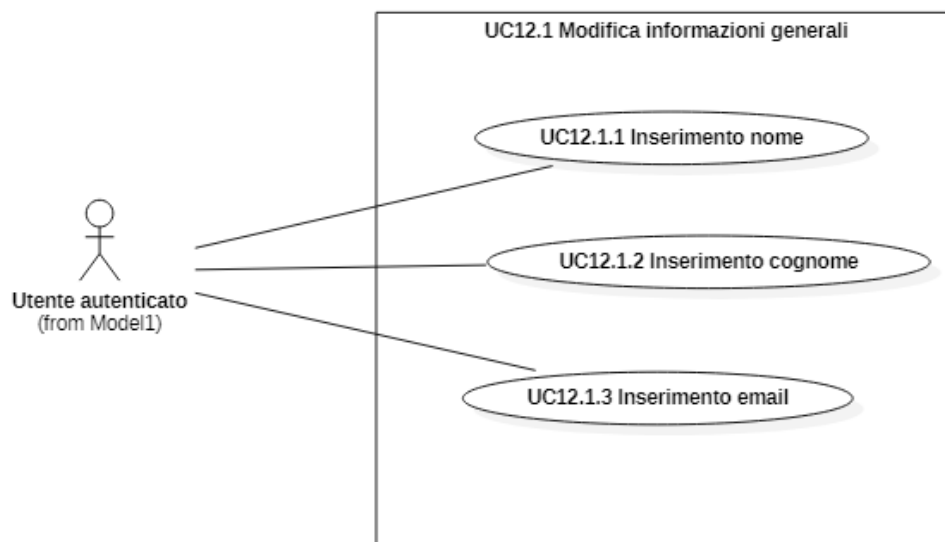


Figura 3.11: UC12.1: Modifica informazioni generali

UC12.1.1: Inserimento nome

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e vuole modificare le informazioni generali dell'account.

Descrizione: L'utente vuole modificare il nome associato al suo account.

Postcondizioni: L'utente ha modificato il nome associato al suo account.

Scenario Principale:

1. L'utente modifica il nome associato al suo account.

UC12.1.2: Inserimento cognome

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e vuole modificare le informazioni generali dell'account.

Descrizione: L'utente vuole modificare il cognome associato al suo account.

Postcondizioni: L'utente ha modificato il cognome associato al suo account.

Scenario Principale:

1. L'utente modifica il cognome associato al suo account.

UC12.1.3: Inserimento email

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e vuole modificare le informazioni generali

dell'account.

Descrizione: L'utente vuole modificare l'email associata al suo account.

Postcondizioni: L'utente ha modificato l'email associata al suo account.

Scenario Principale:

1. L'utente modifica l'email associata al suo account.

UC12.2: Modifica password

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e vuole modificare le informazioni dell'account.

Descrizione: L'utente vuole modificare la password usata per autenticarsi.

Postcondizioni: L'utente ha modificato la password del proprio account.

Scenario Principale:

1. L'utente inserisce la password attuale (UC12.2.1);
2. L'utente inserisce la nuova password (UC12.2.2);
3. L'utente inserisce la conferma della nuova password (UC12.2.3).

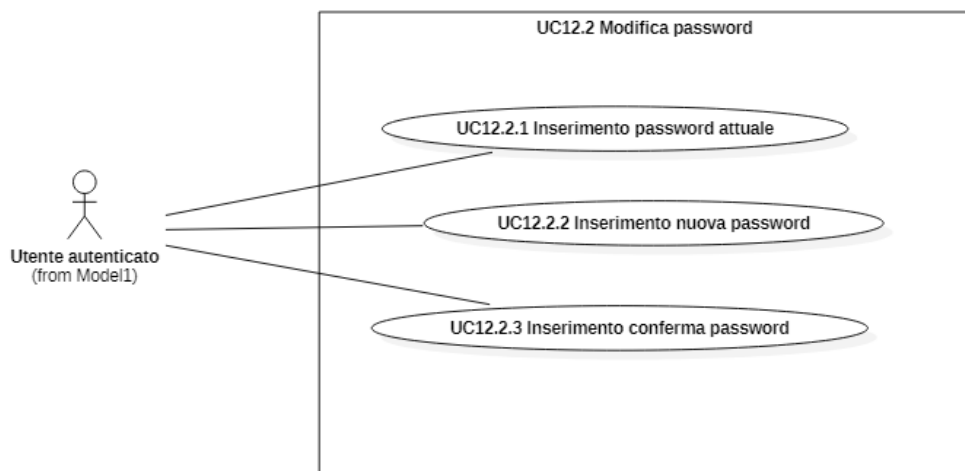


Figura 3.12: UC12.2: Modifica password

UC12.2.1: Inserimento password attuale

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e vuole modificare la password dell'account.

Descrizione: L'utente vuole modificare la password associata al suo account. Per

farlo deve prima inserire la password attuale per assicurare al sistema di essere lui il proprietario dell'account.

Postcondizioni: L'utente ha inserito la password attuale del suo account.

Scenario Principale:

1. L'utente inserisce la password attuale.

UC12.1.2: Inserimento nuova password

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e vuole modificare la password dell'account.

Descrizione: L'utente vuole modificare la password associata al suo account. Per farlo deve inserire la nuova password che desidera impostare.

Postcondizioni: L'utente ha inserito la nuova password.

Scenario Principale:

1. L'utente inserisce la nuova password.

UC12.1.3: Inserimento conferma password

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e vuole modificare la password dell'account.

Descrizione: L'utente vuole modificare la password associata al suo account. Per farlo deve confermare la nuova password così da assicurarsi di non aver sbagliato a digitare.

Postcondizioni: L'utente ha inserito la conferma della password.

Scenario Principale:

1. L'utente inserisce la conferma della password.

UC13: Logout

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e vuole dal proprio account.

Descrizione: L'utente vuole uscire dal proprio account così che chiunque usi l'applicazione dopo di lui non abbia accesso al suo account.

Postcondizioni: L'utente non è più autenticato.

Scenario Principale:

1. L'utente esce dal proprio account.

3.2 Tracciamento dei requisiti

Da un'attenta analisi dei requisiti e degli use case effettuata sul progetto è stata stilata la tabella che traccia i requisiti in rapporto agli use case.

Sono stati individuati diversi tipi di requisiti e si è quindi fatto utilizzo di un codice identificativo per distinguerli.

Il codice dei requisiti è così strutturato $R[F/Q/V][N/D/O]$ dove:

R = requisito

F = funzionale

Q = qualitativo

V = di vincolo

N = obbligatorio (necessario)

D = desiderabile

Z = opzionale

Le fonti dei requisiti possono essere le seguenti:

- UC[numero]: indica un caso d'uso;
- Committente: indica il capo progetto;
- Interno: indica il gruppo di stagisti.

Nella tabella [3.1](#), [3.2](#) e [3.3](#) sono riassunti i requisiti e il loro tracciamento con le fonti delineate in fase di analisi.

Tabella 3.1: Tabella del tracciamento dei requisiti funzionali

Requisito	Descrizione	Fonte
RFO-1	Il sistema deve permettere la registrazione di un nuovo utente	UC1
RFO-1.1	Il sistema deve permettere di inserire il nome in fase di registrazione	UC1.1
RFO-1.2	Il sistema deve permettere di inserire il cognome in fase di registrazione	UC1.2
RFO-1.3	Il sistema deve permettere di inserire l'email in fase di registrazione	UC1.3
RFO-1.4	Il sistema deve permettere di inserire la password in fase di registrazione	UC1.4
RFO-1.5	Il sistema deve permettere di confermare la password in fase di registrazione	UC1.5
RFO-2	Il sistema deve visualizzare l'errore "email già esistente"	UC2
RFO-3	Il sistema deve visualizzare l'errore "password non coincidono"	UC3
RFO-4	Il sistema deve permettere l'autenticazione ad un utente già registrato	UC4
RFO-4.1	Il sistema deve permettere di inserire la mail in fase di autenticazione	UC4.1
RFO-4.2	Il sistema deve permettere di inserire la password in fase di autenticazione	UC4.2
RFO-5	Il sistema deve visualizzare l'errore "email o password errati"	UC5
RFO-6	Il sistema deve permettere la visualizzazione delle posizioni attuali di tutti i veicoli	UC6
RFO-7	Il sistema deve permettere la visualizzazione della lista dei veicoli monitorati	UC7
RFO-7.1	Il sistema deve permettere la visualizzazione delle informazioni di un singolo veicolo della lista	UC7.1
RFO-7.1.1	Il sistema deve permettere la visualizzazione della targa di un singolo veicolo della lista	UC7.1.1

Continuazione della tabella 3.1		
Requisito	Descrizione	Fonte
RFO-7.1.2	Il sistema deve permettere la visualizzazione del numero seriale di un singolo veicolo della lista	UC7.1.1
RFO-7.1.3	Il sistema deve permettere la visualizzazione delle scadenze di un singolo veicolo della lista	UC7.1.3
RFO-7.1.3.1	Il sistema deve permettere la visualizzazione della data di ultimo cambio dell'olio di un veicolo	UC7.1.3.1
RFO-7.1.3.2	Il sistema deve permettere la visualizzazione della data di ultima revisione di un veicolo	UC7.1.3.2
RFO-7.1.4	Il sistema deve permettere la visualizzazione dei parametri di un singolo veicolo della lista	UC7.1.4
RFO-7.1.4.1	Il sistema deve permettere la visualizzazione dei giorni passati dall'ultimo cambio delle spazzole di un singolo veicolo	UC7.1.4.1
RFO-7.1.4.2	Il sistema deve permettere la visualizzazione del chilometraggio di un singolo veicolo	UC7.1.4.2
RFO-7.1.4.3	Il sistema deve permettere la visualizzazione delle ore di accensione del motore di un singolo veicolo	UC7.1.4.3
RFO-7.1.4.4	Il sistema deve permettere la visualizzazione delle ore di lavorazione di un singolo veicolo	UC7.1.4.4
RFO-7.1.4.5	Il sistema deve permettere la visualizzazione delle ore di trasferimento di un singolo veicolo	UC7.1.4.5
RFO-7.1.4.6	Il sistema deve permettere la visualizzazione delle ore funzionamento del trituratore di un singolo veicolo	UC7.1.4.6
RFO-7.1.5	Il sistema deve permettere la visualizzazione degli allarmi di un singolo veicolo della lista	UC7.1.5
RFO-7.1.5.1	Il sistema deve permettere la visualizzazione dell'allarme del livello dell'acqua di un singolo veicolo	UC7.1.5.1
RFO-7.1.5.2	Il sistema deve permettere la visualizzazione dell'allarme del filtro dell'aria intasato di un singolo veicolo	UC7.1.5.2
RFO-7.1.5.3	Il sistema deve permettere la visualizzazione dell'allarme del motore intasato di un singolo veicolo	UC7.1.5.3

Continuazione della tabella 3.1		
Requisito	Descrizione	Fonte
RFO-7.1.5.4	Il sistema deve permettere la visualizzazione dell'allarme della temperatura dell'acqua di un singolo veicolo	UC7.1.5.4
RFO-7.1.5.5	Il sistema deve permettere la visualizzazione della pressione dell'olio di un singolo veicolo	UC7.1.5.5
RFO-7.1.5.6	Il sistema deve permettere la visualizzazione dell'allarme del livello dell'olio di un singolo veicolo	UC7.1.5.6
RFO-7.1.5.7	Il sistema deve permettere la visualizzazione dell'allarme dell'errore della pompa dell'olio di un singolo veicolo	UC7.1.5.7
RFO-7.1.5.8	Il sistema deve permettere la visualizzazione dell'allarme della batteria di un singolo veicolo	UC7.1.5.8
RFO-7.1.6	Il sistema deve permettere la visualizzazione delle coordinate GPS di un singolo veicolo della lista	UC7.1.6
RFO-8	Il sistema deve permettere la ricerca di un veicolo in base alla targa	UC8
RFO-9	Il sistema deve permettere la selezione di un intervallo di tempo di cui vedere i dati dei veicoli	UC9
RFO-10	Il sistema deve permettere l'eliminazione di un veicolo	UC10
RFD-11	Il sistema deve permettere la visualizzazione delle informazioni di un account autenticato	UC11
RFD-11.1	Il sistema deve permettere la visualizzazione del nome di un account autenticato	UC11.1
RFD-11.2	Il sistema deve permettere la visualizzazione del cognome di un account autenticato	UC11.2
RFD-11.3	Il sistema deve permettere la visualizzazione dell'email di un account autenticato	UC11.3
RFD-12	Il sistema deve permettere la modifica delle informazioni di un account autenticato	UC12
RFD-12.1	Il sistema deve permettere la modifica delle informazioni generali di un account autenticato	UC12.1

Continuazione della tabella 3.1		
Requisito	Descrizione	Fonte
RFD-12.1.1	Il sistema deve permettere la modifica del nome un account autenticato	UC12.1.1
RFD-12.1.2	Il sistema deve permettere la modifica del cognome un account autenticato	UC12.1.2
RFD-12.1.1	Il sistema deve permettere la modifica dell'email un account autenticato	UC12.1.1
RFD-12.2	Il sistema deve permettere la modifica della password di un account autenticato	UC12.2
RFD-12.2.1	Il sistema deve permettere l'inserimento della password attuale	UC12.2.1
RFD-12.2.2	Il sistema deve permettere l'inserimento della nuova password	UC12.2.2
RFD-12.2.3	Il sistema deve permettere l'inserimento della conferma della nuova password	UC12.2.3
RFO-13	Il sistema deve permettere all'utente di uscire dal proprio account	UC13

Tabella 3.2: Tabella del tracciamento dei requisiti qualitativi

Requisito	Descrizione	Fonte
RQO-1	Deve essere fornito un documento tecnico che descriva le maschere	Committente
RQO-2	Il codice deve essere coperto da test di unità	Interno
RQD-3	L'applicazione web deve essere accessibile a utenti con disabilità	Interno
RQD-4	L'applicazione web deve essere responsive	Interno

Tabella 3.3: Tabella del tracciamento dei requisiti di vincolo

Requisito	Descrizione	Fonte
RVD-1	La maschera della lista dei veicoli e delle informazioni deve essere sviluppata in un'unica pagina senza scorrimento verticale o orizzontale	Committente
RVD-2	La maschera delle informazioni di un veicolo deve essere il più 'personalizzabile' possibile permettendo all'utente di nascondere la lista dei veicoli e ingrandire a tutto schermo mappe e grafici	Committente

Capitolo 4

Progettazione e codifica

In questo capitolo vengono descritte in primo luogo le tecnologie utilizzate durante il progetto e in secondo luogo la progettazione e l'implementazione delle maschere e dei servizi dell'applicazione.

4.1 Tecnologie

Typescript

Typescript è un linguaggio di programmazione open source che basa le sue caratteristiche su ECMAScript 6. Il linguaggio estende la sintassi del linguaggio javascript facendo sì che qualsiasi programma scritto in javascript possa funzionare anche con typescript. Tra le principali funzionalità aggiuntive di typescript ci sono la possibilità di tipizzare le variabili e di creare interfacce e classi [8].

Angular

Angular è un framework javascript open source per creare applicazioni web dinamiche grazie a una serie di funzionalità e strumenti forniti dallo stesso. L'architettura modulare consente di strutturare al meglio un'applicazione e di avere un elevato riutilizzo del codice. Permette inoltre un'elevata manutenibilità in quanto ogni componente è adibito ad un'unica funzione [1]. Utilizzato per sviluppare l'applicazione web.

Node.js

Node.js è un framework utilizzato da Angular per gestire le dipendenze. Permette di dichiarare due insiemi di dipendenze: per gli sviluppatori e per far funzionare l'applicativo. In questo modo è possibile differenziare quali librerie si possono tralasciare in fase di deploy dell'applicazione perchè, ad esempio, necessarie solo per effettuare i test. Permette inoltre, usando dei semplici comandi, di scaricare all'interno del progetto le librerie necessarie, funzionalità molto utile in fase di [Continuous Integration/Continuous Delivery \(CI/CD\)](#)_G .

Angular Material

Angular Material è una libreria grafica creata appositamente per Angular ed è basata sullo stile grafico di Google. Mette a disposizione componenti grafiche già fatte e testate anche dal punto di vista dell'accessibilità [2]. Utilizzato per sviluppare la grafica delle pagine web.

Jasmine

Jasmine è un framework per eseguire test di codice javascript. I test vengono inseriti all'interno di file con estensione *.spec.ts e descritti uno a uno con `describe(...)` così che in caso di fallimento si possa velocemente capire cosa tale test stava testando [5]. Utilizzato durante lo sviluppo dell'applicazione web per scrivere i test di unità e assicurarsi il corretto funzionamento del prodotto.

4.2 Progettazione

4.2.1 Architettura di Angular

Il componente principale di Angular è il modulo, che raggruppa un insieme di funzionalità legate tra loro. Ogni modulo contiene un component alla quale è associato un template.

Un component è una classe che si occupa di gestire la vista e definire la logica del codice, mentre il template è la vista stessa dove viene definito codice HTML per visualizzare i dati contenuti nel component. Questi due elementi lavorano a stretto contatto con altri due componenti di Angular rispettivamente: servizi e direttive.

I servizi vengono richiamati dai component e svolgono compiti ben precisi come ad esempio la gestione dell'autenticazione utente.

Le direttive vengono richiamate dai template e permettono di personalizzare il codice HTML creando dei tag propri che è poi possibile riutilizzare in diverse viste.

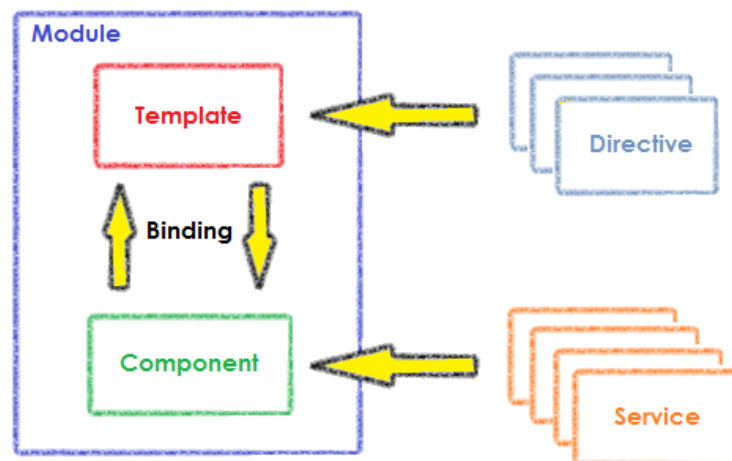


Figura 4.1: Schema logico che rappresenta l'architettura di un'applicazione Angular
(Fonte: [6])

Ogni component può contenere diversi component figli, in questo modo si può creare una maschera in modo modulare. Si può creare un component padre che rappresenta la pagina completa e poi diversi figli per ogni elemento contenuto in quella pagina, per esempio un component per il menu, un component per la sezione delle notizie e così via. Ognuno dei component figli si occuperà così di gestire la grafica di quella determinata funzionalità e di chiamare i servizi di cui necessita e sarà poi compito del padre mettere i figli insieme. Così facendo, si rende la struttura dell'applicazione più ordinata e il codice più manutenibile.

4.2.2 Architettura dell'applicazione sviluppata

L'architettura dell'applicazione è stata progettata seguendo le best practise già consolidate nell'ambiente di Angular.

Ogni maschera dell'applicazione deve avere una propria cartella contenente il component, il template e il file di test ed eventualmente altri component figli.

Ogni maschera deve inoltre avere un proprio modulo così da poter implementare il lazy loading delle componenti di cui è composta.

Sempre in concordanza con le best practise sono state create due cartelle: Core e Shared.

La cartella Core contiene tutti i servizi e componenti importanti per ogni maschera come ad esempio i servizi e le classi degli oggetti.

La cartella Shared contiene tutti quei componenti o direttive che sono condivise tra più maschere come ad esempio il menu che rimane invariato da pagina a pagina.

4.2.3 Progettazione delle maschere

Dopo aver avuto un incontro con il committente in cui spiegava quali fossero le funzionalità che desiderava, abbiamo sviluppato un mock-up di quelle che potevano essere le pagine dell'applicazione così da poter avere un riscontro prima di metterci a lavorare.

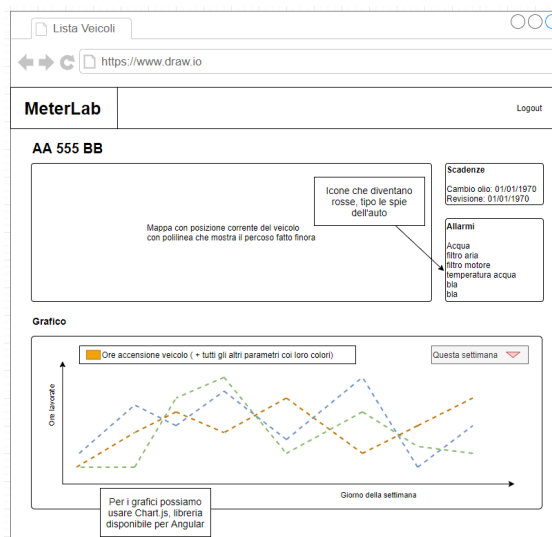


Figura 4.2: Mock-up della pagina delle informazioni di un veicolo

In fase di progettazione delle maschere si è deciso che la struttura delle pagine deve essere simile ad altre applicazioni web di uso comune così da non disorientare l'utente. Tra le regole seguite:

- Barra di navigazione sul lato sinistro o superiore della pagina;
- Bottoni per le funzioni di accesso, disconnessione e impostazioni account in alto a destra;
- Barra di ricerca posizionata direttamente sopra la lista degli elementi da filtrare;
- Uso di icone ampiamente conosciute dagli utenti per facilitare la navigazione;
- I bottoni di eliminazione devono avere colore rosso per avvertire l'utente che provocherà un'azione distruttiva;
- I bottoni disabilitati devono avere un colore diverso dal loro stato normale così da far capire all'utente chiaramente quale sia il loro stato;
- I campi dei form che risultano errati devono essere segnalati in colore rosso e con un messaggio di errore per avvisare l'utente che per procedere devono essere corretti.

4.2.4 Progettazione delle API

Una volta completata la progettazione delle maschere e avuto chiaro quali dati era necessario visualizzare e quali dati sarebbe stato utile ricevere con chiamate HTTP dedicate si è proceduto con la progettazione delle [API](#). A tale scopo è stato utilizzato [stoplight studio](#) che, usando lo standard [OpenAPI](#), ci ha permesso di generare un file contenente tutte le informazioni degli [endpoint](#) che il server avrebbe messo a disposizione. Tra le informazioni più importanti:

- La lista degli [endpoint](#);
- I campi di autenticazione necessari a fare le richieste;
- La struttura delle risposte HTTP;
- I tipi di risposte che potevano essere restituite (e.g. 200 ok, 400 bad request, 500 internal server error).

In questo modo abbiamo stabilito un contratto che doveva essere rispettato sia da chi sviluppava il [front end](#) che da chi sviluppava il [back end](#) così che al momento dell'integrazione delle due parti non ci sarebbero stati problemi.

4.3 Design Pattern utilizzati

Dependency Injection

Il dependency injection è un pattern già integrato con Angular per migliorare l'efficienza e la modularità. Il dependency injection usato da Angular è di tipo constructor, questo prevede che in ogni componente vengano dichiarati nel costruttore di quali servizi ha bisogno e in fase di inizializzazione tali dipendenze gli vengono fornite dall'esterno. Questo pattern migliora molto anche la testabilità del codice in quanto permette di fornire servizi di mock evitando di, per esempio, inviare chiamate http al server vero e proprio.

Singleton

Il singleton è un pattern già integrato con Angular che ha lo scopo di garantire che di un determinato servizio venga creata una e una sola istanza. Un altro vantaggio è che i servizi, condivisi tra più componenti, posso modificare il proprio stato in seguito ad una richiesta di un componente e condividere tale cambiamento con il resto dei componenti.

Feature Service

Il feature service è un pattern già integrato con Angular utile per estrarre dai componenti la logica relativa ad una feature specifica, come ad esempio l'autenticazione di un utente. In questo modo, i servizi si specializzano e mediante il pattern singleton e dependency injection possono essere usati dai vari componenti.

Decorator

Il decorator è un pattern già integrato con Angular utile per estendere la funzionalità degli oggetti senza dover modificare il codice originale. Questo pattern è usato in Angular per indicare se una classe è di tipo component o module (class decorator) oppure se una variabile è visibile all'esterno della classe in input o output (property decorator) senza dover scrivere codice aggiuntivo.

Lazy Loading

Il lazy loading è un pattern che permette di ritardare il caricamento di determinati moduli solo in caso di utilizzo. Questo aiuta a mantenere bassa la quantità di dati scaricata dall'utente e a rendere i tempi di caricamento più veloci.

Observer

L'observer è un pattern che permette di rimanere in 'ascolto' di determinati eventi. Questo pattern è stato utilizzato con le chiamate HTTP che, in quanto tali, potevano richiedere un tempo variabile per recuperare i dati.

Builder

Il builder è un pattern che è stato utilizzato per aumentare la leggibilità e facilitare la creazione di oggetti contenenti molti campi. Questo pattern è utile poiché separa la costruzione di un oggetto complesso dalla sua rappresentazione, cosicché il processo di costruzione stesso possa creare diverse rappresentazioni.

4.4 Codifica

4.4.1 Maschere

Registrazione

Questa pagina permette ad un utente di registrarsi nell'applicazione web.

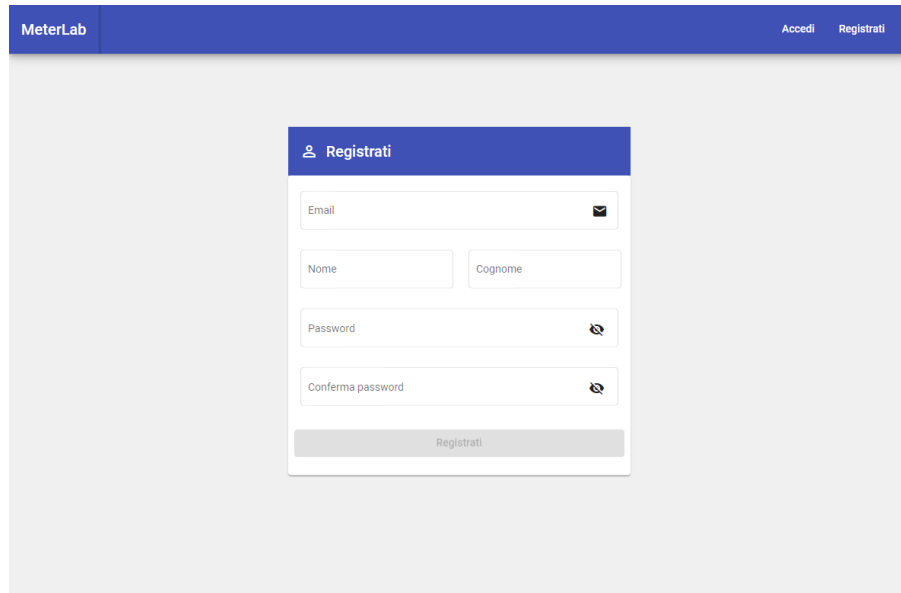


Figura 4.3: Pagina di registrazione

Componenti

- [NavComponent](#);
- [SignupComponent](#).

Servizi Usati

- [AuthService](#);
- [UserService](#);
- [HttpErrorHandlerService](#).

Login

Questa pagina permette ad un utente già registrato di autenticarsi e effettuare l'accesso all'applicazione web.

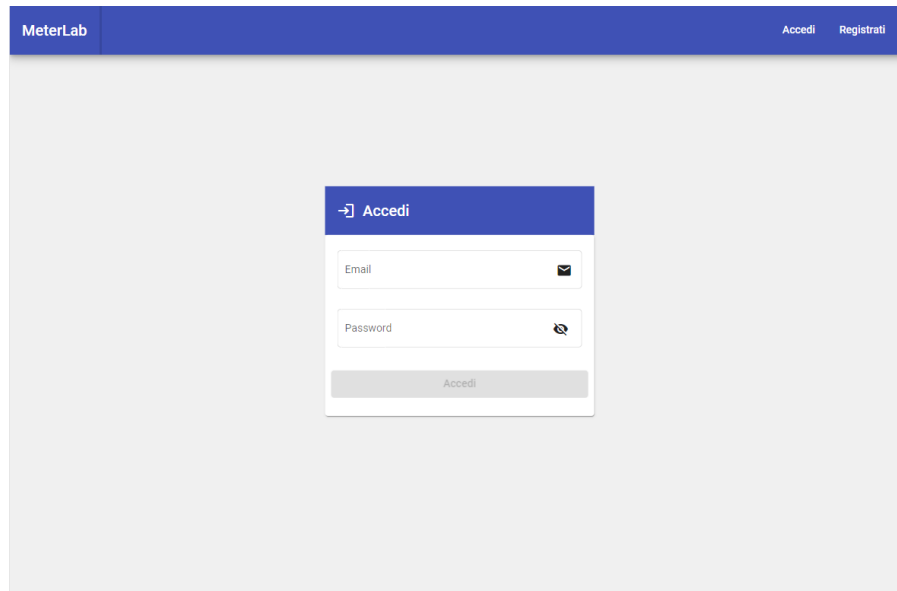


Figura 4.4: Pagina di login

Componenti

- [NavComponent](#);
- [LoginComponent](#).

Servizi Usati

- [AuthService](#);
- [UserService](#);
- [HttpErrorHandlerService](#).

Veicoli

Questa pagina permette ad un utente autenticato di visualizzare tutti i veicoli e le relative informazioni.

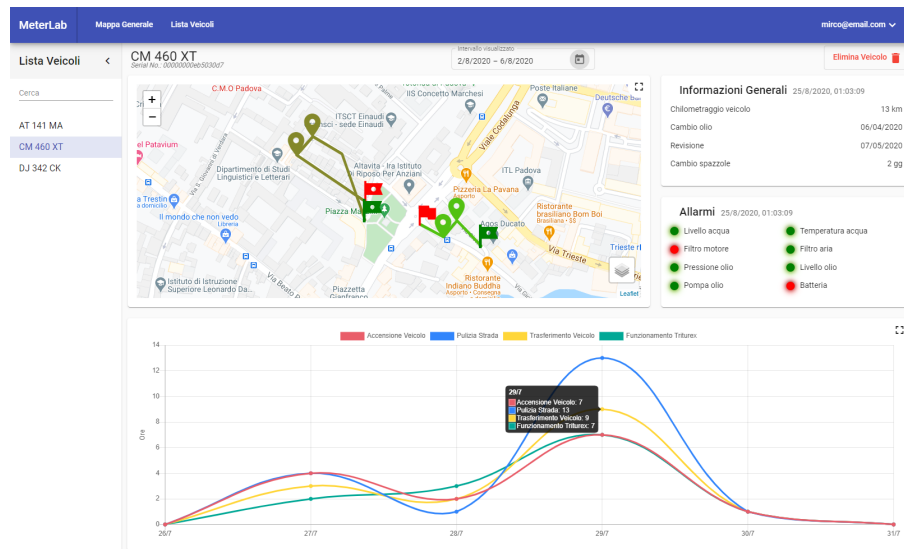


Figura 4.6: Pagina dei veicoli

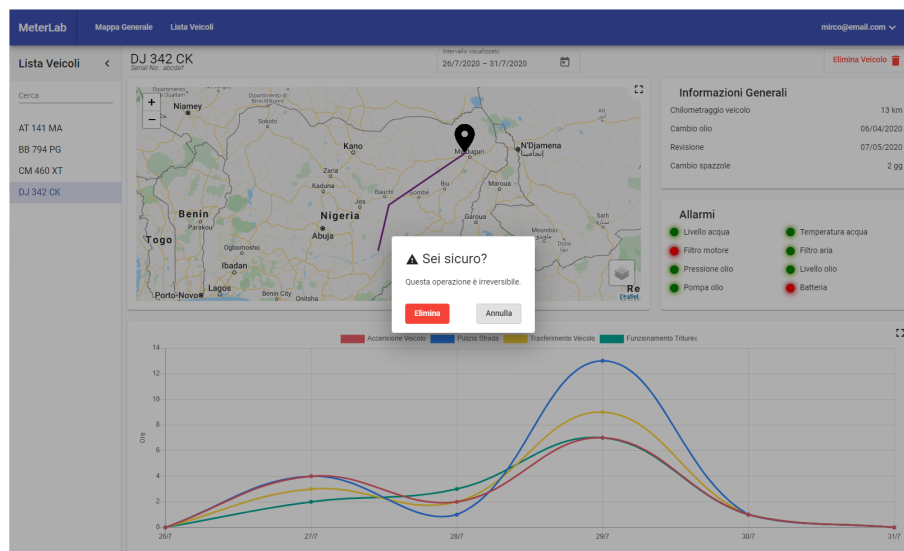


Figura 4.7: Pagina dei veicoli - Dialogo di conferma eliminazione veicolo

Componenti

- [NavComponent](#);
- [VehicleComponent](#);

Servizi Usati

- [UserService](#);
- [VehicleService](#);
- [HttpErrorHandlerService](#).

Impostazioni

Questa pagina permette ad un utente autenticato di modificare le proprie informazioni e la password.

The screenshot shows the 'Impostazioni Account' (Account Settings) page. The header is dark blue with the text 'MeterLab', 'Mappa Generale', 'Lista Veicoli', and a user profile 'mirco@email.com'. The main content area is light gray. In the center, there is a white box titled 'Impostazioni Account' with a gear icon. Inside this box, there are two tabs: 'Informazioni Generali' (selected) and 'Cambio Password'. The 'Informazioni Generali' tab contains three input fields: 'Nome', 'Cognome', and 'Email' (with an email icon). A 'Salva' button is at the bottom of the form.

Figura 4.8: Pagina di impostazioni - Informazioni generali

The screenshot shows the 'Impostazioni Account' (Account Settings) page in the MeterLab application. The page has a dark blue header with the 'MeterLab' logo, navigation links 'Mappa Generale' and 'Lista Veicoli', and a user profile 'mirco@email.com' with a dropdown arrow. The main content area is light gray and features a gear icon and the title 'Impostazioni Account'. Below this is a white modal form with two tabs: 'Informazioni Generali' and 'Cambio Password'. The 'Cambio Password' tab is active. It contains three password input fields: 'Password attuale' (current password), 'Nuova password' (new password), and 'Conferma password' (confirm password). Each field has a toggle icon to the right. At the bottom of the form is a 'Cambia Password' button.

Figura 4.9: Pagina di impostazioni - Cambio password

Componenti

- [NavComponent](#);
- [UserSettingsComponent](#).

Servizi Usati

- [UserService](#);
- [HttpErrorHandlerService](#).

4.4.2 Componenti

NavComponent

Questo componente permette all'utente di spostarsi tra le diverse pagine dell'applicazione. La barra cambia a seconda se l'utente è autenticato o meno.

Se l'utente **non** è autenticato si può trovare:

- Accedi;
- Registrati.

Mentre se l'utente è autenticato si può trovare:

- Mappa generale;
- Lista veicoli;
- Email dell'account;
- Impostazioni;
- Esci.



Figura 4.10: Componente NavComponent - utente non autenticato

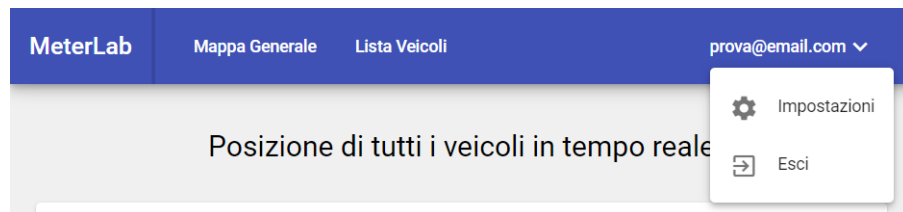


Figura 4.11: Componente NavComponent - utente autenticato

Servizi usati

- [AuthService](#).

Metodi

- **ngOnInit**: all'inizializzazione del componente viene controllato, mediante il servizio AuthService, se l'utente è autenticato o meno così da poter visualizzare la navigazione corretta;
- **logout**: effettua, mediante il servizio AuthService, il logout dell'utente.

SignupComponent

Questo componente è composto da un form che richiede i dati necessari alla registrazione. Questi dati sono:

- Email;
- Nome;
- Cognome;
- Password;
- Conferma password.

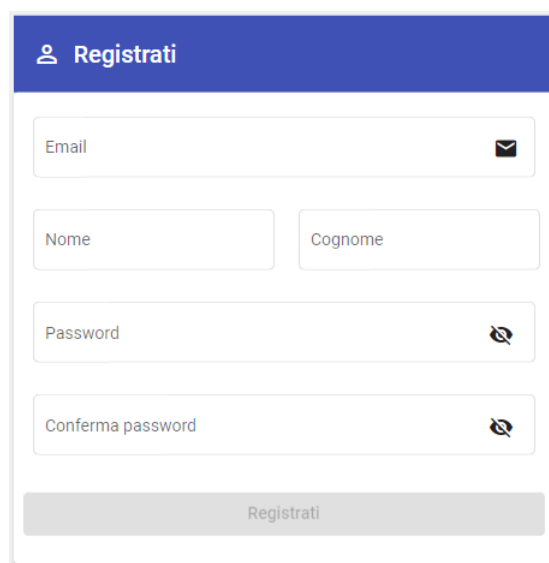
The image shows a registration form titled "Registrati" with a user icon. It contains five input fields: "Email" with an envelope icon, "Nome" and "Cognome" as separate fields, "Password" with an eye icon, and "Conferma password" with an eye icon. A "Registrati" button is at the bottom.

Figura 4.12: Componente SignupComponent

Servizi usati

- [AuthService](#);
- [HttpErrorHandlerService](#).

Metodi

- **onSubmit:** prende i dati dei campi del form e chiama il metodo signUp del servizio AuthService per registrare l'utente. Oltre a gestire gli errori utili al componente derivati da tale chiamata (nel caso di utente già presente) delega il resto degli errori al servizio HttpErrorHandlerService;

LoginComponent

Questo componente è composto da un form che richiede i dati necessari all'autenticazione di un utente. Necessita dei seguenti dati:

- Email;
- Password;

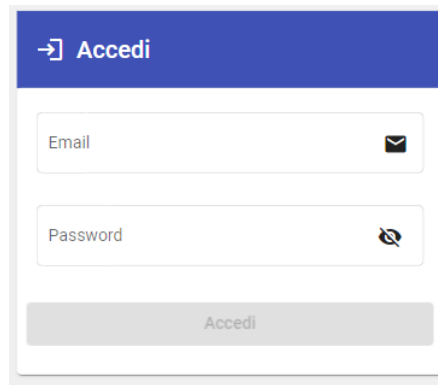
The image shows a login form with a blue header bar containing a right-pointing arrow icon and the text 'Accedi'. Below the header are two input fields: 'Email' with an envelope icon on the right, and 'Password' with an eye icon on the right. At the bottom of the form is a wide, light gray button labeled 'Accedi'.

Figura 4.13: Componente LoginComponent

Servizi usati

- [AuthService](#);
- [HttpErrorHandlerService](#).

Metodi

- **onSubmit:** prende i dati dei campi del form e chiama il metodo login del servizio AuthService per autenticare l'utente. Oltre a gestire gli errori utili al componente derivati da tale chiamata (nel caso di credenziali errate) delega il resto degli errori al servizio HttpErrorHandlerService;

VehiclePositionsComponent

Questo componente è stato sviluppato da un mio collega.

In breve, questo componente visualizza una mappa con dei marker che rappresentano le ultime posizioni di ciascun veicolo monitorato.

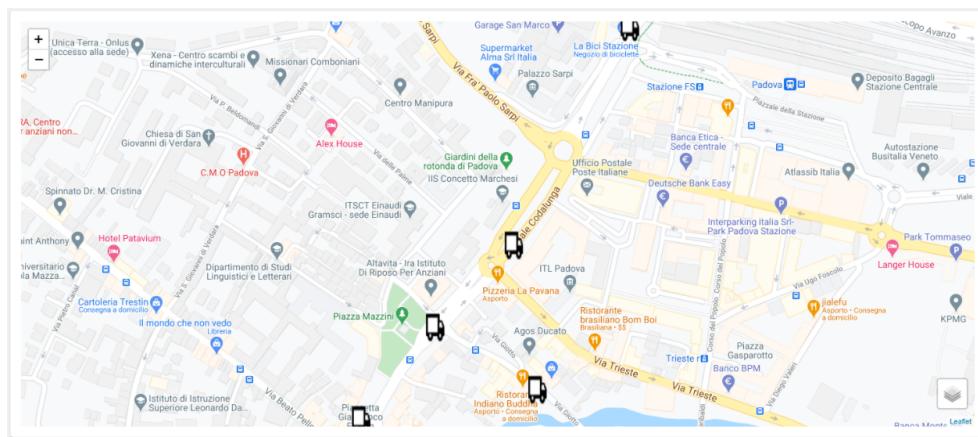


Figura 4.14: Componente VehiclePositionsComponent

VehicleComponent

Questo componente fa da contenitore ad una serie di altri componenti. Le funzionalità fornite direttamente da questo componente sono:

- Visualizzare la targa del veicolo selezionato;
- Visualizzare il numero seriale del veicolo selezionato;
- Selezionare un intervallo di tempo in base alla quale visualizzare i dati dei componenti figli;
- Eliminare il veicolo selezionato.

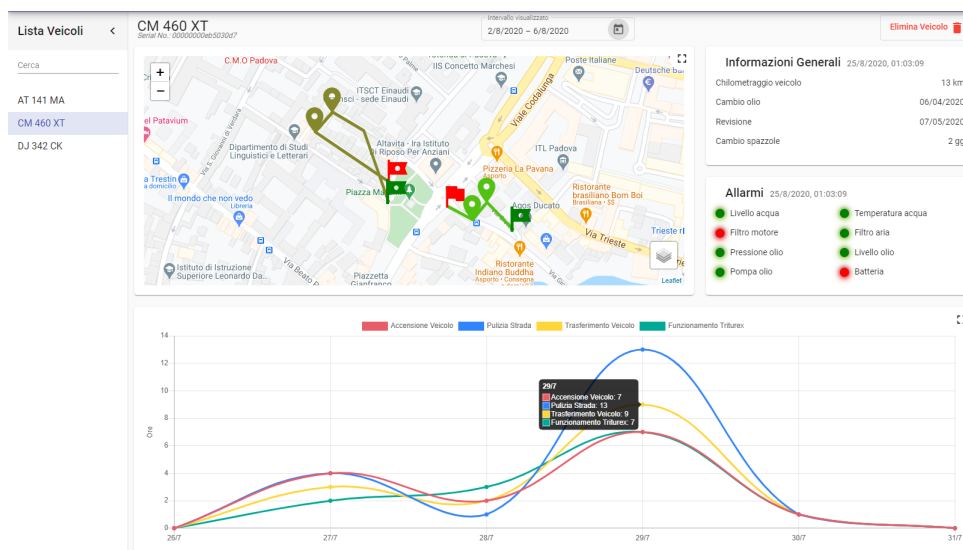


Figura 4.15: Componente VehicleComponent

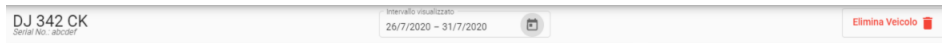


Figura 4.16: Funzionalità gestite direttamente da VehicleComponent

Componenti

- [SideNavComponent](#);
- [VehicleMapComponent](#);
- [VehicleInformationsComponent](#);
- [VehicleAlarmsComponent](#);
- [VehicleGraphComponent](#);
- [DeleteDialogComponent](#).

Servizi usati

- [AuthService](#);
- [VehicleService](#);
- [HttpErrorHandlerService](#).

Metodi

- **set selectedVehicle:** quando viene cambiato il veicolo si occupa, utilizzando il servizio `VehicleService`, di ottenere le informazioni per inizializzare alcuni dei componenti figli e delega la gestione degli errori al servizio `HttpErrorHandlerService`;
- **onSelectVehicle:** cambia il veicolo selezionato;
- **deleteVehicleConfirmation:** chiede conferma per eliminare il veicolo selezionato;
- **deleteVehicle:** elimina il veicolo selezionato e delega la gestione degli errori al servizio `HttpErrorHandlerService`;
- **addEndDate:** memorizza l'intervallo di tempo selezionato dall'utente.

SideNavController

Questo componente permette di visualizzare una lista di tutti i veicoli monitorati e di cliccare su uno di essi per vederne le informazioni. Permette inoltre di fare una ricerca sulla lista.

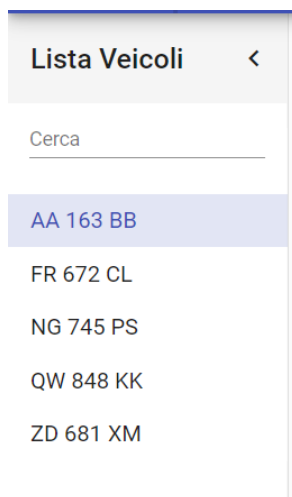


Figura 4.17: Componente SideNavController

Servizi usati

- [VehicleService](#);
- [HttpErrorHandlerService](#).

Output

- **selectedVehicle**: emette un evento ogni volta che viene selezionato un nuovo veicolo.

Metodi

- **filterVehicle**: quando l'utente inserisce qualche lettera, numero o parola nell'apposita barra di ricerca del veicolo, viene chiamato questo metodo che mostrerà solo i veicoli che nel nome della targa contengono i dati inseriti dall'utente. Funge quindi da filtro per trovare più velocemente i veicoli;
- **ngOnInit**: alla generazione del componente chiede la lista di tutti i veicoli;
- **getVehicleList**: ritorna la lista di tutti i veicoli utilizzando il servizio VehicleService e delega la gestione degli errori al servizio HttpErrorHandlerService;
- **toggleSidenav**: cambia il valore booleano dello stato della sidenav. Da true a false e viceversa così da aprire e chiudere la barra laterale;
- **selectVehicle**: emette un evento per informare che un veicolo è stato selezionato.

VehicleMapComponent

Questo componente è stato sviluppato da un mio collega.

In breve, questo componente permette di visualizzare uno o più tracciati in base all'intervallo dei giorni nel calendario selezionati dall'utente. Permette inoltre di visualizzare, con dei marker, i punti in cui si è rilevato un cambio di stato, come ad esempio l'accensione di un allarme del motore.

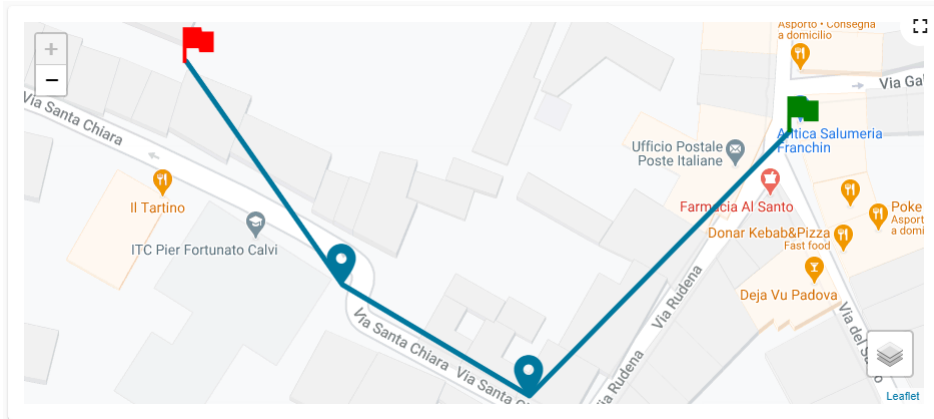


Figura 4.18: Componente VehicleMapComponent

VehicleInformationsComponent

Questo componente visualizza le informazioni generali di un veicolo. Quando viene cliccato un marker della mappa questo componente visualizza le informazioni che il veicolo aveva in quel momento. Tali informazioni sono:

- Chilometraggio veicolo;
- Cambio olio;
- Revisione;
- Cambio spazzole.

Informazioni Generali 21/8/2020, 11:18:20	
Chilometraggio veicolo	15 km
Cambio olio	06/04/2020
Revisione	07/05/2020
Cambio spazzole	2 gg

Figura 4.19: Componente vehicleInformationsComponent

Input

- **vehicleInformations**: oggetto contenente i dati da visualizzare nel componente.

VehicleAlarmsComponent

Questo componente visualizza gli allarmi di un veicolo. Quando viene cliccato un marker della mappa questo componente visualizza gli allarmi che il veicolo aveva in quel momento. Gli allarmi visualizzati sono:

- Livello acqua;
- Temperatura acqua;
- Filtro motore;
- Filtro aria;
- Pressione olio;
- Livello olio;
- Pompa olio;
- Batteria.

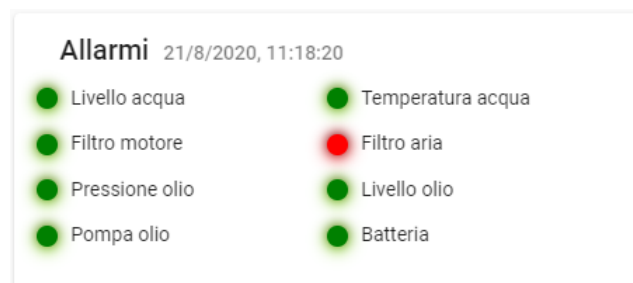


Figura 4.20: Componente vehicleAlarmsComponent

Input

- **vehicleAlarms**: oggetto contenente i dati da visualizzare nel componente.

Metodi

- **generateLedTitle**: funzione di accessibilità per generare un title per il led in base al suo stato.

VehicleGraphComponent

Componente che visualizza un grafico delle ore utilizzate (asse y) in un certo arco di tempo selezionato dall'utente (asse x) su diverse categorie. Tali categorie sono:

- Ore accensione veicolo;
- Ore pulizia strada;
- Ore trasferimento veicolo;
- Ore funzionamento trituratore.

Viene visualizzato un grafico a barre se è selezionato un solo giorno altrimenti viene visualizzato un grafico a linee.

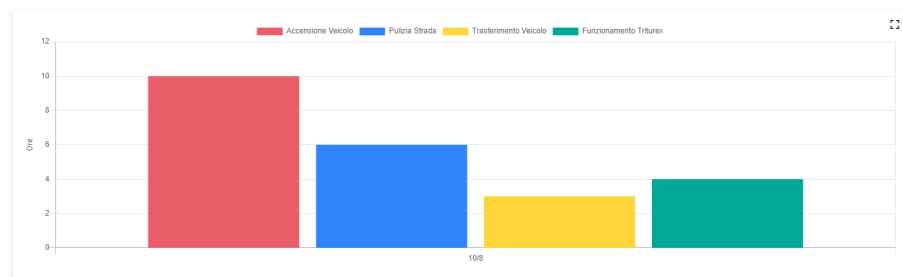


Figura 4.21: Componente VehicleGraphComponent - Visualizzazione di un solo giorno

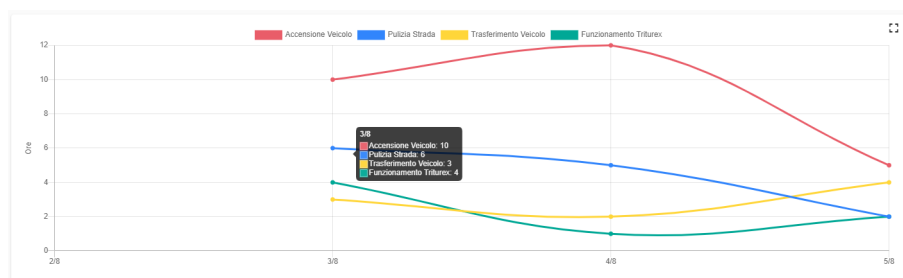


Figura 4.22: Componente VehicleGraphComponent - Visualizzazione di un intervallo di giorni

Servizi usati

- [VehicleService](#);
- [HttpErrorHandlerService](#).

Input

- **isGraphFullScreen**: booleano per settare lo stato del componente (a tutto schermo o normale);
- **selectedVehicle**: oggetto contenente il veicolo selezionato dall'utente e di cui chiedere le informazioni da visualizzare;

- **rangeSelected:** oggetto contenente il range di date selezionato dall'utente da essere visualizzate.

Output

- **isGraphFullScreen:** emette un evento booleano che indica se lo stato del componente è cambiato.

Metodi

- **toggleGraphDimension:** cambio il valore booleano che indica se il componente è a tutto schermo oppure no;
- **createContent:** controlla che sia selezionato un veicolo e un range di date e decide che tipo di grafico visualizzare;
- **populateGraph:** popola il grafico con tutte le informazioni utilizzando il servizio VehicleService e delegando la gestione degli errori al servizio ErrorHandlerService;
- **generateLabels:** crea le etichette da mettere nell'asse delle x del grafico. Una etichetta per ogni giorno compreso nel range selezionato;
- **insertDataSerie:** inserisce un array di dati con una etichetta nel grafico;

DeleteDialogComponent

Componente che chiede all'utente se è sicuro di voler eliminare un determinato veicolo.

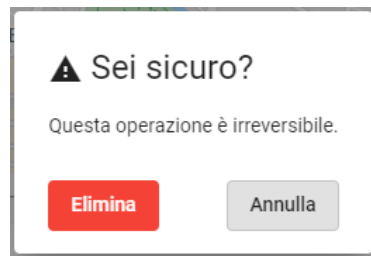


Figura 4.23: Componente DeleteDialogComponent

Metodi

- **onCancel:** chiude la finestra di dialogo inviando come evento quello di annulla azione;
- **onDelete:** chiude la finestra di dialogo inviando come evento quello di conferma eliminazione.

UserSettingsComponent

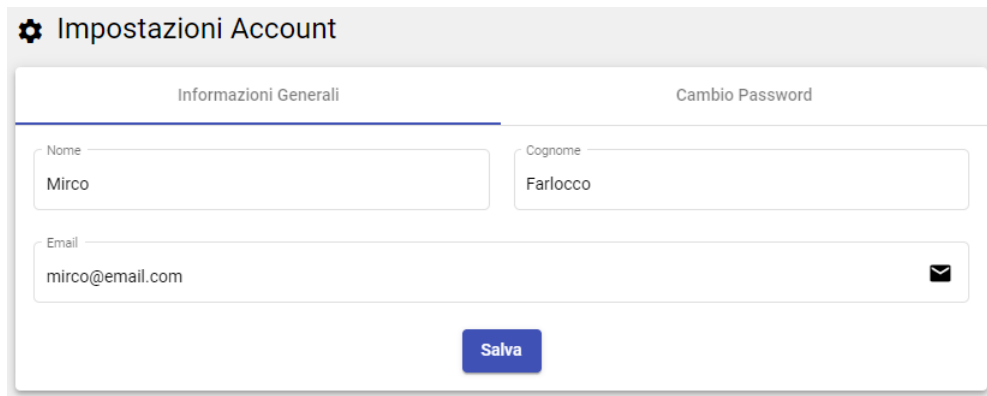
Questo componente è composto da due form. Il primo mostra le informazioni dell'utente e da la possibilità di modificarle mentre il secondo da la possibilità di cambiare la password dell'account.

Il form per cambiare le informazioni richiede:

- Nome;
- Cognome;
- Email.

Il form per cambiare la password richiede:

- Password attuale;
- Nuova password;
- Conferma password.



The screenshot shows a web interface titled "Impostazioni Account" with a gear icon. It features two tabs: "Informazioni Generali" (selected) and "Cambio Password". Under the "Informazioni Generali" tab, there are three input fields: "Nome" containing "Mirco", "Cognome" containing "Farlocco", and "Email" containing "mirco@email.com" with an email icon on the right. A blue "Salva" button is positioned at the bottom center of the form.

Figura 4.24: Componente UserSettingsComponent - Informazioni generali

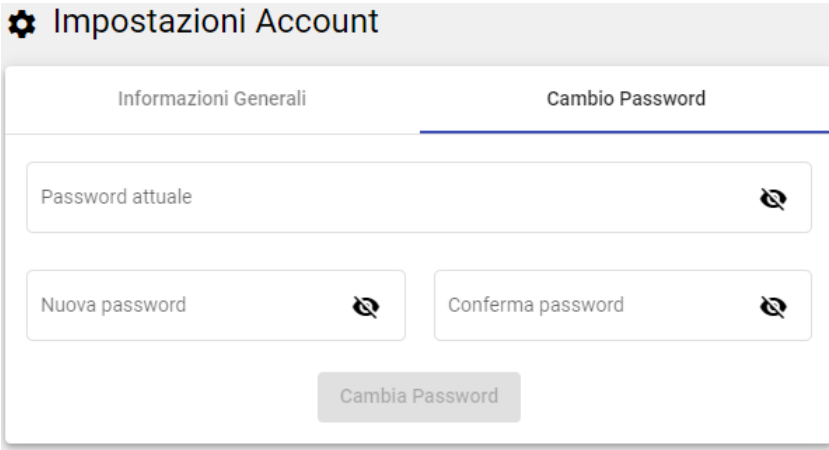


Figura 4.25: Componente UserSettingsComponent - Cambio password

Servizi usati

- [UserService](#);
- [HttpErrorHandlerService](#).

Metodi

- **ngOnInit**: alla generazione del componente recupera le informazioni dell'utente;
- **onSubmitInformations**: prende i dati dei campi del form per cambiare le informazioni e chiama il metodo `updateUserInformations` del servizio `UserService`. Oltre a gestire gli errori utili al componente derivati da tale chiamata (nel caso di email già usata) delega il resto degli errori al servizio `HttpErrorHandlerService`;
- **onSubmitPassword**: prende i dati dei campi del form per cambiare la password e chiama il metodo `updateUserPassword` del servizio `UserService`. Oltre a gestire gli errori utili al componente derivati da tale chiamata (nel caso di password attuale errata) delega il resto degli errori al servizio `HttpErrorHandlerService`;

4.4.3 Servizi

AuthService

Questo servizio si occupa della parte di autenticazione dell'utente facendo chiamate HTTP al [back end](#) e gestendo il token JWT. Permette quindi di autenticare un utente e registrarne uno nuovo.

Output

- **userLoginState**: emette un evento ogni volta che lo stato dell'utente cambia (da autenticato a non e viceversa) con valore `true` se l'utente è autenticato.

Metodi

- **set isUserLogged**: metodo setter che si occupa, oltre che a settare il valore della variabile interna *isUserLogged*, di emettere un evento sulla variabile di output *userLoginState*;
- **isUserLoggedIn**: restituisce un valore booleano che indica se l'utente è autenticato o meno;
- **login**: effettua l'autenticazione di un utente codificando la password in sha512, facendo una chiamata POST per verificare le credenziali e ricevendo il token di autenticazione;
- **signUp**: effettua la registrazione di un nuovo utente codificando la password in sha512, facendo una chiamata POST per registrare l'utente e ricevendo il token di autenticazione;
- **logout**: effettua il logout di un utente autenticato reindirizzando alla pagina di login;
- **getToken**: ritorna il token JWT salvato a seguito dell'autenticazione;
- **getResponseToken**: ritorna il token JWT contenuto nell'header HTTP di una risposta di autenticazione.

VehicleService

Questo servizio si occupa di ottenere, facendo chiamate HTTP al [back end](#), le informazioni riguardanti i veicoli.

Output

- **vehicleDeleted**: emette un evento ogni volta che un veicolo viene eliminato così che gli ascoltatori possano chiedere le informazioni aggiornate.

Metodi

- **getAuthHeader**: funzione ausiliaria per costruire l'header con il token JWT da agganciare alle richieste HTTP;
- **getVehicles**: ritorna la lista dei veicoli monitorati facendo una chiamata HTTP GET;
- **getVehicleActualData**: ritorna le informazioni "statiche" di un veicolo quali allarmi e informazioni generali facendo una chiamata HTTP GET;
- **getVehicleWorkHours**: ritorna le ore lavorate da un veicolo su ogni categoria per ogni giorno richiesto facendo una chiamata HTTP GET;
- **deleteVehicle**: elimina un veicolo facendo una chiamata HTTP DELETE.

UserService

Questo servizio si occupa di ottenere, facendo chiamate HTTP al [back end](#), le informazioni riguardanti gli utenti e di modificarle.

Output

- **emailChanged**: emette un evento ogni volta che l'email dell'utente cambia così da poter aggiornare i componenti che la visualizzano.

Metodi

- **getAuthHeader**: funzione ausiliaria per costruire l'header con il token JWT da agganciare alle richieste HTTP;
- **getUserInformations**: ritorna le informazioni di un utente facendo una chiamata HTTP GET;
- **updateUserInformations**: modifica le informazioni dell'utente facendo una chiamata HTTP PUT;
- **updateUserPassword**: modifica la password dell'utente facendo una chiamata HTTP POST;

HttpErrorHandlerService

Questo servizio si occupa di gestire gli errori HTTP in modo da standardizzarne il comportamento.

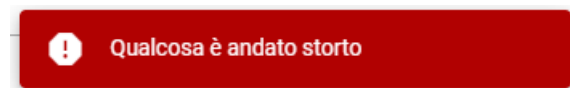


Figura 4.26: Messaggio di errore generico

Metodi

- **handle**: gestisce gli errori HTTP
 - **401**: fa il logout dell'utente, reindirizzandolo alla pagina di login per farlo autenticare di nuovo;
 - **500**: mostra a schermo un messaggio di errore avvisando che si tratta un errore lato server;
 - **default**: mostra a schermo un messaggio di errore generico per qualsiasi altro errore.
- **displayErrorMessage**: mostra a schermo, usando uno `snackBar`, un messaggio di errore con testo customizzato.

RouteGuard

Questo servizio si occupa di proteggere l'accesso a determinate pagine che necessitano che l'utente sia loggato per accedervi.

Metodi

- **canActivate:** ritorna un booleano che indica se l'utente è loggato o meno reindirizzando alla pagina di login in caso negativo.

Capitolo 5

Verifica e validazione

5.1 Accessibilità

5.1.1 Attributi HTML

Per rendere il sito più accessibile sono stati usati diversi attributi HTML.

- **title**: utilizzato per descrivere elementi che altrimenti non sarebbero leggibili dagli [screen reader](#), totalmente o parzialmente. Uno degli usi è stato per specificare lo stato dei led degli allarmi;
- **scope**: utilizzato per facilitare la lettura delle tabelle. Usato su una tabella nascosta visivamente agli utenti ma visibile agli [screen reader](#) per rendere i dati rappresentati dal grafico accessibili;
- **lang**: utilizzato per indicare allo [screen reader](#) in che lingua leggere una parola o frase diversa dall'italiano.

5.1.2 Colori

I colori del sito sono stati selezionati per passare il test WCAG AA che richiede un rapporto di contrasti di almeno 4.5:1 per il testo normale e 3:1 per il testo in grassetto. Di seguito è riportata la tabella dei colori del testo e del loro sfondo e il loro rapporto di contrasto.

Tabella 5.1: Rapporto contrasti testo-sfondo

Colore testo	Colore sfondo	Rapporto
#FFFFFF	#3F51B5	6.87:1
#000000	#F5F5F5	19.26:1
#F44336	#F0F0F0	3.23:1
#000000	#FFFFFF	3.23:1
#FFFFFF	#3F51B5	6.87:1

5.1.3 Altri elementi di accessibilità

- È stato aggiunto un suggerimento per gli [screen reader](#) fornendo un link prima del menu di navigazione così da poterlo saltare e andare direttamente al contenuto della pagina;
- È stato aggiunto un suggerimento per gli [screen reader](#) fornendo un link prima del menu laterale, contenente la lista dei veicoli, così da poterlo saltare e andare direttamente alle informazioni del veicolo;
- Sono stati rimossi i link circolari per evitare confusione negli utenti e permettere di identificare univocamente il percorso raggiunto;
- Si sono evitati testi scorrevoli, lampeggianti, barrati ed in generale font troppo elaborati per agevolare la lettura;
- Come già citato, è stata resa visibile solo agli [screen reader](#) una tabella contenente i dati rappresentati dal grafico;
- Sono stati utilizzati gli attributi [Web Accessibility Initiative - Accessible Rich Internet Applications \(WAI-ARIA\)](#) per descrivere meglio alcuni elementi:
 - **role**: utilizzato per definire la funzione di alcuni tag;
 - **aria-hidden**: utilizzato per nascondere icone inutili agli [screen reader](#) perchè già presenti descrizioni testuali;
 - **aria-label**: utilizzato per definire una descrizione testuale per bottoni rappresentati solo da icone;
 - **aria-pressed**: utilizzato per indicare un bottone con due stati (nascondi/-visualizza, apri/chiudi) in quale stato si trova;
 - **aria-labelledby**: utilizzato per indicare una relazione tra un elemento e la sua descrizione.

5.2 Verifica

In questa sezione vengono elencati i test scritti per il [front end](#) del progetto. Il codice dei test è così strutturato TU[numero], dove:

- **TU**: Test di unità;
- **[numero]**: indica un numero progressivo per identificare univocamente il test.

Tabella 5.2: Test di unità

ID	Elemento	Descrizione	Stato
TU1	NavComponent	Il componente deve essere creato	✓
TU2	NavComponent	Il metodo logout deve far uscire l'utente dal suo account	✓

Continuazione della tabella 5.2			
ID	Elemento	Descrizione	Stato
TU3	SignupComponent	Il componente deve essere creato	✓
TU4	SignupComponent	Il metodo onSubmit deve registrare l'utente	✓
TU5	SignupComponent	Il metodo onSubmit deve gestire l'errore HTTP 409 in caso di email già esistente	✓
TU6	SignupComponent	Il metodo onSubmit deve gestire qualsiasi altro errore HTTP	✓
TU7	LoginComponent	Il metodo onSubmit deve autenticare l'utente	✓
TU8	LoginComponent	Il metodo onSubmit deve gestire l'errore HTTP 400 in caso di credenziali errate	✓
TU9	LoginComponent	Il metodo onSubmit deve gestire qualsiasi altro errore HTTP	✓
TU10	VehicleComponent	Il componente deve essere creato	✓
TU11	VehicleComponent	Il metodo setSelectedVehicle deve ottenere i dati generali di un veicolo	✓
TU12	VehicleComponent	Il metodo setSelectedVehicle non deve ottenere i dati generali se il veicolo non è valido	✓
TU13	VehicleComponent	Il metodo setSelectedVehicle non deve ottenere i dati generali se il veicolo è quello già visualizzato	✓
TU14	VehicleComponent	Il metodo setSelectedVehicle deve gestire gli errori HTTP	✓
TU15	VehicleComponent	Il metodo onSelectVehicle deve memorizzare il veicolo selezionato dall'utente	✓
TU16	VehicleComponent	Il metodo deleteVehicleConfirmation deve eliminare il veicolo se riceve la conferma	✓
TU17	VehicleComponent	Il metodo deleteVehicleConfirmation non deve eliminare il veicolo se non riceve la conferma	✓
TU18	VehicleComponent	Il metodo deleteVehicle deve eliminare il veicolo	✓
TU19	VehicleComponent	Il metodo deleteVehicle deve gestire gli errori HTTP	✓

Continuazione della tabella 5.2			
ID	Elemento	Descrizione	Stato
TU20	VehicleComponent	Il metodo addEndDate deve memorizzare l'intervallo di tempo selezionato	✓
TU21	SidenavComponent	Il componente deve essere creato	✓
TU22	SidenavComponent	Il componente deve aggiornare la sua lista dei veicoli quando un veicolo viene eliminato	✓
TU23	SidenavComponent	Il metodo ngOnInit deve ottenere la lista dei veicoli	✓
TU24	SidenavComponent	Il metodo getVehicleList deve ottenere la lista dei veicoli	✓
TU25	SidenavComponent	Il metodo getVehicleList deve gestire gli errori HTTP	✓
TU26	SidenavComponent	Il metodo toggleSidenav deve invertire lo stato del menu (aperto/chiuso)	✓
TU27	SidenavComponent	Il metodo set searchTerm deve filtrare la lista dei veicoli ogni volta che l'input cambia	✓
TU28	SidenavComponent	Il metodo filterVehicles deve filtrare i veicoli	✓
TU29	SidenavComponent	Il metodo selectVehicle deve emettere un evento se è stato selezionato un veicolo valido	✓
TU30	SidenavComponent	Il metodo selectVehicle non deve emettere un evento se non è stato selezionato un veicolo valido	✓
TU31	VehicleInformations	Il componente deve essere creato	✓
TU32	VehicleAlarms	Il componente deve essere creato	✓
TU33	VehicleAlarms	Il metodo generateLedTitle deve ritornare una descrizione appropriata se l'allarme è attivo	✓
TU34	VehicleAlarms	Il metodo generateLedTitle deve ritornare una descrizione appropriata se l'allarme non è attivo	✓
TU35	VehicleGraph	Il componente deve essere creato	✓
TU36	VehicleGraph	Il metodo set selectedVehicle deve modificare il grafico quando il veicolo cambia	✓

Continuazione della tabella 5.2			
ID	Elemento	Descrizione	Stato
TU37	VehicleGraph	Il metodo set rangeSelected deve modificare il grafico quando l'intervallo cambia	✓
TU38	VehicleGraph	Il metodo toggleGraphDimension deve invertire lo stato del grafico (normale/ingrandito)	✓
TU39	VehicleGraph	Il metodo createContent deve creare il grafico se il veicolo è valido	✓
TU40	VehicleGraph	Il metodo createContent non deve creare il grafico se il veicolo non è valido	✓
TU41	VehicleGraph	Il metodo createContent deve usare un grafico a barre se è selezionato un solo giorno	✓
TU42	VehicleGraph	Il metodo createContent deve usare un grafico a linee se è selezionato più di un giorno	✓
TU43	VehicleGraph	Il metodo populateGraph deve popolare il grafico	✓
TU44	VehicleGraph	Il metodo populateGraph deve gestire gli errori HTTP	✓
TU45	VehicleGraph	Il metodo generateLabels deve creare l'array delle etichette dell'asse x	✓
TU46	VehicleGraph	Il metodo insertDataSerie deve inserire nel grafico una nuova serie di dati	✓
TU47	DeleteDialog	Il componente deve essere creato	✓
TU48	DeleteDialog	Il metodo onCancel deve chiudere la finestra di dialogo con evento 'cancel'	✓
TU49	DeleteDialog	Il metodo onDelete deve chiudere la finestra di dialogo con evento 'delete'	✓
TU50	UserSettings	Il componente deve essere creato	✓
TU51	UserSettings	Il metodo ngOnInit deve ottenere le informazioni dell'utente	✓
TU52	UserSettings	Il metodo ngOnInit deve gestire gli errori HTTP	✓
TU53	UserSettings	Il metodo onSubmitInformations deve modificare i dati utente	✓

Continuazione della tabella 5.2			
ID	Elemento	Descrizione	Stato
TU54	UserSettings	Il metodo onSubmitInformations deve gestire l'errore HTTP 400 in caso di email già esistente	✓
TU55	UserSettings	Il metodo onSubmitInformations deve gestire qualsiasi altro errore HTTP	✓
TU56	UserSettings	Il metodo onSubmitPassword deve modificare la password dell'utente	✓
TU57	UserSettings	Il metodo onSubmitPassword deve gestire l'errore HTTP 400 in caso di password attuale errata	✓
TU58	UserSettings	Il metodo onSubmitPassword deve gestire qualsiasi altro errore HTTP	✓
TU59	AuthService	Il servizio deve essere creato	✓
TU60	AuthService	Il metodo set isUserLogged deve emettere un evento di stato cambiato	✓
TU61	AuthService	Il metodo isUserLoggedIn deve ritornare false se non trova nessun token	✓
TU62	AuthService	Il metodo isUserLoggedIn deve ritornare true se trova un token	✓
TU63	AuthService	Il metodo login deve autenticare l'utente	✓
TU64	AuthService	Il metodo signUp deve registrare l'utente	✓
TU65	AuthService	Il metodo logout deve eliminare il token salvato	✓
TU66	AuthService	Il metodo getToken deve ritornare il token memorizzato	✓
TU67	AuthService	Il metodo getResponseToken deve ritornare il token salvato nell'header di una risposta HTTP di autenticazione	✓
TU68	VehicleService	Il servizio deve essere creato	✓
TU69	VehicleService	Il metodo getVehicles deve ritornare la lista dei veicoli	✓
TU70	VehicleService	Il metodo getVehiclePositions deve ritornare la posizione dei veicoli	✓

Continuazione della tabella 5.2			
ID	Elemento	Descrizione	Stato
TU71	VehicleService	Il metodo <code>getVehiclePositionById</code> deve ritornare le posizioni di un veicolo	✓
TU72	VehicleService	Il metodo <code>getVehicleWorkHours</code> deve ritornare la lista delle ore lavorate in ogni categoria ogni giorno	✓
TU73	VehicleService	Il metodo <code>getVehiclePositionAndData</code> deve ritornare le posizioni di un veicolo e i dati legati ad ogni posizione	✓
TU74	VehicleService	Il metodo <code>deleteVehicle</code> deve eliminare il veicolo	✓
TU75	UserService	Il servizio deve essere creato	✓
TU76	UserService	Il metodo <code>getUserInformations</code> deve ritornare le informazioni dell'utente ora autenticato	✓
TU77	UserService	Il metodo <code>updateUserInformations</code> deve modificare le informazioni dell'utente ora autenticato	✓
TU78	UserService	Il metodo <code>updateUserPassword</code> deve modificare la password dell'utente ora autenticato	✓
TU79	HttpErrorHandler	Il servizio deve essere creato	✓
TU80	HttpErrorHandler	Il metodo <code>handle</code> deve gestire l'errore HTTP 500 (errore interno al server)	✓
TU81	HttpErrorHandler	Il metodo <code>handle</code> deve gestire l'errore HTTP 401 (errore di autenticazione)	✓
TU82	HttpErrorHandler	Il metodo <code>handle</code> deve gestire di default tutti gli altri errori HTTP	✓
TU83	HttpErrorHandler	Il metodo <code>displayErrorMessage</code> deve aprire una finestra di avviso per mostrare un errore	✓
TU84	RouteGuardService	Il servizio deve essere creato	✓
TU85	RouteGuardService	Il metodo <code>canActivate</code> deve ritornare <code>true</code> se l'utente è autenticato	✓
TU86	RouteGuardService	Il metodo <code>canActivate</code> deve ritornare <code>false</code> se l'utente non è autenticato	✓

File		Statements	Branches	Functions	Lines
src	<div></div>	100%	3/3	100%	0/0
src/app	<div></div>	100%	2/2	100%	0/0
src/app/core/classes	<div></div>	91.15%	103/113	100%	13/13
src/app/core/route-guards	<div></div>	100%	8/8	100%	2/2
src/app/core/services/auth	<div></div>	100%	33/33	100%	0/0
src/app/core/services/http-error-handler	<div></div>	100%	12/12	100%	4/4
src/app/core/services/user	<div></div>	100%	16/16	100%	0/0
src/app/core/services/vehicle	<div></div>	100%	15/15	100%	0/0
src/app/login	<div></div>	100%	16/16	100%	2/2
src/app/shared	<div></div>	100%	9/9	100%	4/4
src/app/shared/material	<div></div>	100%	3/3	100%	0/0
src/app/shared/nav	<div></div>	100%	12/12	100%	0/0
src/app/shared/pipes/formatDate	<div></div>	100%	6/6	100%	2/2
src/app/shared/pipes/formatPlate	<div></div>	100%	5/5	100%	2/2
src/app/shared/pipes/numberWithDots	<div></div>	100%	3/3	100%	0/0
src/app/shared/pipes/toReadableDate	<div></div>	100%	3/3	100%	0/0
src/app/signup	<div></div>	100%	17/17	100%	2/2
src/app/user-settings	<div></div>	100%	35/35	100%	4/4
src/app/vehicles	<div></div>	100%	35/35	100%	10/10
src/app/vehicles/delete-dialog	<div></div>	100%	6/6	100%	0/0
src/app/vehicles/sidenav	<div></div>	96.67%	29/30	100%	6/6
src/app/vehicles/vehicle-alarms	<div></div>	100%	4/4	100%	2/2
src/app/vehicles/vehicle-graph	<div></div>	100%	60/60	100%	6/6
src/app/vehicles/vehicle-informations	<div></div>	100%	3/3	100%	0/0
src/environments	<div></div>	100%	1/1	100%	0/0

Figura 5.1: Copertura del codice

5.3 Validazione e collaudo

Una volta alla settimana è stata organizzata una riunione di allineamento con tutte le persone che stavano lavorando al progetto. In queste riunioni si otteneva un feedback sul lavoro svolto fino a quel momento e una lista dei possibili cambiamenti da fare per raggiungere l'obiettivo voluto dal committente.

L'ultima settimana di stage è stata fatta una presentazione finale con il [front end](#) e il [back end](#) installati su una macchina, dove si è mostrata l'applicazione funzionante e si è collaudato tutte le funzionalità richieste.

Capitolo 6

Conclusioni

6.1 Raggiungimento degli obiettivi

Per quanto riguarda gli obiettivi prefissati a inizio dello stage (sezione 2.2) risultano tutti raggiunti. Di seguito la tabella riassuntiva.

Tabella 6.1: Riepilogo dello stato degli obiettivi

Obiettivo	Stato
O01	Soddisfatto
O02	Soddisfatto
O03	Soddisfatto
D01	Soddisfatto
F01	Soddisfatto

6.2 Analisi del lavoro svolto

Nonostante l'emergenza covid-19 lo stage si è completato senza problemi. Molto del lavoro svolto è stato fatto in modalità smart working e la collaborazione con il resto del team di sviluppo è stata supportata da strumenti di videoconferenza come Discord.

Il prodotto consegnato è risultato all'altezza delle aspettative e l'azienda ci ha confermato che questa applicazione verrà usata come prototipo da mostrare ad un loro potenziale cliente.

Un piccolo appunto sul prodotto è che, essendo stato il primo utilizzo della tecnologia Angular, ho scoperto solo tardi l'esistenza di alcune funzionalità avanzate che permettono di disaccoppiare ulteriormente il codice e di specializzare i servizi. Un esempio è HTTP interceptor, una funzionalità che permette di intercettare i messaggi HTTP e di effettuare delle azioni su di essi. Implementandolo avrei potuto creare una classe che si sarebbe dedicata ad inserire in ogni chiamata HTTP il token di autorizzazione invece

di lasciare l'onere ad ogni singolo servizio. Così facendo, anche in estensioni future del codice, non sarebbe stato necessario preoccuparsi dell'autorizzazione perché sarebbe stato gestito da una classe completamente indipendente e specializzata.

Gli strumenti utilizzati sono stati più che sufficienti a portare a termine il lavoro. In particolare, Visual Studio Code mi ha fatto scoprire un editor che, pur essendo gratuito, risulta essere all'altezza delle controparti a pagamento che ho utilizzato fino ad ora. Il punto di forza di Visual Studio Code (1.5) è la vasta scelta di estensioni che coprono moltissimi linguaggi di programmazione, sviluppate da altri programmatori e messe a disposizione di tutti.

Un altro strumento risultato molto utile, che sicuramente riutilizzerò in futuro, è stato Stoplight Studio (1.5) che ha permesso di progettare le API in modo semplice e di essere usato come punto di riferimento per mantenere allineati [front end](#) e [back end](#).

6.3 Valutazione personale

L'esperienza di stage è stata molto stimolante e costruttiva e mi ha permesso una crescita personale, affacciandomi per la prima volta nel mondo del lavoro. Mi ha dato l'opportunità di conoscere di più me stesso, i miei limiti e le mie capacità in campo lavorativo per migliorare le mie potenzialità e per interagire meglio e con più facilità con gli altri.

In conclusione, grazie a questo stage, ho avuto la possibilità di mettere in pratica diversi concetti studiati all'università, di migliorare le mie abilità di gestione del tempo e di cooperare più efficacemente nel lavoro di squadra.

Glossario

API in informatica con il termine *Application Programming Interface API* (ing. interfaccia di programmazione di un'applicazione) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. [77](#)

Back end in informatica il termine *back end* denota la parte non visibile all'utente di un programma e che permette l'effettivo funzionamento delle interazioni con il front end. [2-5](#), [42](#), [61-63](#), [72](#), [74](#)

CI/CD *Continuous Integration/Continuous Delivery* è un metodo per la distribuzione frequente delle app ai clienti, che prevede l'introduzione dell'automazione nelle fasi di sviluppo dell'applicazione. Principalmente, si basa sui concetti di integrazione, distribuzione e deployment continui. [77](#)

Endpoint in informatica il termine *endpoint* indica una interfaccia esposta tramite un canale di comunicazione. Gli *endpoint* favoriscono uno strato di astrazione alla programmazione che permette a sistemi software diversi di comunicare tra loro. [3](#), [5](#), [42](#)

Front end in informatica il termine *front end* denota la parte visibile all'utente di un programma e con cui egli può interagire, tipicamente un'interfaccia utente. [iii](#), [2-4](#), [42](#), [66](#), [72](#), [74](#)

GPS *GPS, Global Positioning System* (ing. Sistema di Posizionamento Globale) è un sistema elettronico che permette di individuare con precisione la posizione di un oggetto in movimento mediante l'uso di un satellite. [77](#)

ICT in informatica con il termine *Information and Communications Technology* (tecnologie dell'informazione e della comunicazione) si indica l'insieme dei metodi e delle tecniche utilizzate nella trasmissione, ricezione ed elaborazione di dati e informazioni. [77](#)

IntelliSense una funzione di completamento del codice in alcuni ambienti di programmazione che accelera il processo di codifica delle applicazioni riducendo gli errori di battitura e altri errori comuni. [4](#)

Screen reader uno *screen reader* (ing. letteralmente lettore dello schermo) è un'applicazione software che identifica e interpreta il testo mostrato sullo schermo di un computer, presentandolo tramite sintesi vocale. [65](#), [66](#)

UML in ingegneria del software *UML*, *Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico. [77](#)

Acronimi

API [Application Program Interface](#). 3, 4, 42, 74, 75

CI/CD [Continuous Integration/Continuous Delivery](#). 39, 75

GPS [Global Positioning System](#). 25, 35, 75

ICT [Information and Communications Tecnology](#). 1, 75

UML [Unified Modeling Language](#). 11, 76

WAI-ARIA [Web Accessibility Initiative - Accessible Rich Internet Applications](#). 66,
77

Bibliografia

Siti web consultati

- [1] *Angular*. URL: https://www.mrwebmaster.it/javascript/introduzione-angular_12716.html (cit. a p. 39).
- [2] *Angular Material*. URL: <https://material.angular.io/> (cit. a p. 40).
- [3] *Geolocalizzazione mezzi*. URL: <https://www.localizzazionemezzi.it/>.
- [4] *Git*. URL: <https://git-scm.com/> (cit. a p. 4).
- [5] *Jasmine*. URL: <https://jasmine.github.io/> (cit. a p. 40).
- [6] *Schema applicazione Angular*. URL: <https://www.targetweb.it/guida-angular2-introduzione-al-framework-del-futuro/> (cit. a p. 40).
- [7] *Sync Lab*. URL: <https://www.synclab.it> (cit. a p. 1).
- [8] *Typescript*. URL: https://www.mrwebmaster.it/javascript/introduzione-typescript_12482.html (cit. a p. 39).
- [9] *WAI-ARIA*. URL: <https://www.w3.org/WAI/standards-guidelines/aria/>.