Politecnico di Torino

Politecnico di Torino

Ingegneria Informatica

A.a. 2022/2023

Sessione di laurea Dicembre 2022

ATP Sassari analytics: applicazione web per l'analisi di dati ottenuti da veicoli dotati di sensori

Relatore:

Luigi De Russis

Tutor aziendali:

Stefano Salaris

Alberto Secondi

Candidato:

Sofia Catalano

Sommario

Il presente lavoro di tesi effettua una ricerca sulle possibilità, problemi e soluzioni offerte nel campo del monitoraggio dell'attività dei veicoli dell'azienda del trasporto pubblico dell'area di Cagliari e Sardegna ATP Sassari, attraverso l'analisi di dati sul tempo di viaggio ottenuti da veicoli dotati di sensori GPS. I sensori GPS rappresentano il sistema di localizzazione automatica dei veicoli più comunemente utilizzati, principalmente perché sono una tecnologia collaudata nel tempo e relativamente economica. Dopo una ricerca dettagliata sulle modalità di studio e filtraggio dei dati ricevuti e sui vari scenari applicativi, si è pensato di implementare una applicazione web che permettesse di produrre statistiche rilevanti per gli amministratori dell'azienda in modo da poter monitorare i propri mezzi di trasporto e ricavare risultati utili.

In particolare, in una prima fase del lavoro si sono studiate le modalità di raccolta e memorizzazione dei dati. In una seconda fase, i dati raccolti sono stati studiati, al fine di riscontrare tutte le possibili anomalie e procedere ad una fase di preelaborazione dei dati: sono stati identificati gli outliers e corrette le inconsistenze, in modo da estrarre dai dati le caratteristiche e le informazioni rilevanti. Infine, sulla base di queste osservazioni, è stata progettata e implementata un'applicazione web con l'obiettivo di supportare gli amministratori dell'ATP di Sassari nel tracciare e analizzare i percorsi effettuati dai propri veicoli, e dunque di porsi come strumento per monitorare le prestazioni e le problematiche degli autobus lungo i tragitti. L'applicazione mira a garantire una semplice ma efficace esperienza di navigazione: l'utente ha la possibilità di visualizzare le informazioni supportato da una sidebar che indica le principali macro-aree: statistiche di carattere generale, analisi dei percorsi e analisi degli autobus. Per ogni percorso è possibile usufruire di una rappresentazione grafica, attraverso la visualizzazione su mappa della relativa linea e delle fermate. È anche possibile mettere un percorso a confronto con un altro della stessa linea, in un qualsiasi o preciso periodo di tempo. Infine, è possibile visionare statistiche relative ad un singolo autobus.

Per ciò che concerne le scelte tecnologiche, queste risultano conformi ai trend attuali: in particolare per l'applicazione web è stato scelto un ambiente interamente basato su Javascript, utilizzando la libreria React.js per lo sviluppo di una Single Page Application, e Node.js per la parte server. Inoltre, React Router.js è la libreria di routing utilizzata per simulare la navigazione multi-pagina tra i vari componenti dell'applicazione React. Questi sono stati realizzati grazie al supporto della libreria Material-UI, in modo da rendere la User Interface e la User Experience in più possibile usabili, veloci, utili, funzionali e graficamente piacevoli, in accordo con le linee guida del Material Design di Google. Per quanto riguarda il backend, la libreria Express.js è stata utilizzata per la creazione di RESTful APIs di routing e per rispondere alle richieste HTTP.

Ringraziamenti

Ringrazio il mio professore Luigi De Russis e i tutors aziendali, Stefano Salaris e Alberto Secondi, per avermi guidata con grande disponibilità e professionalità durante la realizzazione del presente progetto di tesi.
Un ringraziamento speciale va alla mia famiglia che con immenso amore mi ha sostenuta in qualunque momento, gliene sarò per sempre grata.
Grazie a Giuseppe, che mi ha supportata in questi anni e con cui spero di condividerne ancora altri.
Grazie a mia zia Lucia, ragazza disabile che ha lottato non per essere normale, ma per essere sé stessa e mi ha insegnato che i limiti spesso non sono reali ma solo negli occhi di chi guarda.
Grazie ai miei nonni Fina e Gaspare, per le preghiere e per aver creduto in me.
Grazie ai miei nonni Lia e Gaspare, per avermi insegnato il valore di sorridere di fronte ad ogni avversità.
Grazie a mia nonna Franca, per tutte le benedizioni che mi dava.
Grazie ai miei zii e cugini, per l'affetto che mi hanno sempre trasmesso.
Grazie a Eleonora e Alessia, le amiche di sempre, su cui poter sempre contare.
Grazie a Mariagrazia, l'amica con cui ho condiviso momenti importanti di questo percorso.
Per finire vorrei ringraziare tutti voi qui presenti per aver voluto condividere con me questo giorno, rendendolo ancora più indimenticabile!

Table of Contents

Elenco delle figure VIII

1 Introduzione 1
- 1.1 Obiettivo . . . 2
- 1.2 Struttura della tesi . . . 2

2 Architettura e strumenti di sviluppo 3
- 2.1 Architettura di un'applicazione web-based . . . 3
- 2.2 Pattern architetturali . . . 5
  - 2.2.1 Multi-Page Application . . . 6
  - 2.2.2 Single Page Application Architecture . . . 6
  - 2.2.3 Progressive Web Application . . . 7
  - 2.2.4 Isomorphic Application Architecture . . . 7
- 2.3 Strumenti e tecnologie di sviluppo . . . 7
  - 2.3.1 Node.js . . . 8
  - 2.3.2 Express.js . . . 10
  - 2.3.3 React.js . . . 11
  - 2.3.4 DBMS . . . 14

3 Progettazione 17
- 3.1 Analisi dell'architettura . . . 17
- 3.2 Processo di elaborazione dei dati . . . 18
  - 3.2.1 Raccolta e analisi dei requisiti . . . 18
  - 3.2.2 Raccolta dei dati . . . 19
  - 3.2.3 Correzione dei dati . . . 21
  - 3.2.4 Trasformazione dei dati . . . 22
  - 3.2.5 Visualizzazione dei risultati . . . 24
- 3.3 Progettazione dell'interfaccia . . . 24
- 3.4 Realizzazione dei prototipi . . . 24
  - 3.4.1 Figma . . . 25
  - 3.4.2 Prototipi . . . 25

3.5 Struttura DBMS 28

4 Implementazione 29
4.1 Caratteristiche generali 29
4.2 Autenticazione 30
4.3 Routing 31
4.4 Documentazione delle APIs 33
4.5 Connessione al database MySQL 34
4.6 Gestione degli errori 36
4.7 Librerie di terze parti 36
4.8 Implementazione delle funzionalità 37
4.8.1 Homepage 37
4.8.2 Dashboard 38
4.8.3 Schermata di analisi delle linee 39
4.8.4 Schermata delle dei veicoli 41

5 Conclusioni 42
5.1 Sviluppi futuri 42

Bibliografia 43

Elenco delle figure

2.1 Architettura di un'applicazione web 4
2.2 Confronto fra un'architettura MPA e SPA 6
2.3 Frameworks più utilizzati lato backend e frontend 8
2.4 Confronto chiamata di rete in Express.js e Node.js 10
2.5 Frameworks più utilizzati nel 2022 11
2.6 Componente React 12
2.7 Componente React con props definite 13
2.8 React useState Hook 13
2.9 Confronto fra prima e dopo l'introduzione di useEffect in React . . 14
2.10 DBMS più popolari ad Agosto 2022 15

3.1 Componenti dell'architettura proposta 17
3.2 Processo di elaborazione dei dati 18
3.3 Riga di dati estratta da un CSV file 19
3.4 Confronto tra distanza euclidea e DTW 23
3.5 Prototipo Sidebar 25
3.6 Prototipo Dashboard 26
3.7 Prototipo Homepage 26
3.8 Prototipo schermata di analisi delle linee 27
3.9 Prototipo schermata di analisi dei veicoli 27

4.1 Form di login 30
4.2 Form di registrazione 31
4.3 Homepage di ATP Sassari Analytics 37
4.4 Dashboard di ATP Sassari Analytics 38
4.5 Schermata dei risultati della linea SH_11B_32 39
4.6 Schermata delle statistiche del percordo con ID:1 40
4.7 Schermata dei risultati del veicolo BUS_0 41
4.8 Dettaglio problemi riscontrati nel percorso selezionato 41

Capitolo 1

Introduzione

I rapidi progressi nelle tecnologie dei sensori hanno portato a un maggiore utilizzo di tali dispositivi installati all'interno dei veicoli di nuova generazione, e le funzioni rispetto al passato superano la sola navigazione satellitare ampliando le proprie funzionalità. Il sistema GPS è uno strumento di grande importanza per quelle aziende il cui business si basa principalmente sull'utilizzo di una flotta di veicoli, in quanto contribuiscono sia a diminuire i costi sia ad aumentare la produttività aziendale, attraverso attività di raccolta e di elaborazione dei dati, fornendo informazioni significative per la gestione dei veicoli e del personale addetto. Un localizzatore satellitare permette infatti di monitorare in modo completo le attività di qualsiasi veicolo della flotta, ottimizzando i costi e migliorando la performance aziendale. In particolare, l'azienda di trasporto pubblico di Sassari ha dotato i propri veicoli di sistemi di localizzazione con tracciamento passivo in cui la memorizzazione dei dati avviene all'interno del dispositivo, al contrario del tracciamento attivo in cui l'invio dei dati avviene ad un database centralizzato per il monitoraggio in tempo reale.

Il lavoro proposto nasce al fine di supportare l'azienda nel lavoro di gestione e controllo del parco veicoli, e di analisi dei percorsi effettuati. La quantità di dati raccolta è di grande valore, poiché permette di migliorare tutti gli aspetti relativi alla gestione flotta aziendale, identificando ad esempio il tempo trascorso alla guida del veicolo, i chilometri percorsi, la data e l'ora di partenza e le medesime d'arrivo.

1.1 Obiettivo

Il lavoro di tesi si pone come obiettivo principale l'analisi, la progettazione e lo sviluppo di un'applicazione web di supporto per l'azienda di trasporto pubblico ATP di Sassari. L'applicazione ha lo scopo di permettere all'azienda di:

- Ottenere statistiche di carattere generale per mezzo di una dashboard,
- Analizzare le linee con le relative corse,
- Analizzare i veicoli.

1.2 Struttura della tesi

Il lavoro di tesi è stato sviluppato in più capitoli che trattano i seguenti argomenti:

- **Capitolo 2**: fornisce una descrizione generale dell'architettura di un'applicazione Web-based, e dei tools e tecnologie utilizzate per lo sviluppo di ATP Sassari Analytics.
- **Capitolo 3**: fornisce una descrizione dell'analisi dei requisiti e della progettazione dell'architettura proposta, dettagliandone le funzionalità proposte.
- **Capitolo 4**: fornisce una descrizione dell'implementazione dell'applicazione web.
- **Capitolo 5**: fornisce una valutazione dei risultati ottenuti.
- **Capitolo 6**: fornisce una valutazione di eventuali sviluppi futuri.

Capitolo 2

Architettura e strumenti di sviluppo

2.1 Architettura di un'applicazione web-based

Le applicazioni web[1] sono un tipo di sistema distribuito, basato sulle tecnologie e sugli strumenti del web che adottano come modello di interazione il client-server. I sistemi distribuiti sono composti da più processi che cooperano tra loro perseguendo un unico obiettivo e che, nella maggior parte dei casi, risiedono su macchine diverse e comunicano attraverso la rete scambiandosi messaggi secondo precisi protocolli.

Le applicazioni web si pongono come valida alternativa alle tradizionali applicazioni Client/Server per vari motivi[2]:

- facilità di distribuzione e aggiornamento: un'applicazione web risiede interamente sul server, per cui la sua pubblicazione coincide con la distribuzione e l'aggiornamento ed è automaticamente disponibile a tutti gli utenti;
- accesso multipiattaforma: l'accesso all'applicazione è indipendente dall'hardware e dal sistema operativo utilizzato dagli utenti;
- riduzione del costo di gestione: l'uso di Internet come infrastruttura per un'applicazione web riduce notevolmente sia i costi di connettività che i costi di gestione dei client;
- scalabilità: un'applicazione web ben progettata può crescere insieme alle esigenze dell'azienda senza particolari problemi.

Nella maggior parte dei casi, l'architettura di un'applicazione web si sviluppa su due o più livelli, consentendo di identificare il ruolo di ciascun componente di

un'applicazione e apportare facilmente modifiche al livello corrispondente senza influire sull'applicazione complessiva:

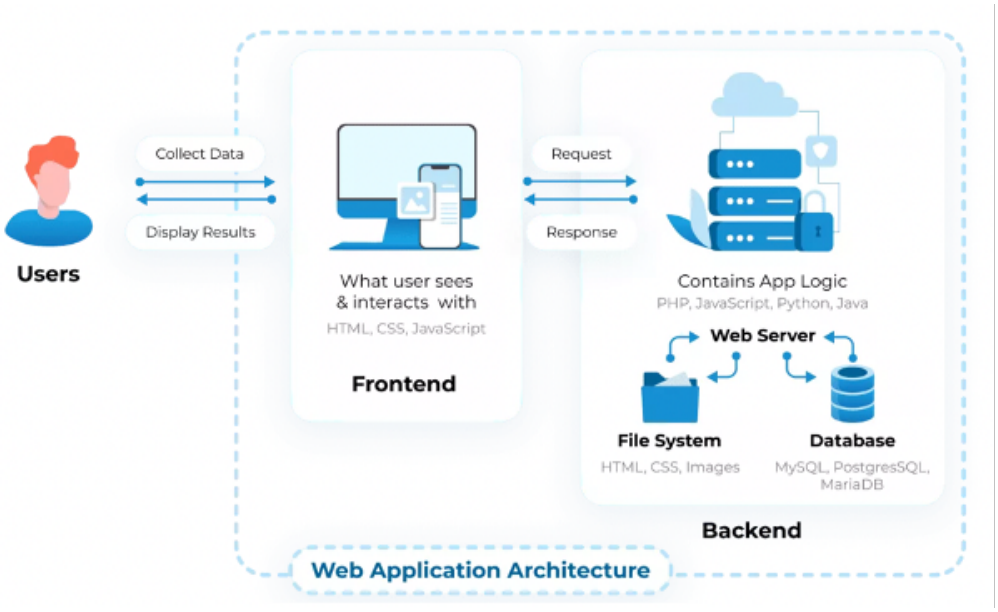


Figura 2.1: Architettura di un'applicazione web

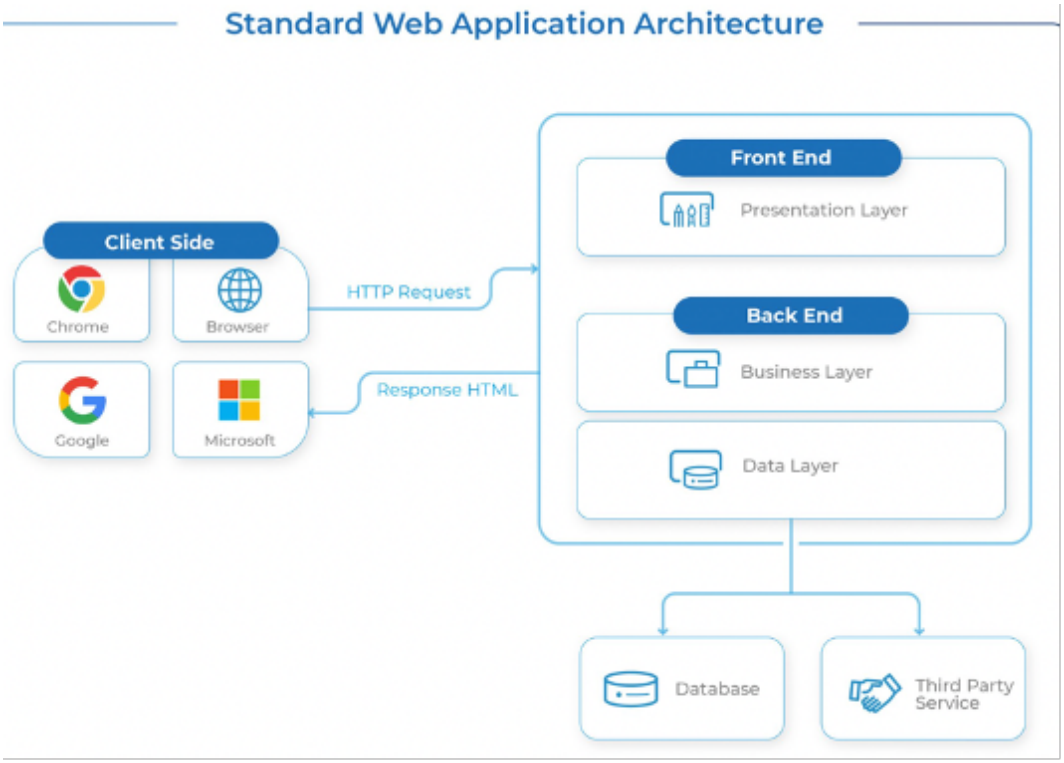
- **Frontend**: di solito ospitato in un browser web, offre un ambiente di esecuzione generico, adatto a visualizzare l'interfaccia utente grafica (GUI) e a fornire un facile accesso al sistema remoto tramite HTTP (unico protocollo con cui i browsers possono interagire).

- **Backend**: spesso creato utilizzando un approccio a più layers. Ogni layer interagisce solo con quelli adiacenti, limitando le reciproche dipendenze e specificando i punti di contatto in termini di interfacce software. Un tale approccio aumenta la modularità del codice, introducendo il principio di 'separation of concerns' tra i moduli. In particolare, si distinguono almeno i seguenti livelli:

  - **Presentation Layer**: rappresenta l'interfaccia utente dell'applicazione e si occupa di acquisire dati, ricevere le richieste HTTP provenienti dagli utenti e inviare opportune risposte.
  - **Business Layer**: si occupa delle elaborazioni dei dati in base alla cosiddetta business logic, cioè all'insieme delle regole per cui i dati sono considerati significativi e le loro relazioni consistenti. Le elaborazioni del presente livello generano i risultati richiesti dall'utente.

– **Data layer**: rappresenta l'insieme dei servizi offerti da applicazioni indipendenti dal Web, come ad esempio uno o più gestori di database, un sistema di gestione di posta elettronica, ecc. In particolare, il DBMS ha il compito di rendere persistenti i dati dell'applicazione e garantirne la coerenza rispetto al modello e i vincoli di business.

Ocorre precisare che non sempre i livelli logici di un'applicazione Web corrispondono a locazioni fisiche sulla rete (livelli fisici). Si va dal caso in cui tutti e tre i livelli risiedono sulla stessa macchina a varie altre distribuzioni fino alla corrispondenza di ciascun livello con una macchina fisica.



2.2 Pattern architetturali

La creazione di un'applicazione web avviene seguendo uno dei diversi tipi di architettura utilizzati nel web. Di seguito vengono illustrati i principali pattern architetturali.

2.2.1 Multi-Page Application

L'architettura multi page application rappresenta il modo tradizionale di creare applicazioni web secondo il quale il client per ogni pagina da mostrare esegue una richiesta al server il quale, dopo aver elaborato la richiesta, restituisce il contenuto della pagina con le relative risorse. Secondo questo pattern si esegue un rendering server-side, in quanto è il server che invia al client la pagina corretta da visualizzare e quindi, bisogna attendere sempre il caricamento della nuova pagina. Quest'architettura è consigliata soprattutto per le applicazioni che non si concentrano molto sull'interfaccia utente e che cercano solo una maggiore scalabilità o sicurezza, in quanto per pagine web complesse o pesanti, poiché il server può impiegare del tempo per generare completamente la pagina, con conseguente ritardo nel primo rendering.

2.2.2 Single Page Application Architecture

Secondo tale architettura l'intera applicazione si basa su una singola pagina che è esattamente la stessa per ogni diverso URL, e il cui contenuto viene aggiornato dinamicamente man mano che l'utente interagisce con l'applicazione. Le Single Page Application sono state introdotte per superare le limitazioni dell'approccio tradizionale, al fine di fornire un'esperienza veloce e senza interruzioni agli utenti finali. Tuttavia, possono essere difficili da ottimizzare per la SEO.

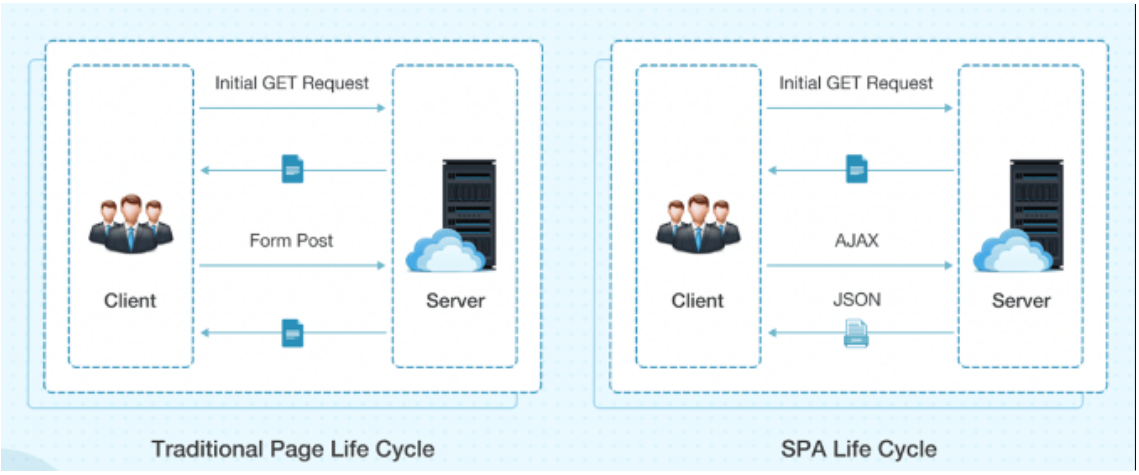


Figura 2.2: Confronto fra un'architettura MPA e SPA

2.2.3 Progressive Web Application

L'architettura Progressive Web Application (PWA) si basa sull'architettura Single Page Application Architecture e fornisce funzionalità offline all' applicazione web. Tecnologie come Capacitor e Ionic sono utilizzate per costruire PWA in grado di fornire agli utenti un'esperienza uniforme su tutte le piattaforme. Come le SPA, le PWA sono fluide e senza interruzioni. Allo stesso tempo, può essere difficile ottimizzare queste applicazioni per la SEO e gli aggiornamenti delle applicazioni installate possono essere difficili da eseguire.

2.2.4 Isomorphic Application Architecture

L'architettura isomorfica mette insieme SPA con MPA, ottenendo un comportamento ibrido secondo il quale alcune pagine vengono costruite dinamicamente lato client mentre altre vengono restituite dal server. Grazie alla SPA, l'elaborazione e l'interfaccia utente sul client sono superveloci. Inoltre, grazie al rendering lato server, otterrete un rendering iniziale rapido e un supporto SEO e di link completo.

Per lo sviluppo di ATP Sassari Analytics è stata scelta un'architettura Single Page Application in quanto garantisce alti livelli di

- **velocità**: l'applicazione si interfaccia direttamente con il server, ne richiede i dati all'occorrenza e li restituisce in tempo reale senza necessità di dover ricaricare la pagina.
- **performance**: grazie alla velocità di risposta, l'intera esperienza utente è più fluida, immediata.
- **scalabilità**: essendo progettata per essere modulare, la SPA permette di aggiungere in maniera progressiva nuove funzionalità e far crescere nel tempo una soluzione sempre più adatta alle esigenze.

2.3 Strumenti e tecnologie di sviluppo

Lo sviluppo di un'applicazione web presuppone l'utilizzo di strumenti e tecnologie al fine di velocizzare il lavoro e ridurre i tempi di realizzazione dell'applicazione. Di seguito sono mostrati i principali framework utilizzati[3] rispettivamente per l'elaborazione lato backend e frontend:

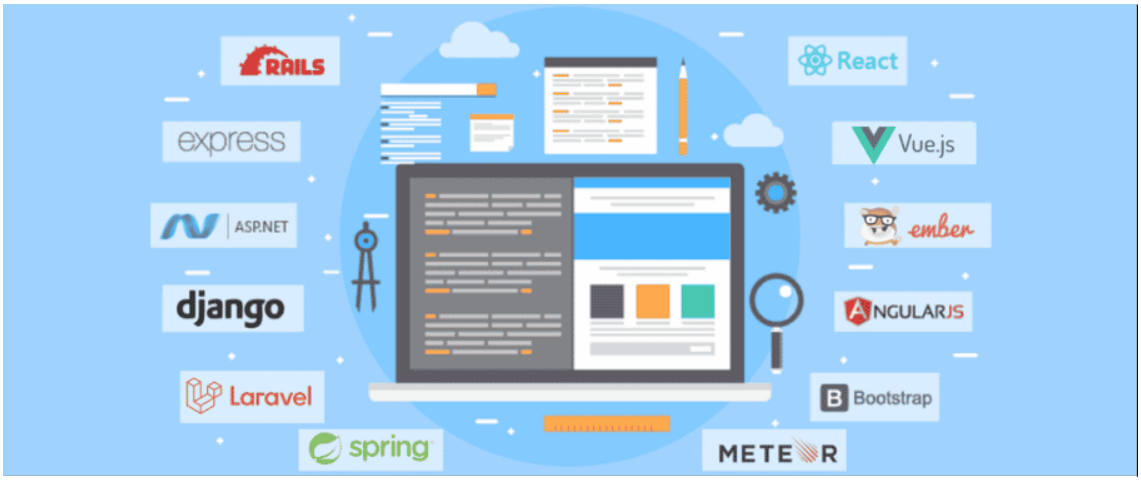


Figura 2.3: Frameworks più utilizzati lato backend e frontend

Per lo sviluppo di ATP Sassari Analytics è stato scelto un ambiente interamente basato su Javascript, utilizzando React.js come framework per l'elaborazione lato frontend, Node.js ed Express per il backend, MySQL come DBMS. Di seguito vengono illustrate le motivazioni di tali scelte tecnologiche, assieme alle loro principali caratteristiche.

2.3.1 Node.js

Node.js[4] è un runtime system open source multipiattaforma orientato agli eventi per l'esecuzione di codice JavaScript, costruito sul motore JavaScript V8 di Google Chrome.

Molti dei suoi moduli base sono scritti in JavaScript, ed è possibile scrivere nuovi moduli nello stesso linguaggio di programmazione. Inoltre, mette a disposizione lo strumento **NPM, Node Package Manager**, che rappresenta il più grande ecosistema di librerie javascript open source al mondo, permettendo di scaricare tali moduli e mantenerli aggiornati durante lo sviluppo del software. Questo rende lo sviluppo del codice più veloce e il codice più manutenibile.

Node.js ha un'architettura orientata agli eventi che rende possibile l'I/O asincrono. A differenza di altri ambienti di esecuzione, un processo Node non si affida al multitreading per l'esecuzione parallela del codice, poiché presenta dei problemi che possono essere difficili da isolare e correggere in un ambiente di produzione, come ad esempio i deadlock e la protezione delle risorse condivise tra più thread.

Un modello I/O asincrono basato sugli eventi, al contrario, offre una gestione delle applicazioni più efficiente e scalabile, impedendo che l'applicazione si blocchi nell'attesa di un'operazione di I/O.

Considerato come un software accessibile, Node.js gode di una grande comunità pronta a offrire feedback e supporto, ed è la stessa comunità che sfrutta la posizione open source di Node.js, creando strumenti per semplificare il processo di sviluppo per tutti.

Node.js si integra benissimo con database relazionali e con frontend Javascript come React.

2.3.2 Express.js

Express.js[5] è un framework di applicazioni web che dispone del più potente e robusto sistema per la creazione di API di routing con Node.js che assiste l'applicazione in risposta a una richiesta del client attraverso un particolare endpoint. Inoltre, comprende una serie di middleware[1] che permettono di creare un processo di sviluppo fluido e ininterrotto.

Fornisce uno strumento di interfaccia a riga di comando chiamato Node Package Manager, da cui è possibile reperire moduli, pacchetti e risorse aggiuntive messi a disposizione dalla community per creare applicazioni web affidabili.

Si tratta di un framework molto flessibile ed è più veloce di qualsiasi altro framework Node.js. Nel corso degli anni, Express.js ha dimostrato di essere molto scalabile grazie al numero di grandi aziende che lo utilizzano quotidianamente sui loro server. Gestisce le richieste e le risposte degli utenti in modo efficiente e non richiede alcuna configurazione aggiuntiva quando si sviluppa un'applicazione web su larga scala. Di seguito si può notare come la gestione di una chiamata di rete attraverso Express.js sia più chiara rispetto a uno sviluppo nativo in Node.js.

```
// Requiring module
const express = require('express');
const app = express();

// Handling '/' request
app.get('/', (req, res) => {
    res.send('<h2>Hello from Express.js server!!</h2>');
});

// Handling '/about' request
app.get('/about', (req,res) => {
    res.send('<h2>GeeksforGeeks- Express.js</h2>');
});

// Server setup
app.listen(8080, () => {
    console.log('server listening on port 8080');
});
```

```
// Requiring the module
const http = require('http');

// Creating server object
const server = http.createServer((req, res) => {
    const url = req.url;

    if(url === '/') {
        res.write('<html>');
        res.write(
'<head><title>GeeksforGeeks</title><head>');
        res.write(
'<body><h2>Hello from Node.js server!!</h2></body>');
        res.write('</html>');
        return res.end();
    }

    if(url === '/about') {
        res.write('<html>');
        res.write(
'<head><title>GeeksforGeeks</title><head>');
        res.write(
'<body><h2>GeeksforGeeks- Node.js</h2></body>');
        res.write('</html>');
        return res.end();
    }
});

// Server setup
server.listen(3000, () => {
    console.log("Server listening on port 3000")
});
```

Figura 2.4: Confronto chiamata di rete in Express.js e Node.js

[1]I middleware sono codici che vengono eseguiti prima che una richiesta HTTP raggiunga il gestore del percorso o prima che un cliente riceva una risposta, dando al framework la possibilità di eseguire uno script tipico prima o dopo la richiesta del client.

2.3.3 React.js

React JS è una libreria Javascript per lo sviluppo di interfacce utente rapide ed interattive. Sviluppata da Facebook nel 2011 (allo scopo iniziale di gestire le news feed all'interno del social stesso), la libreria React è stata collaudata e distribuita a partire dal 2013, ed è attualmente una delle librerie frontend più comunemente utilizzate per lo sviluppo web. Una recente ricerca condotta da Statista[2] mostra che la quota del 35,9% degli intervistati utilizza ReactJS per lo sviluppo del frontend:

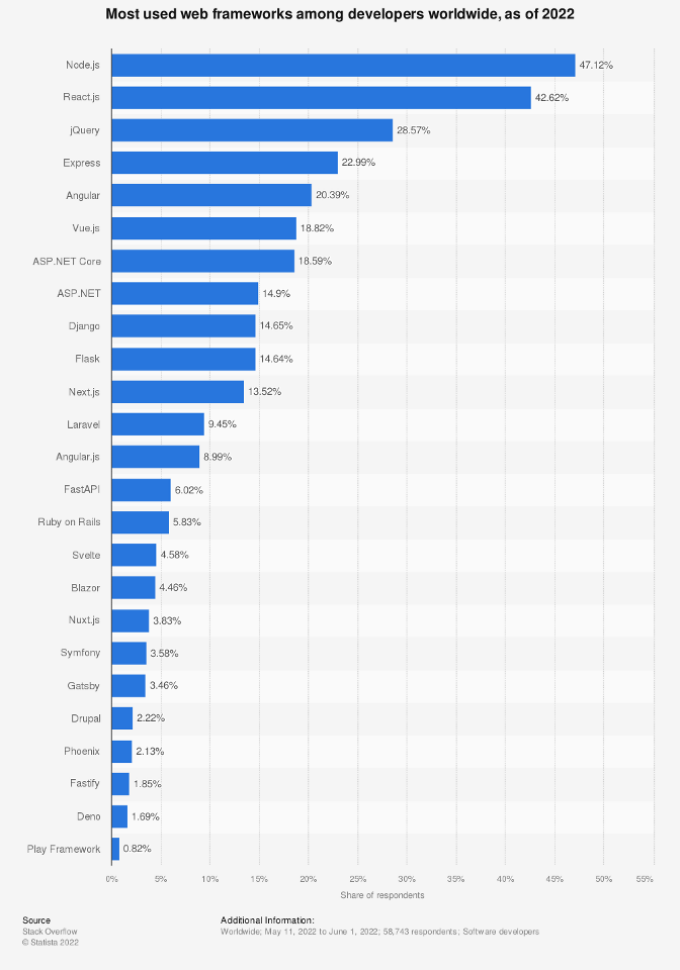


Figura 2.5: Frameworks più utilizzati nel 2022

[2]Statista è un sito web tedesco per la statistica, che rende disponibili dati raccolti da istituzioni che si occupano di ricerca di mercato e di opinioni, così come statistiche riguardanti l'ambito economico e statale. L'azienda afferma che sulla sua piattaforma sono presenti statistiche riguardanti più di 80.000 temi e provenienti da più di 22.500 fonti.

Le motivazioni dietro la popolarità di React come framework per lo sviluppo di applicazione web sono molteplici.

React è una libreria **dichiarativa** in quanto permette di creare l'interfaccia e il sistema di eventi senza mai dover interagire direttamente con il DOM, al contrario di quando avviene utilizzando un approccio imperativo (ad esempio con JQuery). Infatti, una delle più grandi novità introdotte da React è l'utilizzo di un virtual DOM, che rende le applicazioni web più veloci. Il virtual DOM[6] confronta gli stati precedenti dell'applicazione e aggiorna solo gli elementi nel real DOM che sono stati modificati, invece di aggiornare nuovamente tutti i componenti, come fanno le applicazioni web convenzionali.

React (così come Angular e Vue) ha un'architettura **component-based**[7]: i componenti sono gli elementi costitutivi di qualsiasi applicazione React. Si tratta di elementi modulati e isolati che fanno parte dell'interfaccia utente e che possono essere combinati fra loro per creare componenti più complessi. Ciascun componente, definito come una funzione Javascript, segue una propria logica e determinati controlli e può essere riutilizzato in tutta l'applicazione, il che a sua volta riduce drasticamente i tempi di sviluppo dell'applicazione stessa.

```
1   function Welcome() {
2     return <h1>Hello World!</h1>;
3   }
```

Figura 2.6: Componente React

La parte di codice restituita dalla funzione viene definita JSX[8], ed è un'estensione sintattica Javascript che permette di scrivere la struttura dei vari componenti usando una sintassi simile al linguaggio HTML, al fine di creare dei template dinamici, in quanto all'interno è possibile inserire qualsiasi espressione logica che verrà valutata e renderizzata nel codice. Il Team React, nella documentazione ufficiale, suggerisce di usare JSX per lo sviluppo delle applicazioni web; in ogni caso, ogni espressione JSX verrà trasformato in codice Javascript interpretabile da React al momento della compilazione.

Al fine di supportare la comunicazione tra componenti, React adotta un modo esplicito per passare i dati tra i componenti che rende molto facile gestire i cambiamenti di stato ed individuare gli impatti sul resto dell'applicazione. Questo è chiamato **Flusso Dati Unidirezionale** perché i dati scorrono in una direzione, dal componente padre ai figli. Gli aggiornamenti sono mandati all'elemento padre che si occupa di eseguirli.

Sulla base di questo concetto, React incapsula gli attributi definiti dall'utente in un oggetto che prende il nome di **"props"**. Il contenuto delle props è immutabile, cioè non può mai essere modificato all'interno del componente che lo riceve.

```
1   function Welcome(props) {
2     return <h1>Hello, {props.name}</h1>;
3   }
4
5   function App(){
6       return <Welcome name={"Sofia"}/>;
7   }
```

Figura 2.7: Componente React con props definite

A partire dalla versione 16.8, React ha introdotto il meccanismo degli **Hooks**, permettendo la gestione dello stato e della sua evoluzione e l'esecuzione di effetti collaterali all'interno di componenti funzionali, rendendoli così molto più potenti.

In particolare, **useState** è utilizzato per permettere il salvataggio, a livello di un singolo componente, di un dato che si modifica sulla base di eventi scatenati su alcune parti del componente stesso. Tale Hooks accetta in input il valore iniziale dello stato e restituisce un array costituito da due elementi: il valore corrente dello stato e la funzione che permette di modificarlo.

```
1   import React, { useState } from 'react';
2
3   function Esempio() {
4       const [contatore, setContatore] = useState(0);
5
6       return (
7           <div>
8               <p>Hai cliccato {contatore} volte</p>
9               <button onClick={() => setContatore(contatore + 1)}>
10                  Cliccami
11              </button>
12          </div>
13      );
14  }
```

Figura 2.8: React useState Hook

Al fine di permettere l'esecuzione di effetti collaterali, componentDidMount, componentDidUpdate, e componentWillUnmount sono stati sostituiti da un Hook chiamato **useEffect**[9]. Tale hook riceve in ingresso una callback che rappresenta l'azione da eseguire e un array di dipendenze, che suggerisce quando l'azione deve essere eseguita. React memorizza la funzione passata come parametro che viene eseguita dopo ogni render del componente e ogni volta che una variabile osservata cambia valore.

```
class Example extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 0
    };
  }

  componentDidMount() {
    document.title = `You clicked ${this.state.count} times`;
  }
  componentDidUpdate() {
    document.title = `You clicked ${this.state.count} times`;
  }

  render() {
    return (
      <div>
        <p>You clicked {this.state.count} times</p>
        <button onClick={() => this.setState({ count: this.state.count + 1 })}>
          Click me
        </button>
      </div>
    );
  }
}
```

```
import React, { useState, useEffect } from 'react';

function Example() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    // Update the document title using the browser API
    document.title = `You clicked ${count} times`;
  });

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

Figura 2.9: Confronto fra prima e dopo l'introduzione di useEffect in React

Tra gli hooks presenti all'interno della libreria React, oltre a quelli sopra citati, vi sono:

- useContext
- useRef
- useReducer
- useCallback
- useMemo

2.3.4 DBMS

Per ciò che concerne il DBMS, per lo sviluppo di ATP Sassari Analytics, si è scelto di usare MySQL[10]. In particolare, MySQL rappresenta un DBMS relazionale composto da un client a riga di comando e un server. È un software open-source, rilasciato a doppia licenza compresa la GNU General Public License ed è sviluppato per essere il più possibile conforme agli standard ANSI SQL e ODBC SQL.

Le motivazioni di tale scelta sono legate a diverse caratteristiche di MySQL:

- MySQL è lo standard di fatto per i siti web con volumi di traffico elevati, grazie al suo query engine ad alte prestazioni, alla capacità di inserimento dei dati estremamente veloce e al supporto delle funzioni web specializzate, come ad esempio le ricerche full text rapide.
- MySQL è la soluzione più avanzata a livello di scalabilità, grazie alla sua capacità di integrarsi con applicazioni le cui dimensioni non devono superare pochi MB, fino ad enormi data warehouse contenenti terabyte di informazioni. La flessibilità di piattaforma è una funzionalità chiave di MySQL, che è in grado di supportare i sistemi operativi Linux, UNIX e Window.
- MySQL garantisce alta disponibilità e solido supporto delle transazioni

Una recente ricerca[11] condotta da Statista mostra che MySQL occupa la seconda posizione tra i DBMS più popolari ad Agosto 2022.

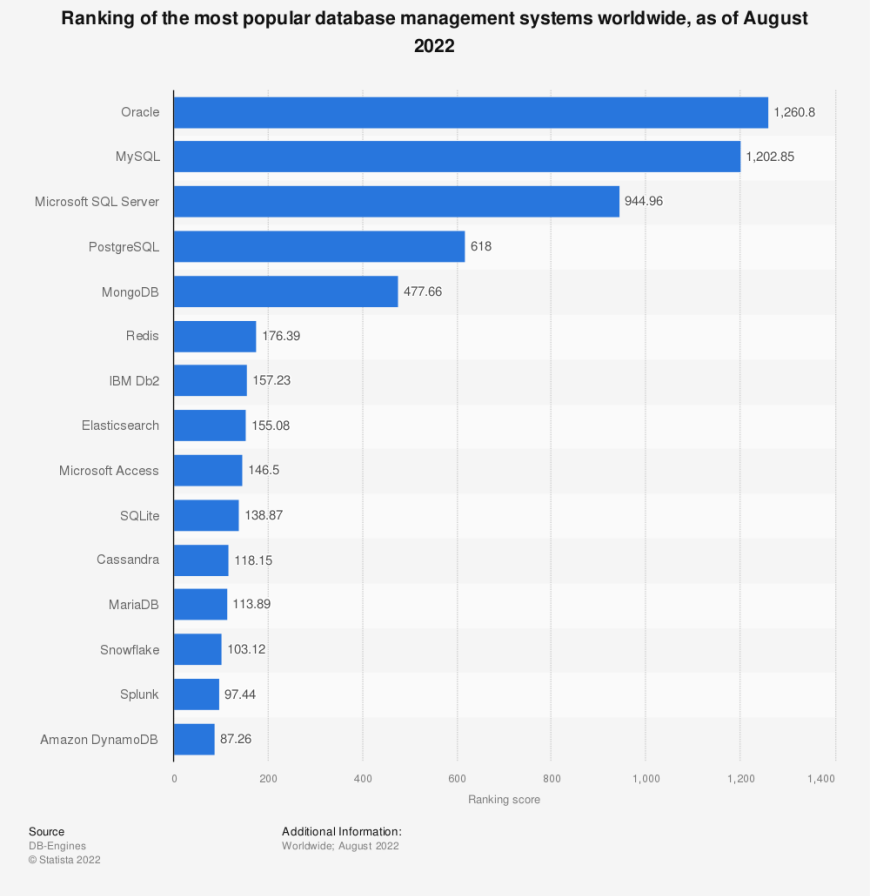


Figura 2.10: DBMS più popolari ad Agosto 2022

Attraverso Node.js è possibile interagire con MySQL: il modulo node *mysql2* può essere facilmente installato tramite il package manager di Node.js lancinado il seguente comando da terminale:

```
1 npm install mysql
```

In seguito, è possibile effettuare delle query a carico del database gestito inviandole direttamente tramite JavaScript.

Capitolo 3

Progettazione

In questo capitolo viene illustrata la fase di progettazione di ATP Sassari Analytics, necessaria per definire le specifiche funzionali e non dell'applicazione web.

3.1 Analisi dell'architettura

Come introdotto nel capitolo precedente, l'architettura dell'applicazione è costituita dai seguenti componenti:

- **Web browser**, rappresentato dai dispositivi degli utenti, tramite i quali è possibile accedere all'applicazione web.
- **Frontend**, realizzato in React.js, espone l'interfaccia grafica.
- **Server**, realizzato per mezzo di Node.js ed Express.js, che consente l'accesso alle API e di interrogare e ricevere dati dal database.
- **Database**, per la persistenza dei dati, tramite MySQL.

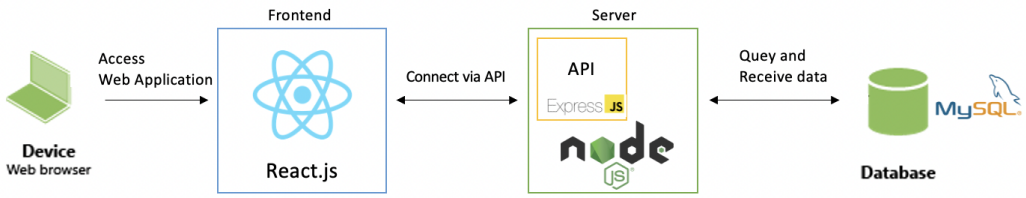


Figura 3.1: Componenti dell'architettura proposta

3.2 Processo di elaborazione dei dati

Il processo di elaborazione dei dati dell'applicazione web si è sviluppato a partire dall'analisi preliminare dei requisiti e dei dati raccolti. Tale processo, che ha portato alla realizzazione dell'applicazione web, ha rispettato i seguenti steps:

Figura 3.2: Processo di elaborazione dei dati

3.2.1 Raccolta e analisi dei requisiti

La fase primordiale del processo di analisi dei dati non può prescindere da un'attenta e chiara definizione di quella che è la necessità che l'analisi stessa ha come obiettivo. L'identificazione dei desiderata e del valore che l'analisi deve portare al business aziendale dell'ATP, ha aiutato a guidare le fasi successive e, infine, a comunicare opportunamente i risultati.
Sono stati dunque identificati i requisiti che seguono e che rappresentano una descrizione dei servizi del software e dei vincoli da rispettare sia in fase di sviluppo che durante la fase di operatività.

Requisiti funzionali

L'amministratore dell'azienda dovrà poter eseguire le seguenti azioni:

- Monitorare le corse, attraverso la conoscenza della velocità e direzione dei veicoli neli vari percorsi;
- riconoscere eventi quali frenate brusche o velocità eccessive;
- valutare la durata delle soste;
- monitorare i singoli autobus relativamente alle corse effettuate;
- ottenere statistiche di carattere generale

Requisiti non funzionali

- Dovrà essere presente un sistema di autenticazione al fine di permettere l'accesso all'applicazione web solo agli utenti amministratori;
- L'interfaccia grafica dovrà essere chiara e ben organizzata;

- Il software dovrà essere il più possibile manutenibile, prevede la suddivisione delle operazioni da svolgere in aree funzionali.
- Il software dovrà essere il più possibile scalabile, al fine di poter aggiungere funzionalità e ulteriori requisiti a posteriori.

3.2.2 Raccolta dei dati

Il processo di raccolta dei dati è avvenuto considerando due principali sorgenti:

- **General Transit Feed Specification (GTFS)**, che definisce un formato comune per gli orari dei trasporti pubblici e le relative informazioni geografiche. Si tratta di una collezione di file CSV[1] memorizzati con estensione *.txt*, il nome di ognuno dei quali rappresenta una tabella specifica:
  - **agency**, corrisponde alla tabella con le informazioni sull'azienda dei trasporti.
  - **route, trips, shapes**, definiscono i percorsi.
  - **stop_times**, definisce gli orari presso una fermata del mezzo di trasporto.
  - **stops**, definisce le informazioni geografiche di ogni fermata.
  - **calendar**, definisce la ricorrenza con cui avviene il passaggio di un mezzo di trasporto presso una fermata come i giorni ed il periodo di esercizio.
- **Dati ottenuti dai sensori** posti sui veicoli dell'azienda ATP e scaricati una volta che i mezzi di trasporto rientrano in deposito. Si tratta di file CSV memorizzati con estensione *.csv* il cui nome rappresenta l'identificativo del veicolo che ha registrato i dati e il timestamp iniziale di registrazione. Ad esempio, il file *BUS_0_0_180422-070136* è stato registrato dal veicolo 'BUS_0' il 18/04/2022 a partire dalle ore 07:01:36.

Di seguito viene riportata una riga di un file CSV con i relativi campi e vengono analizzati i diversi sensori del sistema installato all'interno dei veicoli dell'ATP che hanno permesso di registrare tali dati.

```
bus_id,date_timestamp,latitude,longitude,speed,accel_x,accel_y,accel_z,gyro_x,gyro_y,gyro_z,pressure,temperature,humidity

BUS_0;#030522-164332,40.754639,8.515427,16.240000#-140,125,1375,1680,-2240,-3990,1003.3621,31.00,40.75
```

Figura 3.3: Riga di dati estratta da un CSV file

[1]Il comma-separated values (abbreviato in CSV) è un formato di file basato su file di testo utilizzato per l'importazione ed esportazione di una tabella di dati.

Sistema GPS

I campi riguardanti la data e l'ora, assieme alle coordinate di latitudine e longitudine del veicolo, sono stati rilevati per mezzo di un sensore GPS.
GPS è l'acronimo di Global Positioning System, o **Sistema di Posizionamento Globale**, e indica una tecnologia basata sui satelliti in orbita, in grado di fornire la **posizione e l'ora esatta** a qualsiasi dispositivo dotato di un apposito ricevitore.
Il GPS utilizza un metodo di localizzazione chiamato *triangolazione*. Per conoscere la posizione esatta di qualunque luogo sull'intero pianeta, infatti, è sufficiente conoscere la distanza esatta da almeno tre diversi punti: il localizzatore GPS, dunque, può comunicare costantemente con tre satelliti diversi, grazie ai quali è possibile stabilire la sua posizione, con un margine di precisione abbastanza alto.
L'orologio a bordo del ricevitore, però, non è preciso quanto quello che si trova a bordo dei 3 satelliti impegnati nella triangolazione, dunque il ricevitore stesso non è in grado di conoscere da subito il momento esatto in cui il segnale è partito: per questo, viene sfruttato il segnale inviato da un quarto satellite, utile a "correggere" l'orologio a bordo del ricevitore e sincronizzarlo con quello satellitare. È questo il motivo per cui, in ogni punto della terra, un ricevitore GPS può ricevere i segnali da almeno 5 satelliti differenti (4 operativi più uno "di scorta").
Nonostante l'enorme accuratezza degli orologi atomici presenti sui satelliti, il sistema GPS ha una precisione ottima ma non assoluta, pari in condizioni ottimali a 5 metri o anche meno. Questo dipende da alcuni limiti nella precisione della ricezione del segnale, come il ritardo del segnale quando attraversa la ionosfera. Anche diversi valori di umidità o di pressione dell'aria possono introdurre un errore e quindi una perdita di precisione nel posizionamento. Infine, è molto rilevante il problema del "rimbalzo" del segnale fra edifici, pareti rocciose, canali, boschi eccetera.

Nel caso del sensore GPS posto nei veicoli dell'ATP, il tracciamento GPS è passivo, cioè i dati di localizzazione sono memorizzati nel ricevitore e devono essere scaricati una volta che i veicoli ritornano in sede.

Accelerometro

Il dato di accelerazione lungo i tre assi x, y, z è stato rilevato per mezzo di un accelerometro triassiale che misura simultaneamente vibrazioni nelle 3 direzioni ortogonali.
Nei valori misurati dall'accelerometro è inclusa la forza di gravità, per cui è stato necessario comprendere il modo in cui il dispositivo è orientato nello spazio per rimuovere la forza di gravità dai tre assi e isolare quindi l'accelerazione lineare.

Giroscopio

Il dato relativo alla velocità angolare lungo i tre assi x, y, z è stato misurato per mezzo di un giroscopio. Con tale strumento è possibile determinare l'orientamento del veicolo nello spazio, ed è un'informazione rilevante per interpretare correttamente il valore prodotto dall'accelerometro.

Altre misure

Le altre misure rilevate riguardano i dati di temperatura, pressione e umidità. Dal momento che il sistema di sensori in analisi è posto nelle vicinanze di altri sistemi, tali misurazioni risultano alterate. Per tale motivo, non sono state prese in considerazione ai fini del presente lavoro di tesi.

3.2.3 Correzione dei dati

In seguito all'identificazione dei dati rilevati dai sensori, si rende necessaria una fase preliminare di pulizia e standardizzazione dei dati (Data Cleansing), per correggere dati ridondanti, incompleti o errati. I dati presi in considerazione in questa fase sono stati quelli derivanti dalla seconda sorgente, dando per certa la correttezza della prima.

La pulizia dei dati è avvenuta seguendo due principali aspetti dei dati:

- **inaccuratezza**, ossia valori errati o che si discostano sensibilmente dai valori attesi;

- **incompletezza/inesattezza**, ossia mancanza del valore di alcuni attributi, o presenza di più attributi.

Per ciò che concerne l'inaccuratezza dei dati, i principali errori sono stati individuati nel campo **date_timestamp**, trattandosi spesso di errori dovuti alla propagazione dei bit, per cui ad esempio il timestamp '190422-102339' è stato memorizzato come '109422-102339'. In questo caso al fine di rilevare l'errore si è effettuato il parsing della stringa secondo il formato *DD-MM-YY hh:mm:ss* e verificata la validità della data.
Inoltre, sono stati individuati ed eliminati gli **outliners**, ossia i dati il cui valore si discostava molto dai valori limite attesi.

Per ciò che concerne l'incompletezza/inesattezza dei dati, sono state individuate alcune righe contenenti un numero di campi superiore rispetto a quello previsto. In questo caso, si è deciso di scartare tali records.

Al termine del processo di correzione dei dati, è stato prodotto un file di log contenente le seguenti informazioni:

- I dettagli di ogni record con errore;
- Il numero totale di records processati;
- Il numero totale di records con errore.

3.2.4 Trasformazione dei dati

Ricevuti i file CSV, si è resa fondamentale una trasformazione dei dati in ingresso, ossia consolidare e trasformare i dati in forme più appropriate per le analisi.
In particolare, lo scopo di tale processo è stato quello di aggiungere il dato relativo al percorso: a partire dalla traiettoria effettuata dai veicoli che dura un lungo periodo, sono stati estratti i diversi percorsi. Come accennato precedentemente, questi ultimi sono riportati nei file CSV del GTFS.

Dynamic Time Warping

L'idea iniziale dell'algoritmo per l'estrazione dei percorsi a partire dall'intera traiettoria era quella di trovare il percorso a distanza minima tra quelli presenti nel GTFS. Per calcolare le distanze è stato scelto l'algoritmo di Dynamic Time Warping[12], in quanto permette di calcolare la distanza tra serie temporali aventi lunghezze differenti. Infatti, nel GTFS ogni percorso è rappresentato da circa 350-400 records, mentre il sistema posto all'interno del veicolo registra un record ogni secondo, producendo una quantità di records superiore.
L'idea del DTW di confrontare array con lunghezza diversa è quella di creare corrispondenze uno-a-molti e molti-a-uno in modo da ridurre al minimo la distanza totale tra i due, a differenza della distanza euclidea che prevede un matching uno-a-uno.

Supponendo di avere due diversi array rosso e blu con differenti lunghezze, di seguito viene mostrata la differenza tra la distanza euclidea e quella calcolata con DTW:

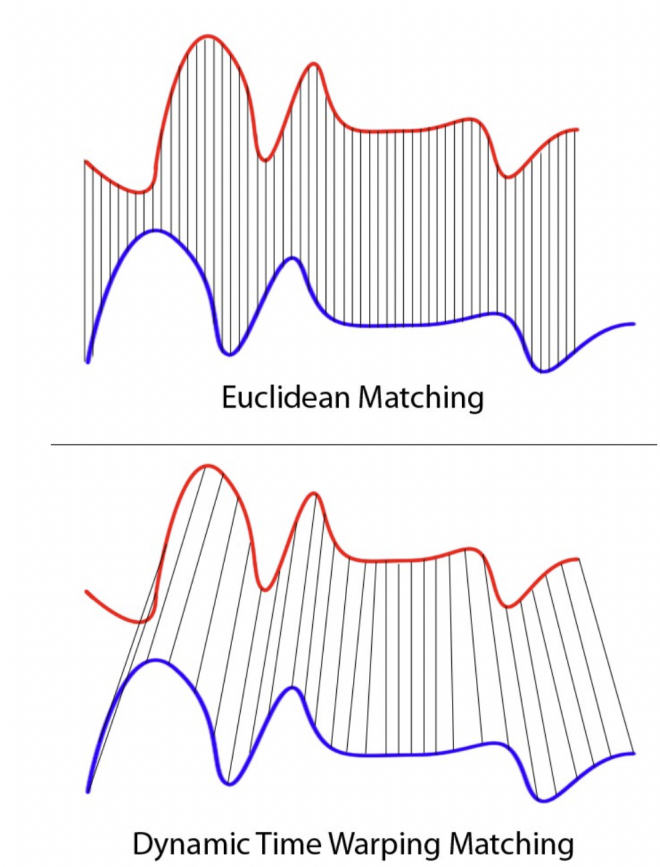


Figura 3.4: Confronto tra distanza euclidea e DTW

Chiaramente queste due serie seguono lo stesso schema, ma la curva blu è più lunga della rossa. Se si applicasse la corrispondenza uno-a-uno, mostrata in alto, la mappatura non è perfettamente sincronizzata e la coda della curva blu viene tralasciata.
DTW risolve il problema sviluppando una corrispondenza uno-a-molti in modo che gli avvallamenti e i picchi con lo stesso schema siano perfettamente abbinati e non vi siano parti escluse per entrambe le curve (mostrate in basso nella prima figura).

3.2.5 Visualizzazione dei risultati

Lo scopo della *Visualizzazione dei risultati* è quello di rendere la raccolta dati analizzata il più fruibile e accessibile possibile.
Nel presente progetto di tesi si è cercato di esplorare una serie di opzioni di visualizzazione, tra cui grafici a barre e a linee, tabelle, mappe geografiche e altri elementi visivi. Questi strumenti consentono agli utenti finali di analizzare i dati interagendo direttamente con una rappresentazione visiva di essi.
Comprendendo i dati analizzati più rapidamente si velocizzano anche le fasi decisionali nella procedura di business che l'azienda sceglie di attuare.

3.3 Progettazione dell'interfaccia

La progettazione dell'interfaccia grafica è avvenuta seguendo i principi dello *User Centered Design*: l'attenzione è stata posta sull'utente finale dell'applicazione web che è rappresentato dall'admin dell'azienda ATP che ha un'elevata conoscenza del dominio, e dunque un alto grado di interpretabilità e di comprensione dei dati.

L'esigenza principale è stata quella di rendere piacevole la navigazione dei contenuti strutturandoli in maniera efficace per rendere le informazioni chiare e fruibili.
Per tale motivo si è scelto di organizzare la struttura dell'applicazione utilizzando una sidebar di supporto per accedere alle 3 principali macro-aree:

- Dashboard contenente statistiche generali
- Analisi delle linee
- Analisi delle vetture

In questo modo, la navigazione risulta fluida ed immediata, premettendo di dare risalto ai contenuti principali.

3.4 Realizzazione dei prototipi

La fase di progettazione dell'interfaccia grafica dell'applicazione web ha previsto la realizzazione dei mockup tramite il tool Figma. che forniscono indicazioni a scopo illustrativo ed espositivo sulla struttura grafica delle pagine web da implementare. Questi hanno rappresentato il punto di partenza per la successiva fase di implementazione, e hanno permesso di esaminare il prodotto completo prima che sia stato effettivamente realizzato.

3.4.1 Figma

Figma è un tool per la progettazione di interfacce grafiche, principalmente basato sul Web con funzionalità offline aggiuntive abilitate dalle applicazioni desktop.
I principali vantaggi che offre tale strumento sono:

- Customizzare le dimensioni dell'ambiente di lavoro per scegliere quelle più adatta alle proprie esigenze;
- Esportare il codice in CSS, XML o Swift;
- Realizzare animazioni per ottenere un vero prototipo;
- Modificare immagini vettoriali;
- Sistema di versioning.

3.4.2 Prototipi

Di seguito sono illustrati i prototipi realizzati.

Figura 3.5: Prototipo Sidebar

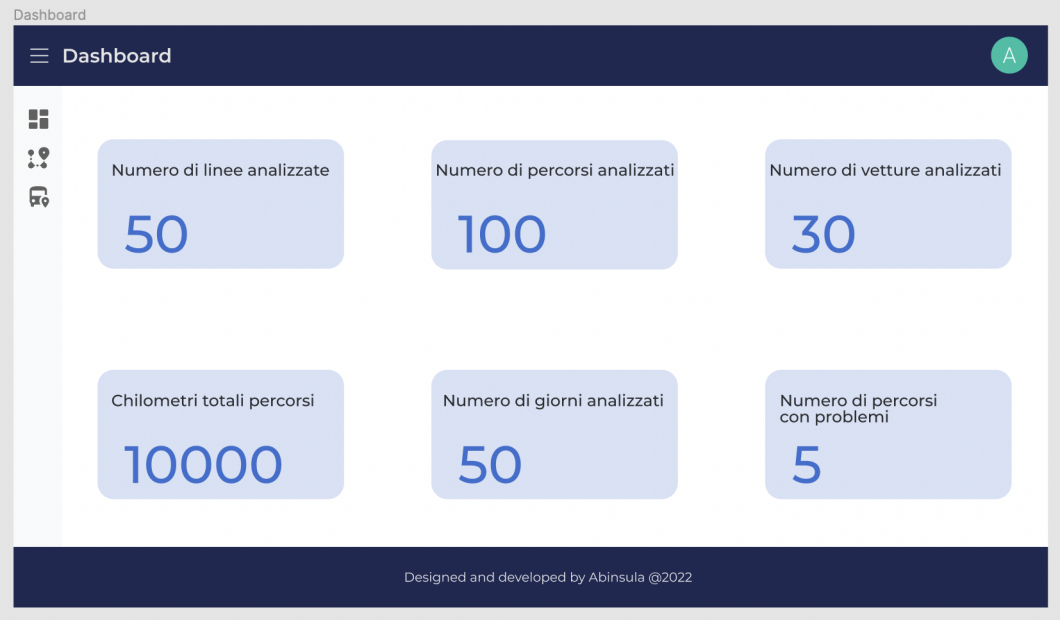


Figura 3.6: Prototipo Dashboard

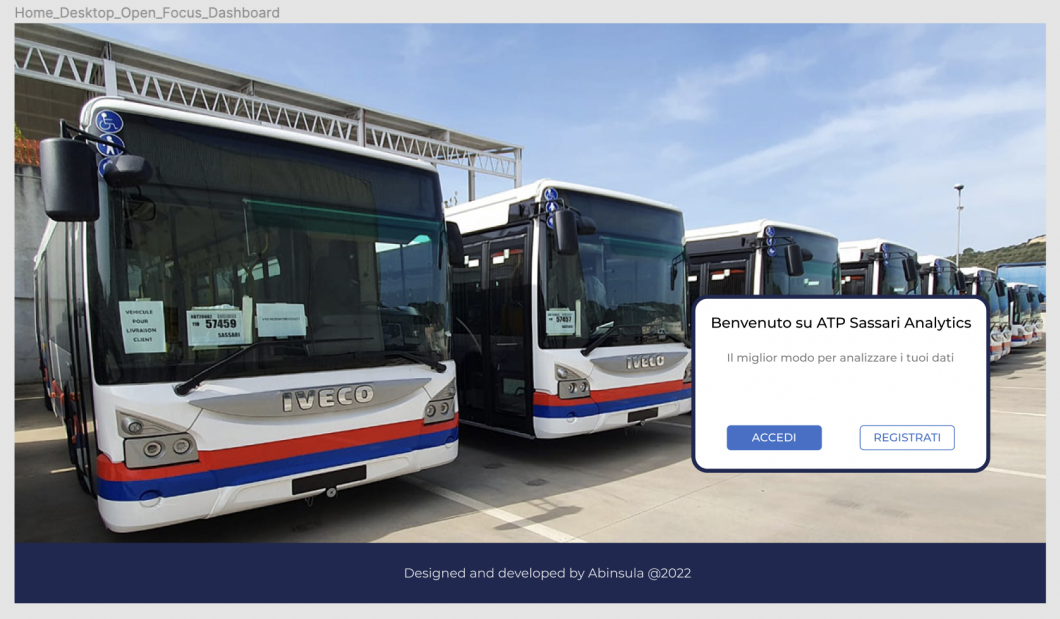


Figura 3.7: Prototipo Homepage

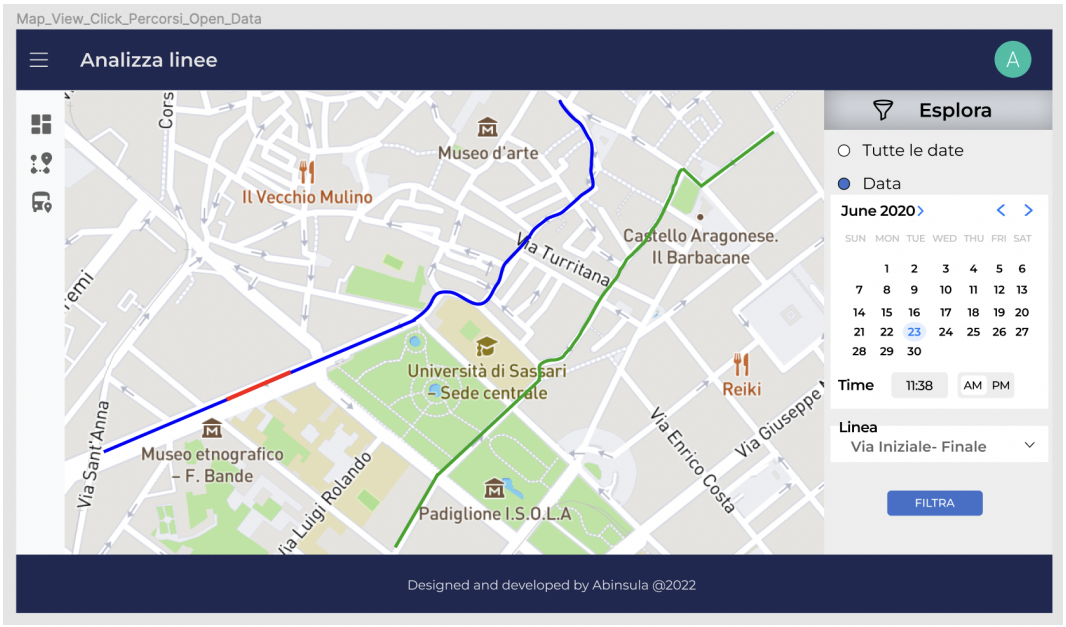


Figura 3.8: Prototipo schermata di analisi delle linee

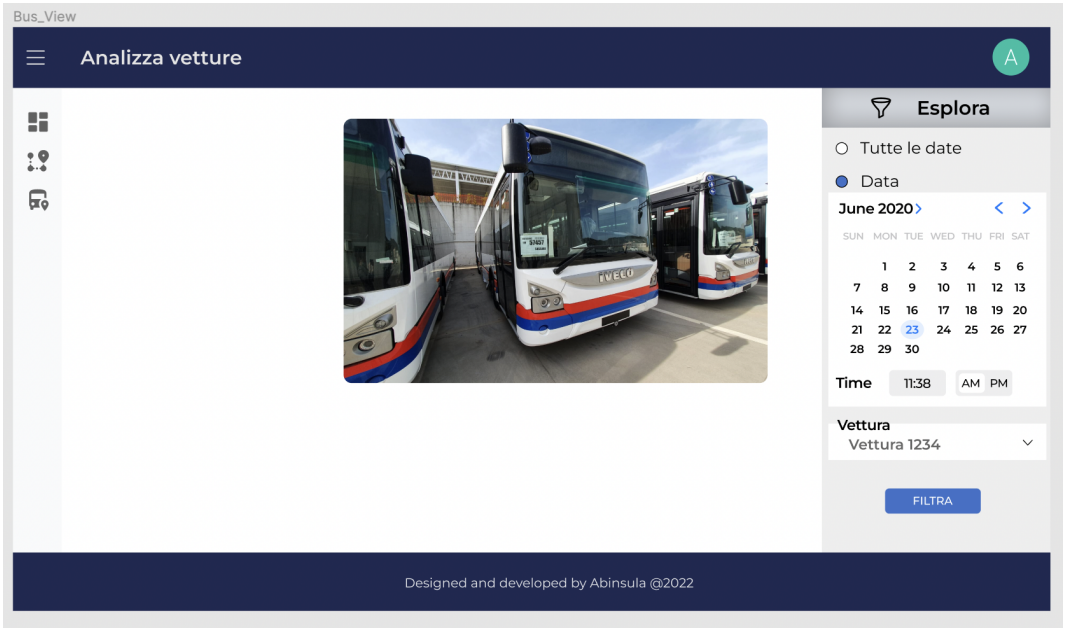


Figura 3.9: Prototipo schermata di analisi dei veicoli

3.5 Studio sulla struttura DBMS

Come accennato precedentemente, MySQL è il DBMS utilizzato per realizzare ATP Sassari analytics. Le informazioni raccolte per mezzo dei sensori e quelle provenienti dal GTFS dell'ATP di Sassari, sono state opportunamente memorizzate in diverse tabelle del database, al fine di garantirne la persistenza e renderle facilmente fruibili all'utente finale.
In particolare, le tabelle costruite sono le seguenti:

- *trips*, che contiene le informazioni di base relative ad ogni linea (identificativo, partenza-capolinea, identificativo del percorso);
- *shapes*, che contiene le coordinate di latitudine e longitudine di ogni percorso;
- *stops*, che contiene le informazioni relative alle fermate effettuate dalle linee;
- *sensor_data*, che contiene i dati provenienti dai sensori posti sugli autobus, citati precedentemente.

Capitolo 4

Implementazione

4.1 Caratteristiche generali

Il progetto di tesi è stato implementato avvalendosi delle potenzialità offerte dall'editor di codice *Visual Studio Code*, che include il supporto per debugging e un controllo per Git integrato. Il versioning del codice è stato realizzato tramite *GitLab*, piattaforma utilizzata dall'azienda Abinsula srl.
Nella fase inziale di sviluppo è stato installato *Node.js* ed è stata inzializzata l'applicazione web lato frontend in *React.js* per mezzo dell'istruzione `npx create-react-app`, dove npx rappresenta un esecutore di pacchetti incluso in npm.
L'interazione con il server per recuperare risorse in modo asincrono è avvenuta utilizzando l'*API Fetch* di Javascript, basata sulle promises.
Una tipica richiesta GET assume la seguente forma:

```
1 // richiesta GET.
2 fetch(url)
3 // gestione successo
4 .then(response => response.json()) // conversione in json
5 .then(json => console.log(json)) // stampa dei dati sulla console
6 .catch(err => console.log('Request Failed', err)); // gestione
     errori
```

Durante la fase di sviluppo, l'applicazione React è stata eseguita sulla porta 3000, mentre il server in ascolto sulla porta 3001. Il collegamento tra l'elaborazione frontend dell'applicazione e la parte backend è avvenuto attraverso la configurazione di un proxy, aggiungendo la seguente riga al *package.json*:

```
1 "proxy": "http://localhost:3001"}
```

4.2 Autenticazione

Al fine di poter utilizzare tutte le funzionalità presenti nell'applicazione, è necessario un processo di autenticazione degli amministratori dell'ATP di Sassari. La piattaforma di autenticazione utilizzata per ATP Analytics è *Auth0*. Essa fornisce un'interfaccia grafica di login universale, che è comunque customizzabile. L'interfaccia di Login, rappresentata nella seguente figura, permette l'inserimento dei campi email address e password.

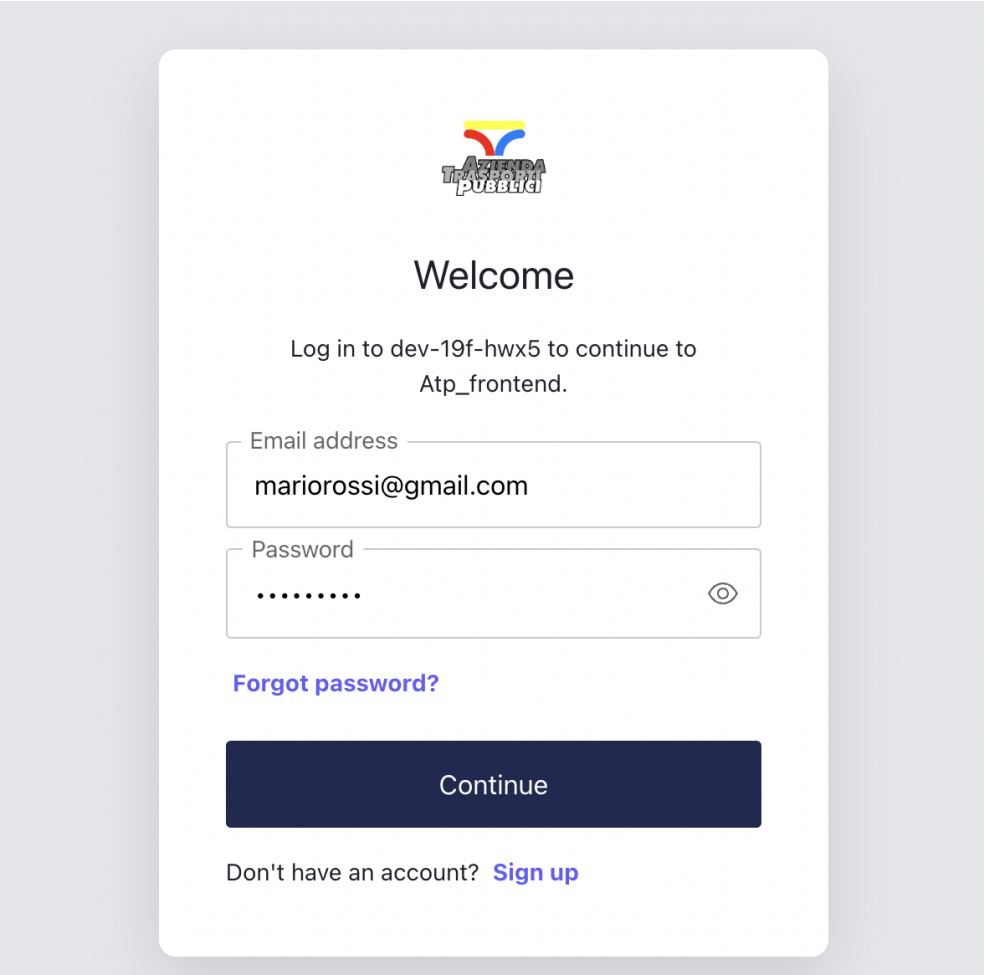


Figura 4.1: Form di login

Una volta completata la procedura di autenticazione, il contesto dell'utente contenente tutte le informazioni ad esso come il nome e l'email, viene salvato nell'applicazione per durata dell'intera sessione.

Nel contesto dell'autenticazione dell'utente, è stata anche sviluppata una schermata di registrazione, mostrata di seguito:

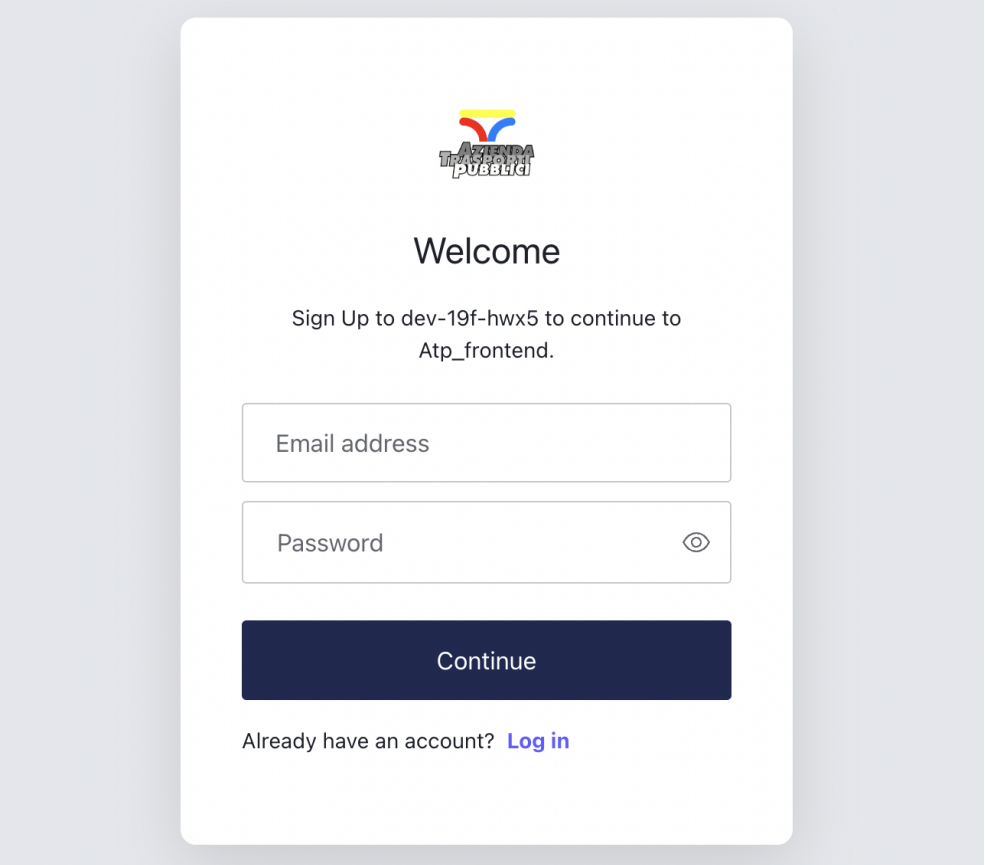


Figura 4.2: Form di registrazione

4.3 Routing

Come accennato in precedenza, una Single Page Application è in grado di eseguire l'intera applicazione nel contesto di una singola pagina. Per simulare una navigazione multi-pagina è necessario implementare un meccanismo di routing, così da guidare l'utente durante la navigazione dell'applicazione. Tale meccanismo è stato realizzato per mezzo della libreria *React Router*[13] che si basa su 3 componenti fondamentali:

- **Router**, posto alla radice della gerarchia di componenti, ha il compito di creare un oggetto *history* per tener traccia dell'URL corrente. Quando questo cambia per mezzo della navigazione dell'utente, il componente figlio viene re-renderizzato.

- **Routes**, è il componente dentro al quale vengono collocate le differenti *Route*, dalla più specifica a quella più generica.
- **Route** è il componente mostrato in modo condizionale che esegue il rendering di un'interfaccia utente quando il suo percorso corrisponde all'URL corrente. Ciascun elemento di tipo Route ha le seguenti props:
  - **path**, rappresenta il pathname assegnato ad un componente.
  - **exact**, usata per indicare che il componente verrà renderizzato solo se il path corrisponde esattamente all'URL.
  - **element**, rappresenta il componente da renderizzare per la specifica route.

React Router fornisce inoltre alcuni hooks che consentono di accedere allo stato del router ed eseguire la navigazione dall'interno dei componenti:

- **useHistory()**, permette di accedere all'istanza di *history*;
- **useLocation()**, rappresenta l'URL corrente;
- **useParams()**, restituisce un oggetto di coppie chiave/valore di parametri dell'URL;
- **useRouteMatch(expression)**, tenta di abbinare l'URL corrente allo stesso modo di un <Route>.

In ATP Sassari analytics, il meccanismo di routing è stato implementato nel seguente modo:

```
1  <Router>
2      <Routes>
3          {isAuthenticated &&
4          <Route element={<RoutedLayout/>}>
5              <Route exact path="/dashboard" element={ <
       DashboardPage/> }/>
6              <Route exact path="/linee" element={ <PathPage/> }/>
7              <Route exact path="/vetture" element={<BusPage/> }/>
8              <Route exact path="/account" element={ <AccountPage/
       >}/>
9          </Route>
10         }
11         <Route exact path="/" element={ <Homepage/> }/>
12     </Routes>
13     <BasicSnackbar/>
14     <Footer/>
15 </Router>
```

Nella seguente tabella sono riportate tutte le route disponibili, i componenti corrispondenti e l'indicazione se per accedere è necessario che l'utente sia autenticato.

Componente	Route	Autenticato
Homepage	/	NO
DashboardPage	/dashboard	SI
PathPage	/linee	SI
BusPage	/vetture	SI
AccountPage	/account	SI

Tabella 4.1: Routes e componenti di ATP Sassari Analytics

4.4 Documentazione delle APIs

Di seguito viene fornita la lista di tutti gli endpoints raggiungibili dall'amministratore di ATP Sassari Analytics:

- GET */paths/trips/:tripId/:startDate*, restituisce la lista di tutti i percorsi effettuati da una linea in un preciso giorno.
- GET */paths/trips/:tripId*, restituisce la lista di tutti i percorsi effettuati da una linea.
- GET */paths/infos/trips/:tripId/:startDate*, restituisce le informazioni più specifiche dei percorsi effettuati da una linea in un preciso giorno.
- GET */paths/infos/trips/:tripId*, restituisce le informazioni più specifiche dei percorsi effettuati da una linea.
- GET */paths*, restituisce il numero totale di percorsi analizzati.
- GET */distances/trips/:tripId/:startDate*, restituisce la distanza media percorsa da una linea in una specifica data.
- GET */distances/trips/:tripId*, restituisce la distanza media percorsa da una linea.
- GET */distances/buses/:busId/:startDate*, restituisce la distanza media percorsa da una vettura in una specifica data.
- GET */distances/buses/:busId*, restituisce la distanza media percorsa da una vettura.

- PUT */buses/:id/:tripId/:startDate/:endDate*, permette di settare il percorso compiuto da una vettura.
- GET */buses/:busId/:startDate*, restituisce le informazioni relative ad una vettura in una specifica data.
- GET */buses/:busId*, restituisce le informazioni relative ad una vettura.
- GET */buses/names*, restituisce la lista contentente gli identificativi di tutte le vetture.
- GET */distances*, restituisce i chilometri totali percorsi.
- GET */shapes/:shapeId*, restituisce il percorso di una linea.
- GET */shapes/stopTimes/:shapeId*, restituisce le fermate effettuate da una linea.
- GET */trips/names*, restituisce la lista contentente tutte le linee.
- GET */trips*, restituisce il numero totale di linee analizzate
- GET */buses*, restituisce il numero totale di vetture analizzate
- GET */days*, restituisce il numero totale di giorni analizzate
- GET */problems*, restituisce il numero totale di percorsi analizzati con problemi.
- GET */speed/trips/:tripId/:startDate*, restituisce la velocità media registrata per una linea in una specifica data.
- GET */speed/trips/:tripId*, restituisce la velocità media registrata per una linea.

4.5 Connessione al database MySQL

Come accennato nel capitolo precedente, il modulo node *mysql2* è stato utilizzato al fine di accedere al database MySQL con Node.js, nel seguente modo:

```
1 const mysql = require("mysql2");
```

Tramite questo modulo è stata aperta una connessione e creata la funzione *execute* al fine di eseguire le queries al database.

```
let connection;
/**
 * generates connection to be used throughout the app
 */
const init = () => {
    try {
        connection = mysql.createConnection({
            host: dataSource.DB_HOST,
            user: dataSource.DB_USER,
            password: dataSource.DB_PASSWORD,
            database: dataSource.DB_DATABASE,
        });
        console.log('MySql Connection generated successfully');
    }
    catch (error) {
        console.error('[mysql.connector][init][Error]: ', error);
        throw new Error('failed to initialized connection');
    }
};

/**
 * executes SQL queries in MySQL db
 *
 * @param {string} query - provide a valid SQL query
 * @param {string[] | Object} params - provide the parameterized
   values used in the query
 */
const execute = function (query, params) {
    try {
        if (!connection)
            throw new Error('Connection was not created. Ensure
   connection is created when running the app.');
        return new Promise( (resolve, reject) => {
            connection.query(query, params, (error, results) => {
                if (error)
                    reject(error);
                else
                    resolve(results);
            });
        });
    }
    catch (error) {
        console.error('[mysql.connector][execute][Error]:',error);
        throw new Error('failed to execute MySQL query');
    }
};
```

4.6 Gestione degli errori

Le gestione degli errori rappresenta uno dei principi più importanti dell'usabilità di un'applicazione web. Nell'ambito del seguente lavoro di tesi si è cercato di fornire un feedback all'utente per le principali azione eseguite.

In particolare, durante la compilazione del form di login e dello strumento per filtrare i dati, in caso di errore, viene segnalato esattamente il campo interessato e indicata la motivazione accanto all'elemento.

Inoltre, in caso di errore nel caricamento dei dati dal server, verrà visualizzata una snackbar customizzata al fine di indicare con precisione il motivo per cui non è stato possibile caricare i dati.

Nel caso in cui viene digitato un URL al quale non corrisponde alcuna pagina, verrà mostrata una schermata di errore.

Per dare un riscontro nell'attesa del caricamento dei dati, è stato utilizzato un elemento di tipo *CircularProgress* che indica l'avanzamento dell'operazione.

4.7 Librerie di terze parti

Durante lo sviluppo di ATP Sassari Analytics sono state incluse nel progetto le seguenti librerie Javascript:

- **recharts**: per la generazione di grafici a linea e a barre.
- **react-leaflet**: per mostrare la mapppa Leaflet e compiere operazioni su di essa.
- **dayjs**: per la gestione del formato delle date
- **react-datetime-picker**: per la generazione di un calendario user friendly.
- **@mui/material**: per la gestione dell'intera interfaccia grafica.
- **@mui/icons-material**: per la gestione delle icone.
- **react-countup**: per la generazione di un contatore animato.

4.8 Implementazione delle funzionalità

Di seguito vengono illustrate le funzionalità implementate in ATP Sassari Analytics attraverso la visualizzazione delle diverse schermate.

4.8.1 Homepage

La schermata iniziale che compare all'apertura dell'applicazione è l'homepage, di seguito mostrata.

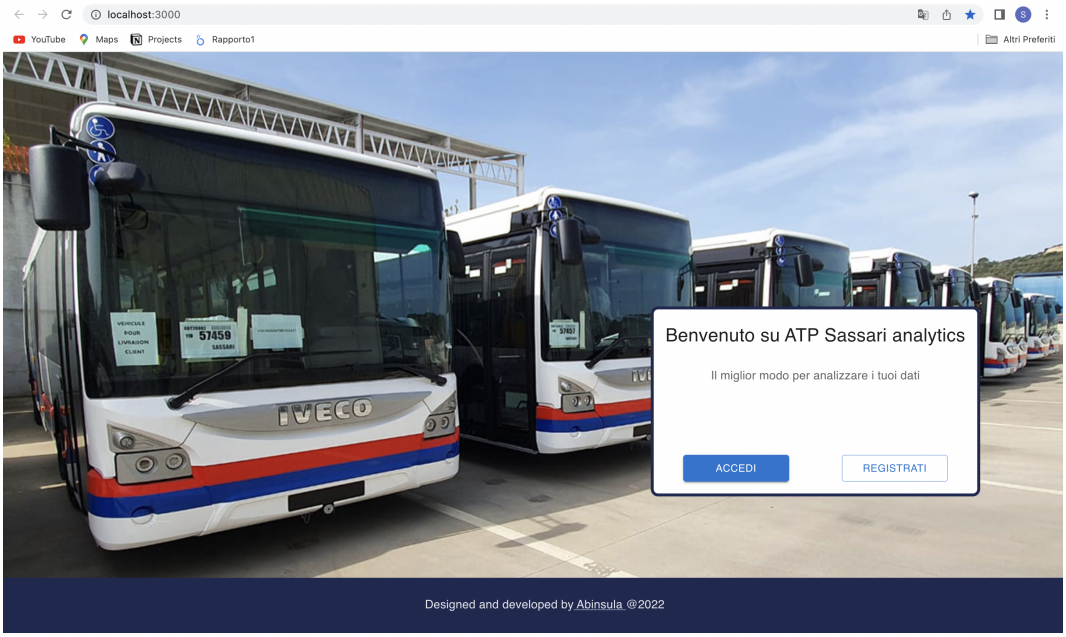


Figura 4.3: Homepage di ATP Sassari Analytics

La struttura di questo componente è molto minimale: questo permette di mettere a fuoco immediatamente la principale azione da compiere, ossia il login. Si è pensato di porre in background un'immagine della flotta di veicoli dell'ATP Sassari per mettere subito in evidenza lo scopo dell'applicazione.
A questo si è aggiunto un semplice *Footer* con l'indicazione dell'azienda produttrice e dell'anno di produzione.
Cliccando sul bottono *Accedi* o *Regisrati*, compaiono le due schermate illustrate precedentemente che danno la possibilità rispettivamente di accedere e registrarsi alla piattaforma.

4.8.2 Dashboard

Una volta effettuato il login, l'amministratore verrà reindirizzato alla dashboard dell'applicazione di seguito illustrata:

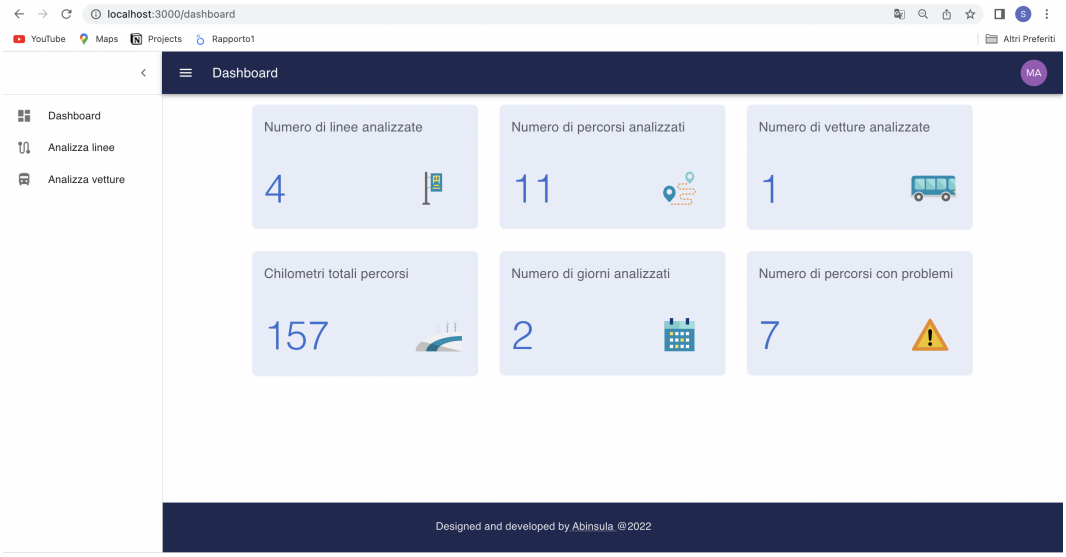


Figura 4.4: Dashboard di ATP Sassari Analytics

La dashboard mostra all'utente statistiche di carattere generale: si tratta di informazioni flash, rappresentate da un *Box* contenente un valore numerico e un'icona.
Inoltre, a sinistra è mostrata la *Sidebar* espansa con i 3 possibili percorsi:

- Dashboard corrente
- Schermata di analisi delle linee
- Schermata di analisi dei veicoli

4.8.3 Schermata analisi delle linee

Spostandosi in *Analizza linee* per mezzo della Sidebar, si viene indirizzati alla schermata di analisi dei percorsi.
In questa pagina è presente una mappa realizzata con la libreria *React Leafleat* che permette di spostarsi nell'area di Sassari e un *Filter tool* per filtrare la linea desiderata, ed eventualmente una data. In questo modo è possibile visionare la tratta e i percorsi che sono stati effettuati. Il risultato è il seguente:

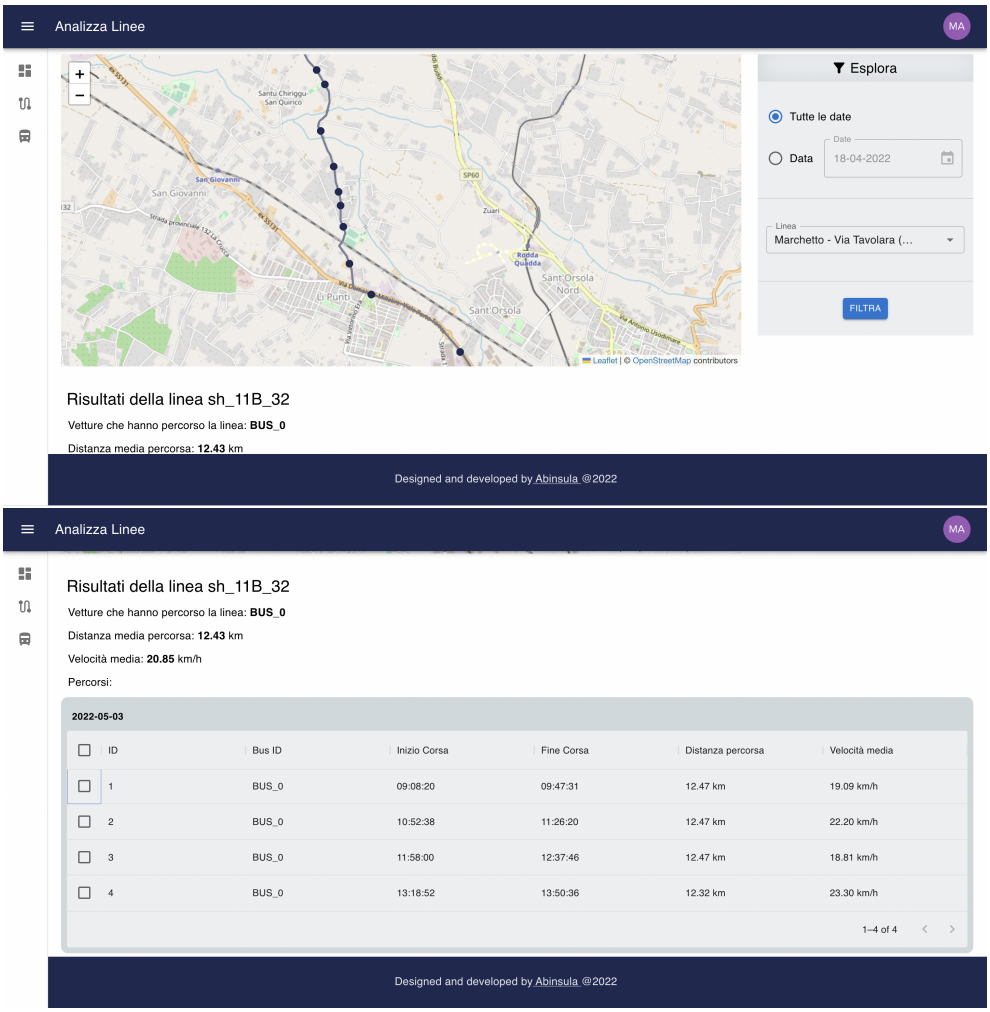


Figura 4.5: Schermata dei risultati della linea SH_11B_32

All'interno della mappa è tracciato il percorso della linea selezionata con le relative fermate. Spostandosi con il cursore sopra una fermata, è possibile vederne i dettagli.
La tabella, invece, mostra i percorsi relativi alla linea selezionata.

Selezionando uno specifico percorso dalla tabella, si ottengono le statiche ad esso correlate:

Figura 4.6: Schermata delle statistiche del percordo con ID:1

I risultati sono rappresentati attraverso due grafici:

- un grafico a linee in cui è rappresentata la variazione di velocità nel tempo.
- un grafico a barre che rappresenta la permanenza (in secondi) per ogni fermata.

Inoltre, anche in questo caso, è possibile visionare sulla mappa il percorso selezionate, in modo da valutare immediatamente se vi sono state deviazioni rispetto al percorso standard.
Infine, è possibile comparare due percorsi, selezionandoli dalla tabella

4.8.4 Schermata analisi dei veicoli

Spostandosi in *Analizza vetture* per mezzo della Sidebar, si viene indirizzati alla schermata di analisi delle vetture. Tale schermata riprende per continuità lo stile della schermata di analisi delle linee. Infatti, anche in questo caso è presente un filter tool, che permette di scegliere uno specifico veicolo, selezionando eventualmente anche una data. In questo modo si ottengono i seguenti risultati:

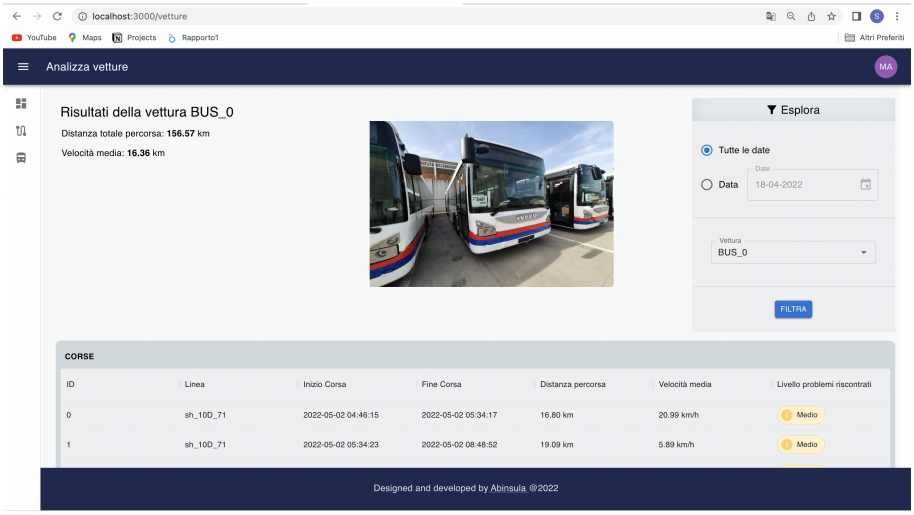


Figura 4.7: Schermata dei risultati del veicolo BUS_0

Nella tabella vengono riportati i percorsi effettuate dall'autobus, con indicazione della linea percorsa e del livello dei problemi riscontrati che varia tra basso, medio e alto. Cliccando sopra il *Chip* è possibile vederne i dettagli:

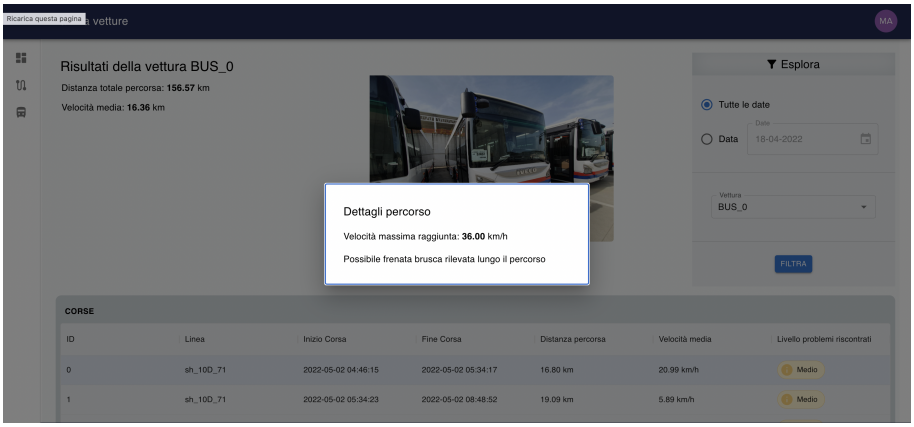


Figura 4.8: Dettaglio problemi riscontrati nel percorso selezionato

Capitolo 5

Conclusioni

5.1 Sviluppi futuri

Bibliografia

[1] LITSLINK. *Modern Web Application Architecture Explained: Components, Best Practices and More*. https://litslink.com/blog/web-application-architecture. 2022 (cit. a p. 3).

[2] Vito La Vecchia. *Caratteristiche e vantaggi delle applicazioni web based (Web Application)*. https://litslink.com/blog/web-application-architecture. 2022 (cit. a p. 3).

[3] Statista. *Most used web frameworks among developers worldwide, as of 2022*. https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/ (cit. a p. 7).

[4] Fondazione OpenJS. *Node.js*. https://nodejs.org/en/docs/ (cit. a p. 8).

[5] Fondazione OpenJS. *Express.js*. https://expressjs.com/it/guide/writing-middleware.html (cit. a p. 10).

[6] Meta Platforms. *Virtual DOM and Internals*. https://reactjs.org/docs/faq-internals.html (cit. a p. 12).

[7] Meta Platforms. *Componenti e props*. https://it.reactjs.org/docs/components-and-props.html (cit. a p. 12).

[8] Meta Platforms. *Introduzione a JSX*. https://it.reactjs.org/docs/introducing-jsx.html (cit. a p. 12).

[9] Meta Platforms. *Usare l'Hook Effect*. https://it.reactjs.org/docs/hooks-effect.html (cit. a p. 14).

[10] Oracle. *MySQL 8.0 Reference Manual*. https://dev.mysql.com/doc/refman/8.0/en/introduction.html (cit. a p. 14).

[11] Statista. *Ranking of the most popular database management systems worldwide, as of August 2022*. https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/ (cit. a p. 15).

[12] databricks. *Understanding Dynamic Time Warping*. `https://www.databricks.com/blog/2019/04/30/understanding-dynamic-time-warping.html` (cit. a p. 22).

[13] Inc. Remix Software. *React Router*. `https://reactrouter.com/en/main/start/overview` (cit. a p. 31).