OraOra Rail: Visualizzatore Interattivo di Orari Ferroviari

Intelligent Agent - Automated Planning module

Ilaria Frandina
Matricola: 264232

Luglio 2025

1 Introduzione

Questo report presenta lo sviluppo di **OraOra Rail**, un'applicazione web interattiva progettata per visualizzare gli orari ferroviari utilizzando dati GTFS (General Transit Feed Specification) forniti da Trenitalia. Il progetto si concentra sulla trasformazione di dati statici degli orari della rete ferroviaria sarda in un sistema di visualizzazione dinamico e user-friendly.

L'obiettivo principale era creare un'interfaccia intuitiva che permettesse agli utenti di esplorare i percorsi dei treni, gli orari e le simulazioni in tempo reale attraverso mappe interattive e animazioni del movimento dei treni.

2 Architettura del Sistema e Stack Tecnologico

OraOra Rail è costruito su un'architettura a due livelli composta da una pipeline di preprocessing dei dati basata su Python e un'applicazione web client-side. Il componente di preprocessing trasforma i dati GTFS grezzi in formati JSON ottimizzati per il consumo web, mentre il frontend fornisce un'interfaccia di visualizzazione interattiva.

La base tecnologica si basa su Python 3.7+ con la libreria Pandas per la manipolazione dei dati, garantendo una gestione robusta di grandi dataset e vari formati di codifica comunemente presenti nei dati di trasporto europei. Il frontend web utilizza JavaScript moderno (ES6+) con Leaflet.js per le capacità di mappatura interattiva e vis-timeline per la visualizzazione temporale degli orari dei treni.

Questa separazione delle responsabilità consente un'elaborazione efficiente dei dati mantenendo al contempo interazioni utente responsive. La fase di preprocessing gestisce la complessità computazionale della trasformazione dei dati GTFS, mentre l'interfaccia web si concentra sul fornire animazioni fluide e controlli utente intuitivi.

3 Elaborazione e Ottimizzazione dei Dati GTFS

La pipeline di preprocessing, implementata in `preprocess_gtfs.py`, gestisce il compito complesso di convertire i feed GTFS grezzi in un formato ottimizzato per la visualizzazione web. Lo script elabora sei file GTFS principali: routes, trips, stops, shapes, stop times e calendar dates, ciascuno con uno scopo specifico nell'ecosistema dei dati ferroviari.

```
def main():
    # Carica e valida i file GTFS con rilevamento encoding
    routes_df = safe_read_csv(in_dir / 'routes.txt')
    trips_df = safe_read_csv(in_dir / 'trips.txt')
    stops_df = safe_read_csv(in_dir / 'stops.txt')
```

```
    # Filtra solo i servizi ferroviari (route_type == 2)
    train_routes = routes_df[routes_df['route_type'] == 2]

    # Elabora e ottimizza le strutture dati
    routes_json = process_routes(train_routes)
    shapes_json = process_shapes_with_generation(train_shapes, stops_df)

    # Genera output JSON ottimizzati
    save_optimized_json(routes_json, 'routes.json')
```

Listing 1: Flusso di preprocessing principale

Un'innovazione chiave nella fase di preprocessing è la generazione automatica di forme geografiche per le rotte prive di dati espliciti di forma. Quando le informazioni sulla forma mancano, il sistema interpola le coordinate tra le fermate delle stazioni, assicurando che tutte le rotte possano essere visualizzate sulla mappa. Questo approccio migliora significativamente la completezza dei dati mantenendo l'accuratezza geografica.

Il processo di ottimizzazione include anche la gestione intelligente dei calendari di servizio, convertendo intervalli di date complessi ed eccezioni in strutture facilmente consultabili. Questa fase di preprocessing riduce drasticamente il carico computazionale lato client e consente il filtraggio in tempo reale delle date di viaggio disponibili.

4 Progettazione e Implementazione dell'Interfaccia Utente

L'interfaccia utente segue i principi contemporanei del web design con un'estetica pulita e professionale che privilegia funzionalità e accessibilità.

L'interfaccia è strutturata attorno a un modello di divulgazione progressiva, dove gli utenti navigano attraverso una sequenza logica di selezioni: linea ferroviaria, stazione di origine, stazione di destinazione, data di viaggio e orario di partenza specifico. Questo approccio gerarchico riduce il carico cognitivo garantendo al contempo che gli utenti possano trovare efficientemente le informazioni rilevanti.

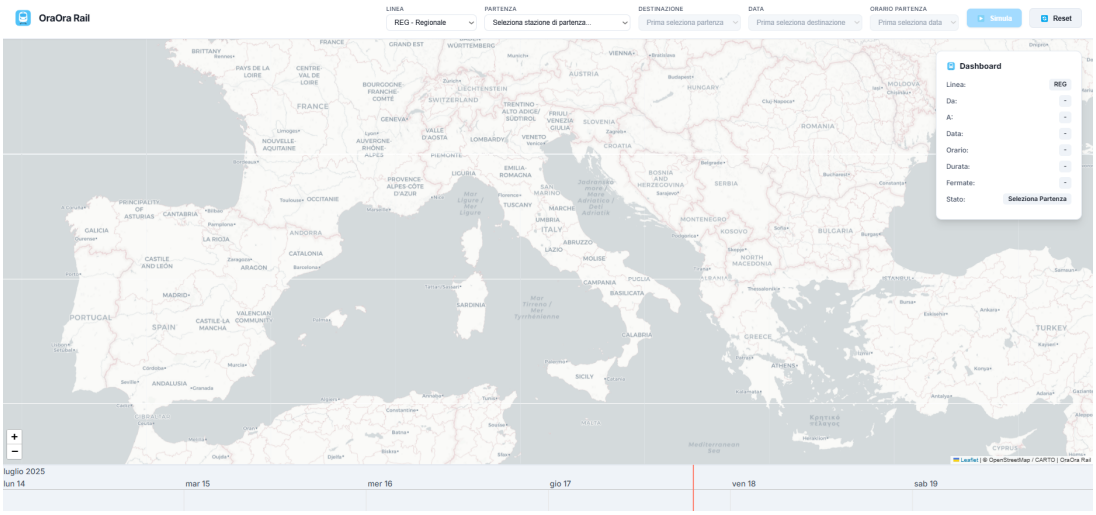

Figura 1: Interfaccia principale che mostra i controlli di selezione del percorso, la mappa interattiva e la visualizzazione della timeline

Il design responsive si adatta senza problemi a diverse dimensioni dello schermo, con la versione desktop che presenta un pannello dashboard completo che visualizza informazioni in tempo reale sul viaggio selezionato. Il componente mappa, alimentato da Leaflet.js, fornisce interazioni fluide di panoramica e zoom mantenendo le prestazioni con grandi dataset.

5 Funzionalità Principali e Caratteristiche

La funzionalità principale dell'applicazione ruota attorno a tre modalità di visualizzazione principali: visualizzazione statica del percorso, navigazione interattiva della timeline e animazione dinamica del treno. Ogni modalità serve diverse esigenze degli utenti mantenendo la coerenza nell'esperienza utente complessiva.

La visualizzazione del percorso presenta le linee ferroviarie come polilinee colorate sovrapposte su mappe geografiche, con le stazioni contrassegnate come marker circolari interattivi. Il sistema distingue tra stazioni di origine e destinazione attraverso la codifica dei colori e le variazioni di dimensione, fornendo feedback visivo immediato sui parametri di viaggio selezionati.

Il componente timeline offre una vista temporale degli orari dei treni, permettendo agli utenti di comprendere i pattern di frequenza e identificare gli orari di partenza ottimali. L'integrazione tra mappa e timeline assicura che selezionando un viaggio specifico si aggiornino immediatamente sia le visualizzazioni geografiche che temporali.

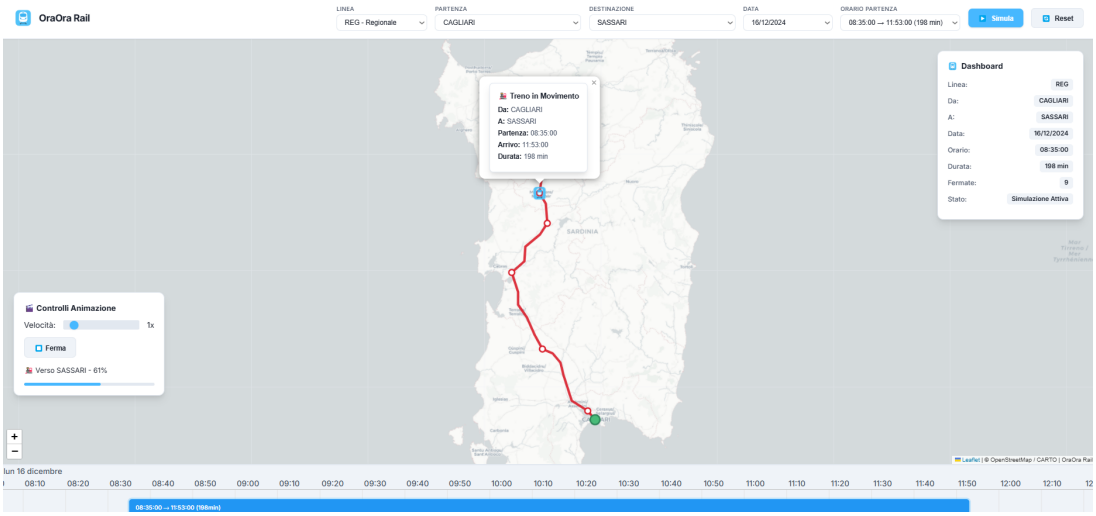

Figura 2: Animazione del treno in corso che mostra il marker del treno in movimento, l'indicatore di progresso e i controlli di animazione

Il sistema di animazione rappresenta la caratteristica più sofisticata, fornendo movimento realistico del treno lungo i percorsi geografici. Il motore di animazione interpola il movimento tra i punti di coordinate mantenendo frame rate fluidi e fornendo controlli utente per la regolazione della velocità. Gli indicatori di progresso e gli aggiornamenti in tempo reale assicurano che gli utenti rimangano informati sullo stato del viaggio durante tutta la simulazione.

6 Dettagli di Implementazione Tecnica

Il codebase JavaScript è organizzato in moduli specializzati, ciascuno responsabile di aspetti distinti della funzionalità dell'applicazione. Il modulo `DataManager` gestisce tutte le operazioni di accesso ai dati e caching, fornendo un'interfaccia unificata per il recupero e le operazioni di filtraggio dei dati GTFS.

La gestione della mappa è incapsulata nel modulo `MapManager`, che astrae la complessità di Leaflet.js fornendo funzionalità specifiche dell'applicazione come la creazione di marker personalizzati e la gestione dei layer. L'architettura modulare garantisce la manutenibilità e consente il testing indipendente dei singoli componenti.

```
const AnimationManager = {
    currentAnimation: null,
    animationSpeed: 5,
```

```
    isAnimating: false,

    animateTrip(tripId) {
        const trip = DataManager.getTrip(tripId);
        const shapeCoords = DataManager.getShape(trip.shape_id);

        this.trainMarker = MapManager.addTrainMarker(shapeCoords[0]);
        this.animateAlongPath(shapeCoords, trip);
    }
};
```

Listing 2: Gestione dello stato di animazione

L'ottimizzazione delle prestazioni si concentra su strutture dati efficienti. L'applicazione impiega il caricamento lazy per le forme geografiche e implementa meccanismi di caching intelligenti per ridurre le chiamate API ridondanti. Le prestazioni di animazione sono mantenute attraverso la programmazione requestAnimationFrame e transizioni CSS accelerate.

7 Risultati e Valutazione

L'applicazione OraOra Rail completata trasforma con successo i dati GTFS statici in una piattaforma di visualizzazione interattiva e coinvolgente. I test con i dataset Trenitalia forniti hanno dimostrato la capacità del sistema di gestire la complessità dei dati di trasporto del mondo reale mantenendo prestazioni responsive.

Il sistema di animazione fornisce movimenti dei treni fluidi e realistici che comunicano efficacemente la durata del viaggio e la complessità del percorso.

La pipeline di preprocessing si è dimostrata robusta attraverso vari formati di dati GTFS, gestendo con successo le variazioni di codifica e i dati di forma mancanti attraverso algoritmi di interpolazione intelligenti. Le prestazioni di elaborazione scalano linearmente con la dimensione del dataset, rendendo il sistema adatto anche a processare dataset più corposi.

8 Conclusioni e Sviluppi Futuri

OraOra Rail raggiunge con successo il suo obiettivo primario di creare una piattaforma di visualizzazione accessibile e interattiva per i dati degli orari ferroviari. L'applicazione dimostra come le tecnologie web moderne possano trasformare informazioni complesse sui trasporti in esperienze utente intuitive.

L'architettura modulare e la pipeline completa di elaborazione dati forniscono una base solida per futuri miglioramenti. Gli sviluppi potenziali includono l'integrazione con sistemi di tracciamento in tempo reale, capacità di analisi avanzate per l'analisi dei ritardi e supporto esteso per reti di trasporto multi-modali.

8.1 Integrazione di Tecniche di Planning avanzate

Il sistema attuale può essere significativamente potenziato attraverso l'integrazione di algoritmi di pianificazione automatica e tecniche di ottimizzazione che estenderebbero le funzionalità di visualizzazione verso un vero e proprio sistema di supporto decisionale per i trasporti.

Pianificazione di Itinerari Ottimali: L'implementazione di algoritmi di pathfinding avanzati come A* o Dijkstra permetterebbe al sistema di calcolare automaticamente percorsi multi-tratta ottimali considerando tempo di viaggio, numero di cambi e costi. Questa evoluzione naturale della visualizzazione esistente consentirebbe agli utenti non solo di visualizzare i percorsi esistenti, ma di ricevere suggerimenti intelligenti per i loro viaggi, trasformando il sistema da strumento di visualizzazione passiva in assistente di viaggio attivo.

Schedulazione Intelligente: Lo sviluppo di un sistema di scheduling basato su PDDL (Planning Domain Definition Language) potrebbe ottimizzare automaticamente gli orari dei treni considerando vincoli operativi, capacità delle linee e domanda passeggeri. Il dominio di pianificazione modellerebbe azioni come "muovi$_t$*reno*", "$fermata_s tazione$" e "$cambio_b inario$" $conprecondizionieeffett$

Ripianificazione Dinamica in Tempo Reale: L'implementazione di algoritmi di ripianificazione permetterebbe al sistema di gestire interruzioni del servizio, ritardi e guasti in tempo reale. Attraverso tecniche sofisticate di ripianificazione, il sistema potrebbe automaticamente ricalcolare percorsi alternativi e aggiornare le visualizzazioni istantaneamente, minimizzando l'impatto sui passeggeri e fornendo soluzioni immediate ai problemi operativi mantenendo il feedback visivo che rende efficace il sistema attuale.

Ottimizzazione Multi-Obiettivo: Lo sviluppo di funzioni di costo complesse che bilanciano tempo di viaggio, comfort, prezzo e affidabilità permetterebbe una pianificazione di viaggi personalizzata. L'integrazione di tecniche di ricerca euristica consentirebbe di trovare soluzioni che soddisfano le diverse preferenze degli utenti, che privilegino velocità, convenienza economica o comfort, mantenendo l'interfaccia di visualizzazione intuitiva.

Pianificazione Predittiva: L'utilizzo di tecniche di pianificazione predittiva potrebbe anticipare problemi operativi e suggerire azioni preventive. Il sistema potrebbe analizzare pattern storici per predire ritardi e proporre strategie di mitigazione, trasformando la gestione dei trasporti da reattiva a proattiva sfruttando l'infrastruttura di visualizzazione esistente per mostrare chiaramente predizioni e raccomandazioni.

9 Bibliografia

- GTFS Static Overview - `https://developers.google.com/transit/gtfs`
- Leaflet Documentation - `https://leafletjs.com/`
- vis-timeline Library - `https://visjs.github.io/vis-timeline/`