OraOra Rail: Interactive Railway Timetable Visualizer

Intelligent Agent - Automated Planning module

Ilaria Frandina
Student ID: 264232

July 2025

1 Introduction

This report presents the development of **OraOra Rail**, an interactive web application designed to visualize railway timetables using GTFS (General Transit Feed Specification) data provided by Trenitalia. The project focuses on transforming static timetable data from the Sardinian railway network into a dynamic, user-friendly visualization system.

The primary objective was to create an intuitive interface that allows users to explore train routes, schedules, and real-time simulations through interactive maps and animated train movements.

2 System Architecture and Technology Stack

OraOra Rail is built on a two-tier architecture consisting of a Python-based data preprocessing pipeline and a client-side web application. The preprocessing component transforms raw GTFS data into optimized JSON formats suitable for web consumption, while the frontend provides an interactive visualization interface.

The technological foundation relies on Python 3.7+ with the Pandas library for data manipulation, ensuring robust handling of large datasets and various encoding formats commonly found in European transportation data. The web frontend utilizes modern JavaScript (ES6+) with Leaflet.js for interactive mapping capabilities and vis-timeline for temporal visualization of train schedules.

This separation of concerns allows for efficient data processing while maintaining responsive user interactions. The preprocessing stage handles the computational complexity of GTFS data transformation, while the web interface focuses on providing smooth animations and intuitive user controls.

3 GTFS Data Processing and Optimization

The preprocessing pipeline, implemented in `preprocess_gtfs.py`, handles the complex task of converting raw GTFS feeds into a format optimized for web visualization. The script processes six core GTFS files: routes, trips, stops, shapes, stop times, and calendar dates, each serving a specific purpose in the railway data ecosystem.

```python
def main():
    # Load and validate GTFS files with encoding detection
    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')

    # Filter for railway services only (route_type == 2)
    train_routes = routes_df[routes_df['route_type'] == 2]

    # Process and optimize data structures
    routes_json = process_routes(train_routes)
    shapes_json = process_shapes_with_generation(train_shapes, stops_df)

    # Generate optimized JSON outputs
    save_optimized_json(routes_json, 'routes.json')
```
Listing 1: Core preprocessing workflow

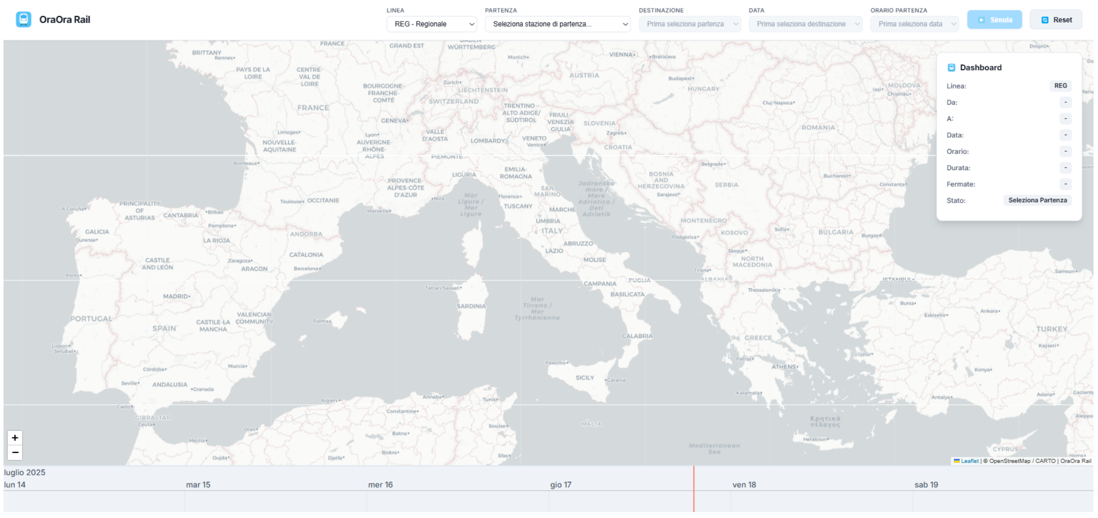
A key innovation in the preprocessing stage is the automatic generation of geographical shapes for routes lacking explicit shape data. When shape information is missing, the system interpolates coordinates between station stops, ensuring that all routes can be visualized on the map. This approach significantly improves data completeness while maintaining geographical accuracy.

The optimization process also includes intelligent handling of service calendars, converting complex date ranges and exceptions into lookup-friendly structures. This preprocessing step dramatically reduces client-side computational overhead and enables real-time filtering of available travel dates.

4 User Interface Design and Implementation

The user interface follows contemporary web design principles with a clean, professional aesthetic that prioritizes functionality and accessibility.

The interface is structured around a progressive disclosure model, where users navigate through a logical sequence of selections: railway line, origin station, destination station, travel date, and specific departure time. This hierarchical approach reduces cognitive load while ensuring users can efficiently find relevant information.



LINEA: REG - Regionale
PARTENZA: Seleziona stazione di partenza...
DESTINAZIONE: Prima seleziona partenza
DATA: Prima seleziona destinazione
ORARIO PARTENZA: Prima seleziona data

Dashboard
Linea: REG
Da: -
A: -
Data: -
Orario: -
Durata: -
Fermate: -
Stato: Seleziona Partenza

Figure 1: Main interface showing the route selection controls, interactive map, and timeline visualization

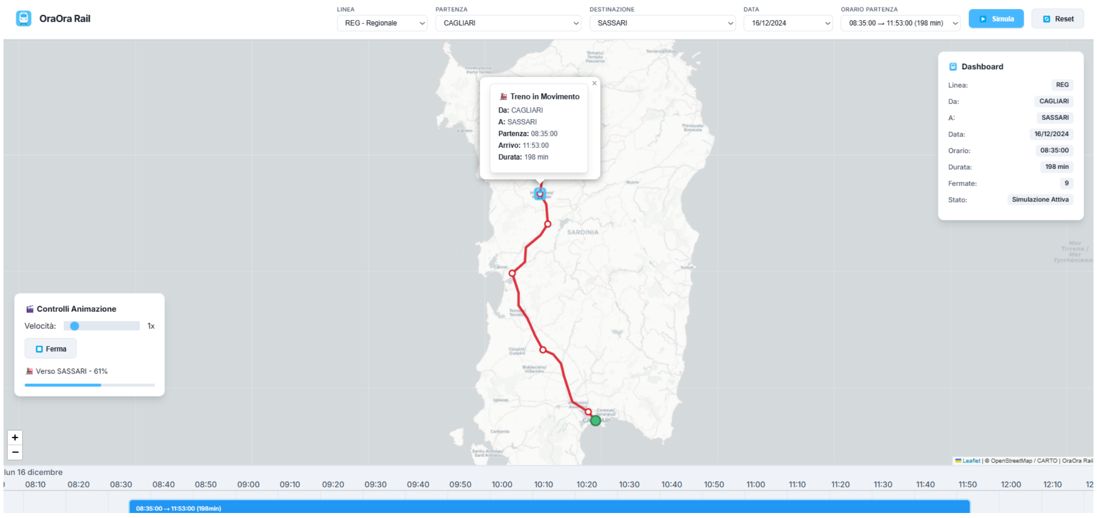
The responsive design adapts seamlessly to different screen sizes, with the desktop version featuring a comprehensive dashboard panel that displays real-time information about the selected journey. The map component, powered by Leaflet.js, provides smooth pan and zoom interactions while maintaining performance with large datasets.

5 Core Functionality and Features

The application's core functionality revolves around three primary visualization modes: static route display, interactive timeline navigation, and dynamic train animation. Each mode serves different user needs while maintaining consistency in the overall user experience.

Route visualization presents railway lines as colored polylines overlaid on geographical maps, with stations marked as interactive circular markers. The system distinguishes between origin and destination stations through color coding and size variations, providing immediate visual feedback about the selected journey parameters.

The timeline component offers a temporal view of train schedules, allowing users to understand frequency patterns and identify optimal departure times. Integration between the map and timeline ensures that selecting a specific trip immediately updates both geographical and temporal visualizations.



LINEA: REG - Regionale
PARTENZA: CAGLIARI
DESTINAZIONE: SASSARI
DATA: 16/12/2024
ORARIO PARTENZA: 08:35:00 → 11:53:00 (198 min)

Treno in Movimento
Da: CAGLIARI
A: SASSARI
Partenza: 08:35:00
Arrivo: 11:53:00
Durata: 198 min

Dashboard

Linea:	REG
Da:	CAGLIARI
A:	SASSARI
Data:	16/12/2024
Orario:	08:35:00
Durata:	198 min
Fermate:	9
Stato:	Simulazione Attiva

Controlli Animazione
Velocità: 1x
Verso SASSARI - 61%

lun 16 dicembre
08:10 08:20 08:30 08:40 08:50 09:00 09:10 09:20 09:30 09:40 09:50 10:00 10:10 10:20 10:30 10:40 10:50 11:00 11:10 11:20 11:30 11:40 11:50 12:00 12:10
08:35:00 → 11:53:00 (198min)

Figure 2: Train animation in progress showing the moving train marker, progress indicator, and animation controls

The animation system represents the most sophisticated feature, providing realistic train movement along geographical routes. The animation engine interpolates movement between coordinate points while maintaining smooth frame rates and providing user controls for speed adjustment. Progress indicators and real-time updates ensure users remain informed about the journey status throughout the simulation.

6 Technical Implementation Details

The JavaScript codebase is organized into specialized modules, each responsible for distinct aspects of the application functionality. The `DataManager` module handles all data access and caching operations, providing a unified interface for GTFS data retrieval and filtering operations.

Map management is encapsulated within the `MapManager` module, which abstracts Leaflet.js complexity while providing application-specific functionality such as custom marker creation and layer management. The modular architecture ensures maintainability and enables independent testing of individual components.

```
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: Animation state management

Performance optimization focuses on efficient data structures. The application employs lazy loading for geographical shapes and implements intelligent caching mechanisms to reduce redundant API calls. Animation performance is maintained through requestAnimationFrame scheduling and hardware-accelerated CSS transitions.

7 Results and Evaluation

The completed OraOra Rail application successfully transforms static GTFS data into an engaging, interactive visualization platform. Testing with the provided Trenitalia datasets demonstrated the system's ability to handle real-world transportation data complexity while maintaining responsive performance.

The animation system provides smooth, realistic train movements that effectively communicate journey duration and route complexity.

The preprocessing pipeline proved robust across various GTFS data formats, successfully handling encoding variations and missing shape data through intelligent interpolation algorithms. Processing performance scales linearly with dataset size, making the system suitable for processing even larger datasets.

8 Conclusions and Future Developments

OraOra Rail successfully achieves its primary objective of creating an accessible, interactive visualization platform for railway timetable data. The application demonstrates how modern web technologies can transform complex transportation information into intuitive user experiences.

The modular architecture and comprehensive data processing pipeline provide a solid foundation for future enhancements. Potential developments include integration with real-time tracking systems, advanced analytics capabilities for delay analysis, and expanded support for multi-modal transportation networks.

8.1 Enhanced Route Planning and Optimization

The current system can be significantly enhanced through the integration of automated planning algorithms and optimization techniques that would extend the visualization functionality towards a comprehensive decision support system for transportation.

Optimal Route Planning: Implementation of advanced pathfinding algorithms such as A* or Dijkstra would enable the system to automatically calculate optimal multi-leg journeys considering travel time, number of transfers, and costs. This natural evolution of the existing visualization would allow users not only to view existing routes but to receive intelligent suggestions for their travels, transforming the system from a passive display tool into an active travel assistant.

Intelligent Scheduling: Development of a PDDL-based (Planning Domain Definition Language) scheduling system could automatically optimize train timetables considering operational constraints, line capacity, and passenger demand. The planning domain would model actions like ”move$_train”, ”station_stop”, and” track_change” withspecificpreconditionsandeffects, enablingautomaticgene$

Dynamic Real-time Replanning: Implementation of replanning algorithms would allow the system to handle service disruptions, delays, and failures in real-time. Through sophisticated replanning techniques, the system could automatically recalculate alternative routes and update visualizations instantly, minimizing passenger impact and providing immediate solutions to operational problems while maintaining the visual feedback that makes the current system effective.

Multi-objective Optimization: Development of complex cost functions that balance travel time, comfort, price, and reliability would enable personalized journey planning. Integration of heuristic search techniques would allow finding solutions that satisfy different user preferences, whether prioritizing speed, cost-effectiveness, or comfort, while maintaining the intuitive visualization interface.

Predictive Planning: Use of predictive planning techniques could anticipate operational problems and suggest preventive actions. The system could analyze historical patterns to predict delays and propose mitigation strategies, transforming transportation management from reactive to proactive while leveraging the existing visualization infrastructure to display predictions and recommendations clearly.

9 References

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