Commandtrein

Project Documentation

Kaya-Sem Van Cauwenberghe
April 12, 2025

Abstract

Commandtrein is a command-line interface (CLI) tool designed to access and display timetables and route information for NMBS (Belgian Railways) directly from your terminal. It provides real-time train schedules, connections between stations, and transit disturbance information through a user-friendly terminal interface. The tool integrates with the iRail API to fetch live data and offers features like station search, timetable viewing, and route planning with customizable shortcuts for frequent routes.

Document version: 1.0

1 Project Overview

1.1 Introduction

Commandtrein is a CLI tool that helps users find train connections in Belgium. It was developed to provide quick and easy access to NMBS train schedules and route information directly from the terminal. The project aims to simplify the process of checking train timetables and planning journeys for Belgian rail users. At the time of writing, I was a student at Ghent University and used the train to get to campus daily. Other students who liked the terminal were also interested in such a tool.

1.2 Key Features

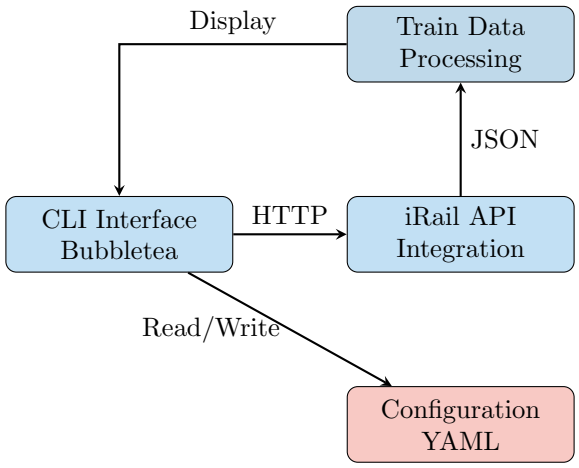
- **Timetables**: Retrieve and display current timetables for any NMBS station
- **Route Planning**: Get detailed connections and travel times between stations
- **Shortcuts**: Configure and use shortcuts for frequently used routes
- **Transit Disturbances**: View current transit issues and planned works
- **Interactive Interface**: User-friendly terminal UI with detailed information display

2 Technologies Used

2.1 Core Technologies

- **Programming Language**: Go
- **Frameworks & Libraries**:
  - Bubbletea (TUI framework)
  - Cobra (CLI framework)
- **External APIs**: iRail API for Belgian train data
- **Tools**:
  - GitHub Actions for CI/CD
  - YAML for configuration

2.2 Technology Diagram



3 Technical Details

3.1 Architecture

The application follows a modular architecture with clear separation of concerns:

- **CLI Layer**: Handles user interaction and command parsing using Cobra
- **API Layer**: Manages communication with the iRail API
- **Data Processing**: Handles JSON parsing and data transformation
- **UI Layer**: Implements the terminal user interface using Bubbletea
- **Configuration**: Manages user preferences and shortcuts using YAML

3.2 Data Flow

1. User input is processed through the CLI interface
2. Commands are parsed and validated
3. API requests are made to iRail for relevant data
4. JSON responses are parsed and transformed
5. Data is displayed in a formatted table using Bubbletea
6. User can interact with the displayed information

4 Challenges & Solutions

4.1 Technical Challenges

- **API Integration**: Handling various API response formats and error cases

  The iRail API was not a joy to work with. The JSON responses it provides are deeply nested and quite long. An average response for a connection between two stations can be thousands of lines long and I've seen nesting up to 25 leves in it. Not fun to parse or read.

  - Solution: I complained loads but managed to work with it. I just hope nothing ever breaks or changes because I do not want to touch that API again. (It is somewhat hypermedia driven which is cool.)

- **Terminal UI**: Creating an interactive and responsive interface that adapts to narrow terminal sizes.

  - Solution: Adapted Bubbletea tables and termsize listening.

- **Cross-Platform Compatibility**: Ensuring consistent behavior across different operating systems

  - Solution: Implemented platform-specific builds for Go using GitHub Actions

4.2 Lessons Learned

I came across some limitations of CLI libraries that do not suit me. Libraries like cobra force you into a certain way of creating and interacting with CLI applications. This is ofcourse understandable. Standardisation and implementation / usage complexity matters.

I find myself too often wanting to implement certain command parsing that I just cant.

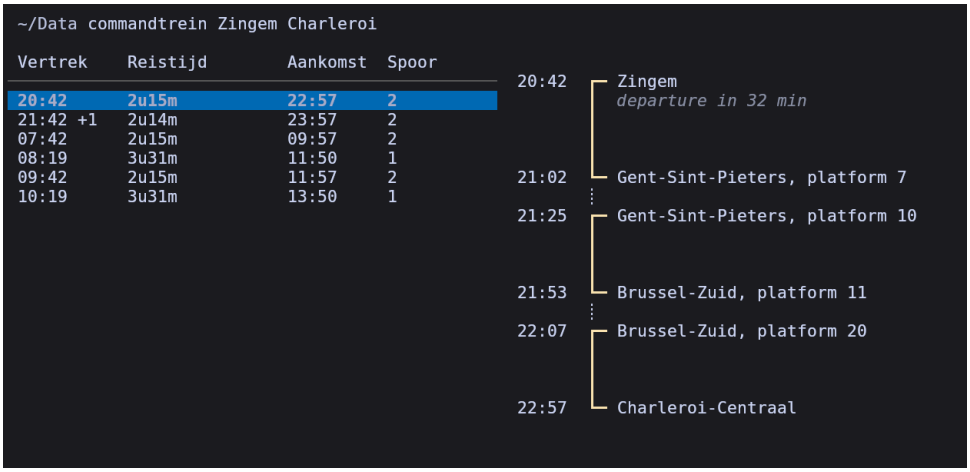


Figure 1: Commandtrein output when asked for connection Zingem - Charleroi

5 Screenshots & Results

6 Conclusion

Commandtrein successfully provides a convenient way to access Belgian train information through the terminal. The project demonstrates effective use of modern Go tooling and frameworks to create a user-friendly CLI application. Future improvements could include:

- Enhanced error handling and recovery
- Additional customization options
- Integration with more transportation services
- Offline functionality for basic operations