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

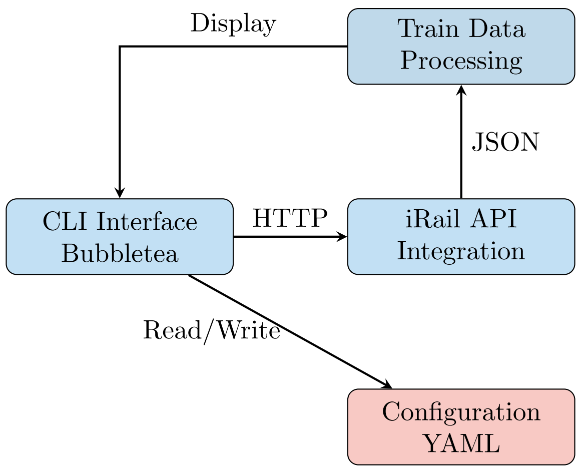
- **Timetables**: Retrieve and display current timetables for any NMBS station
- **Route Planning**: Get detailed connections and travel times between stations
- **Shortcuts**: Configure and use shortcuts for frequently used routes
- **Transit Disturbances**: View current transit issues and planned works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

3.2 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 levels deep. 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

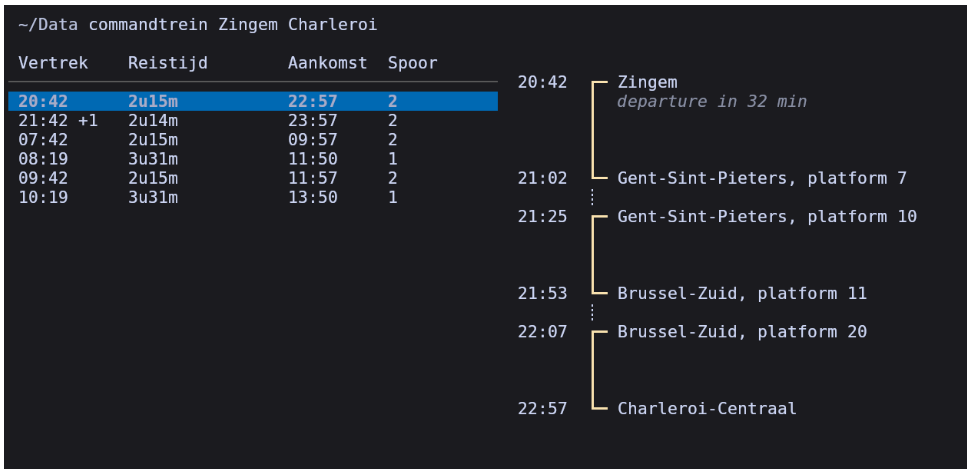
3.3 Thoughts on CLI Libraries

While working on this project, I encountered limitations with existing CLI libraries that didn't align with my needs. Popular libraries like Cobra enforce specific patterns for creating and interacting with CLI applications. While this standardization makes sense from a maintainability perspective, it can be restrictive.

The hierarchical command structure common in these libraries, where arguments must follow a strict command tree pattern, felt overly constraining for my use case. I found myself frequently wanting to implement more flexible command parsing patterns that weren't possible within these frameworks' constraints. This experience highlighted the trade-off between standardized patterns and implementation flexibility in CLI development.

Given these limitations, I am considering developing my own CLI framework that emphasizes scriptability and customization. The goal would be to create a more flexible system that allows developers to define their own command parsing patterns and interaction models, while still maintaining good practices and conventions. This would enable more creative and tailored CLI applications without sacrificing maintainability.

4 Screenshots & Results



```
~/Data commandtrein Zingem Charleroi

Vertrek     Reistijd         Aankomst  Spoor
                                                 20:42  ┌─ Zingem
20:42       2u15m            22:57     2                │  departure in 32 min
21:42 +1    2u14m            23:57     2                │
07:42       2u15m            09:57     2                │
08:19       3u31m            11:50     1                │
09:42       2u15m            11:57     2         21:02  └─ Gent-Sint-Pieters, platform 7
10:19       3u31m            13:50     1                ⋮
                                                 21:25  ┌─ Gent-Sint-Pieters, platform 10
                                                        │
                                                        │
                                                 21:53  └─ Brussel-Zuid, platform 11
                                                        ⋮
                                                 22:07  ┌─ Brussel-Zuid, platform 20
                                                        │
                                                        │
                                                 22:57  └─ Charleroi-Centraal
```

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