Design Proposal for Route Optimization System

List of Contents -

Introduction

The purpose of this document is to provide a detailed architecture design of the Route Optimisation System. This document will address the background for this project, and an architectural overview, which includes a bird's eye view and a full description of sub systems that will be required to address the core requirements. The intention of this document is to help determine how the system will be structured at the highest level.

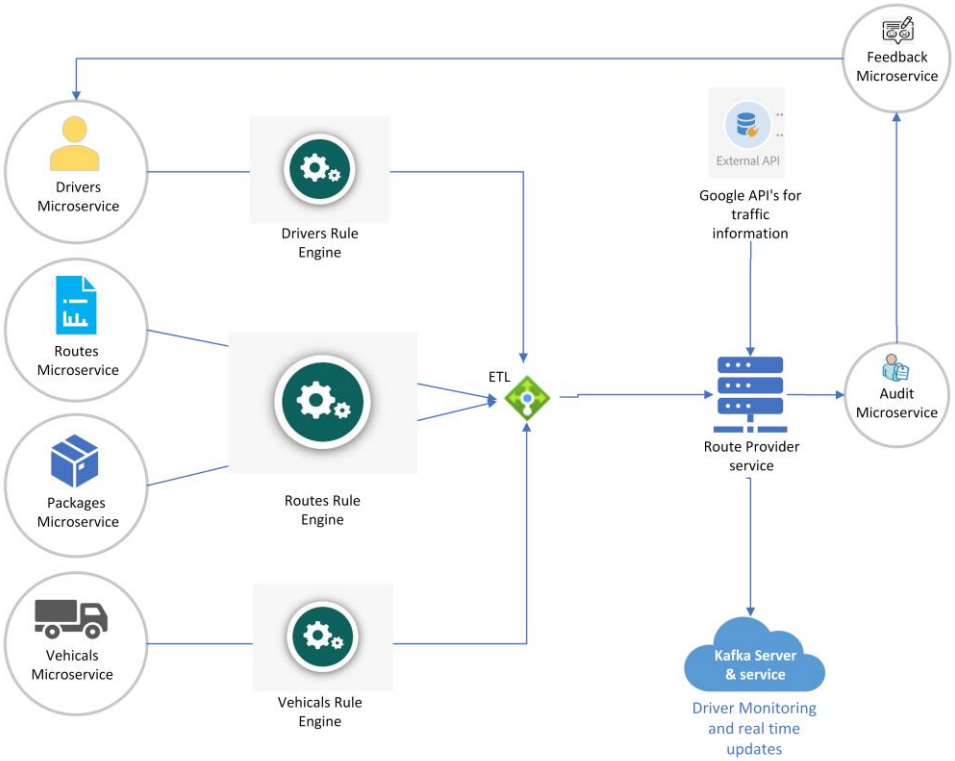
Architecture Overview

The Big Picture

The major subsystems or containers of this system are shown in the picture below. We intend to follow a micro service architecture patten, which has proven to be very successful in complex systems. It provides us with added benefits like high availability, maintainability and testability which are the key quality factors for any IT system.

Each of these microservices will be vital for the decision making and system operation. The Driver's information, routes information and vehicles information are like the basic core information which will help in the decision making and act as inputs for the route provider API, which is the key component.

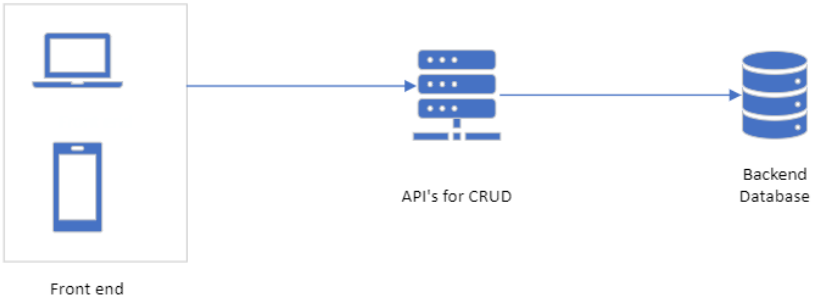
In the next section, we will go through each of these microservices in details.



Drivers Microservice

It will store and provide us with drivers' information like Name, Age, ID details, license information and other personal information.

It will also hold their preferences and holiday schedules. Preferences may include the following information like preferred routes, preferred shifts like day shift or night shift, etc



This will be the basic components of the Drivers microservice – It will have a User-friendly interface, a backend database and an API, which will perform all the CRUD operations on the Driver's data.

This microservice will interact with a rule engine, whenever it needs to send out driver details, which is described in the next part.

Driver Rule engine

It will contain the business rules that will be applied to each driver whenever the driver information is fetched. It will also fetch data based on their preferences and holiday schedules

Example use case

In EU, we have a rule for breaks and rest for Drivers as per the law. A driver has to take a 11 hour rest every day or he can break it up to 9 hours rest 3 times between any 2 weekly rest periods.

So, the rule engine will calculate the numbers of hours that each driver has worked for the week or month, and if he satisfies the rules that are set, then his data will be sent to the next part of the workflow.

The advantage of using this microservice architecture and rule engine associated with it is, if the load increases on any part of the system, its easily scalable

Package Microservice

The architecture of this microservice is same as that of driver microservice mentioned above.

In the package microservice, we will have a frontend, backend and API's. It will store data about how many packages needs to be picked up or dropped off at each route on a particular day.

Routes Microservice

It stores and provides routes information like Place name, the load times, wait time, drop off time, etc

The microservice will have the same architecture as shown for the drivers. It will compromise a Frontend, API calls for CRUD operations and a backend to store information

Routes Rule Engine

The routes rule engine will have any rules associated with the routes. It may be any government laws or rules that the business has in place

Example use case

The rules associated with routes may be as follows.

Rule 1- Allow only certain height width vehicles on the route.

Rule 2 – Has a loading bay area but no unloading docks.

So, whenever any route information is fetched for planning purposes, it will be run through the rule engine, to check if it passes all the criteria that we are looking for.

Vehicles Microservice

It stores and provides vehicle information like Vehicle number, type of vehicle, capacity, type of storage (like - dry / cold), Current status (eg- On road, Available for travel, etc), Total on road time, service days, etc

Vehicles Rule Engine

It will have some rules that might be required for the vehicles which the business might have defined.

Example use case

If the rule engines has rule like -

Rule 1- One vehicle cannot be continuously running on the road for more than X hours.

Rule 2- Vehicle should have permit for cross border deliveries

Before fetching any vehicle information, it will run through the rules and present you with information with all the vehicles that satisfy the conditions.

ETL component

We will have a ETL component, which will fetch, collate and transform the information from all four aforementioned micro services, namely- drivers, vehicles, routes and packages and present it to the core part of the system that will make the decisions i.e., Route provider microservice

Route provider Microservice

This is the core system.

Based on the day or the week and the routes (more like pickup or drop points) that are selected and the packages that needs to be picked up and dropped off. It will get you the number of vehicles that might be required, drivers that will be available and the most optimised route

Here, we will have to use the box stacking problem here to solve the issue of in each pickup and drop off, how the vehicle will be loaded.

And also, for the route optimisation we will use some of the algorithms that would best suit our purpose and have proven to be efficient.

This microservice also needs to interact with other external Google APIs to get the real time data for traffic updates on the selected route.

Driver Monitoring and real time updates

This will be a service which needs to be run in the background, which will poll the backend service from the driver's truck to update the location

Audit Microservice

The architecture of this microservice is same as that of driver microservice mentioned above.

In the audit microservice, we will track each of the pickup and drop off in the designated route and status, whether it was successful or not. The driver / supervisor will be marking off the deliveries and pickups with appropriate status and time.

Feedback Microservice

The architecture of this microservice is same as that of driver microservice mentioned above.

The feedback microservice, will be used to get feedback about the deliveries, drivers and the planned routes.

Technology Stack

Frontend

- ### Mobile Apps for Drivers
    - Flutter for iOS and Android Apps
- ### Web Admin and Consumers
    - Typescript
    - ReactJS
    - Redux – Thunk/Saga
    - Storybook
    - Mui Components

Backend

- Typescript | Python
- Microservices
- Node with Nest.JS for Rest APIs
- Kafka Server
- Services
    - drivers service – REST APIs | Nest.JS | Typescript
    - routes service – REST APIs | Nest.JS | Typescript
    - packages service – REST APIs | Nest.JS | Typescript
    - vehicles service – REST APIs | Nest.JS | Typescript
    - audit service – REST APIs | Nest.JS | Typescript
    - feedback service – REST APIs | Nest.JS | Typescript
    - route provider service – REST APIs | Django/Flask | Python
- Rule Engines – Python based
    - routes engine
    - driver engine
    - vehicle engine
- DB – PostgreSQL and MongoDB
- ETL

Infrastructure

- Cloud
- Pipeline – Jenkins
- Kubernetes Cluster
- Private VPC
- VPN
- Load balancer