Connected Cars

Dev: Jayendra Varma
Email: vkjayendravarma@gmail.com
Contact: 9490230173

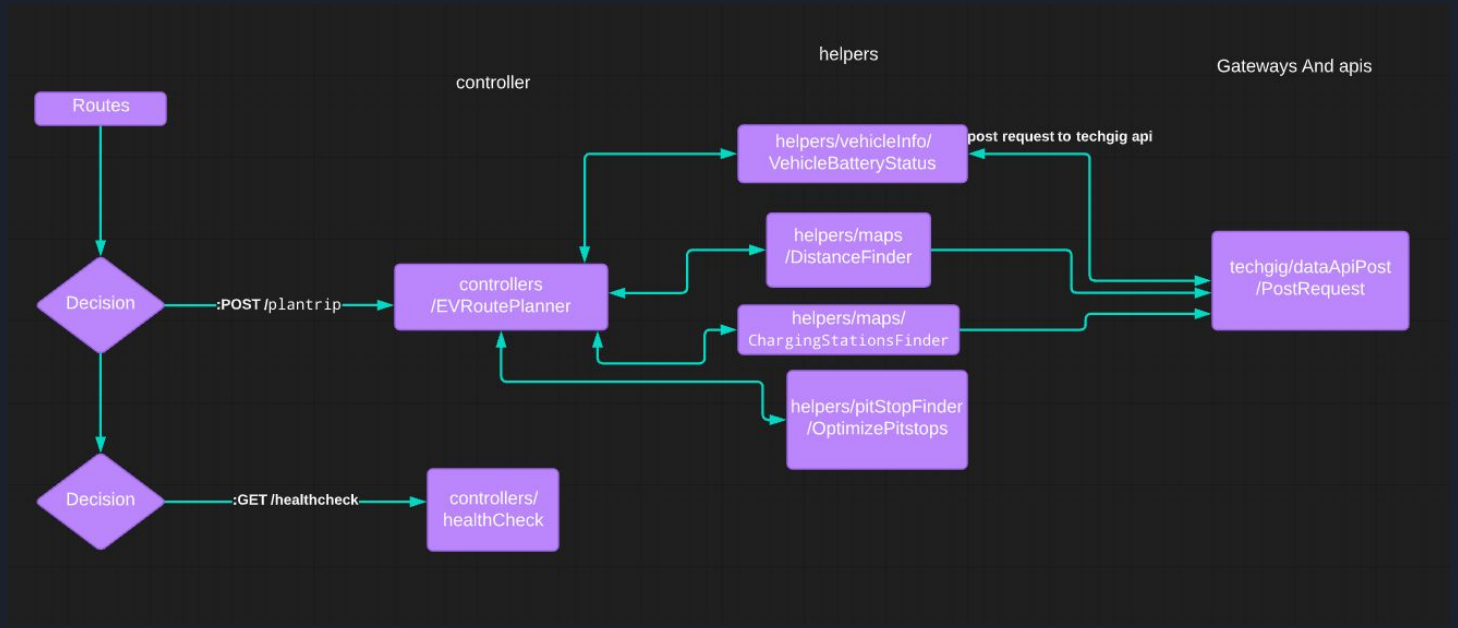
Brief Summary of Solution

When you make the api call to get route plan, application first checks the vehicle battery status and distance. If charging is not required then returns.

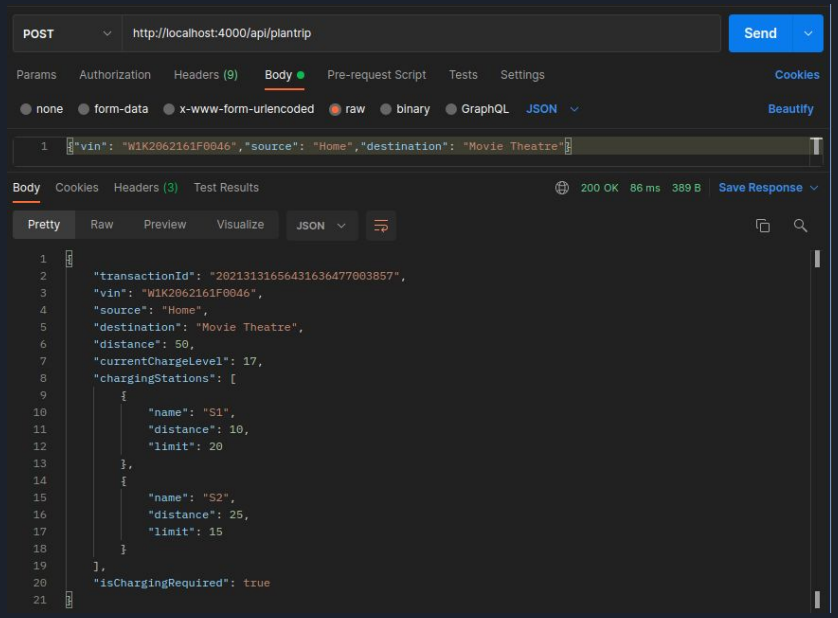
If charging is required, then all the charging stations in the route are retrieved. Then iterating stations I retrieved the maximum possible reach with current battery reserve. Then station which give maximum range is flagged as used and added to the response.

This approach uses concurrent api calls to get battery status and distance because those two are mandatory and charging stations are required only if recharge is required, there by using cpu efficiently to increase performance.

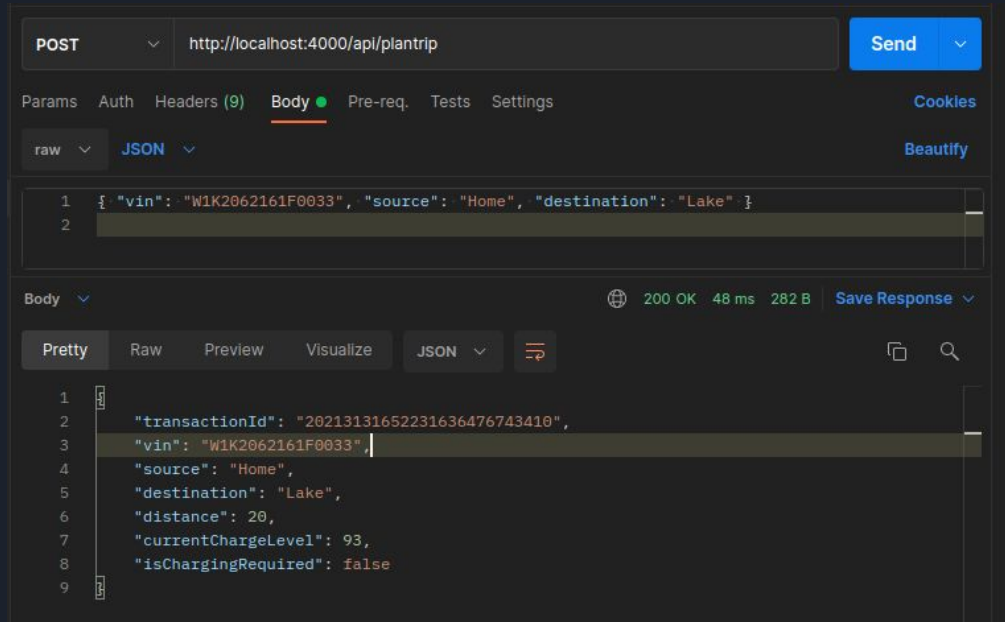
System Design



Demo output



When charging is required - response time 86ms



When charging is not required - response time 48ms

Why is this solution the best

This solution is designed with a mindset of cloud and delivering the response faster. This has a sweet spot of not compromising most in performance and spending a lot on network load and hosting. I used cpus and storage efficiently by using concurrent programming concept and also reduced duplication of data, thereby reducing the response time and storage costs. Response time is as low as 48ms when charging is not required or if there is any error with retrieval of battery status or distance from origin and around 100ms response time when recharge is required.

Tools and techstack

Tech stack:

Golang, gorilla mux

Docker

Tools:

Git

Postman

How to run the app

- Unzip the code file
- `sudo docker build -t <name-of-img> .`
- `sudo docker run -p <port>:4000 -it <name-of-img>`
- Copy the ip address you get in the console
- Open it in browser <ip address>:<`port`> to test

How to check

Use postman to get results

url: http://<IP Address>:5000/api/plantrip
method: post
content-type: application/json
body: { "vin": <vin>, "source": <source>, "destination": <destination> }

Explanation

Routes: This module handles the http calls and calls the respective controller functions

controllers: This module contains controllers

Gateways and Apis: This module contains the configuration and interacts with external apis and gateways to provide data to helper functions and controllers

Helpers: These functions are addition functions which are used by controllers

Prototypes: This is a data layer which contains structures and functions to modify data

Source code

https://github.com/vkjayendravarma/techgig-benz

Thank You