WebApp-API-Backend

The WebApp-API-Backend transforms transportation data sets (routes, stop_times, stops, and trips) into a fully functional API, enabling easy access to real-time transit information. It simplifies integration for developers to query and interact with transit data seamlessly.

GET Get All Routes

http://localhost:3000/api/routes

Reads data from the `routes` file, processes each line into a route object containing `route_id`, `route_desc`, and `route_type`, then returns the list of all routes. It also handles sorting the routes based on a query parameter (`asc` or `desc`) and returns the sorted list as a JSON response.

GET Get Routes By ID

http://localhost:3000/api/routes/1657

Fetches a specific route by its `route_id`, processes the route data, and returns the corresponding route object. If the route is not found, it returns a 404 error message.

GET Get All Trips

http://localhost:3000/api/trips

Reads data from the `trips` file, processes each line into a trip object with `route_id`, `service_id`, and `trip_id`, then returns the list of all trips as a JSON response.

GET Get Trips By ID

http://localhost:3000/api/trips/49794889

Retrieves a specific trip by its `trip_id`, processes the trip data, and returns the corresponding trip object. If no trip is found, it responds with a 404 error message.

GET Get All Stops

http://localhost:3000/api/stops

Reads data from the `stops` file, processes each line into a stop object with `stop_id`, `stop_name`, `stop_lat`, `stop_lon`, and `location_type`, then returns the list of all stops as a JSON response.

GET Get Stops By ID

http://localhost:3000/api/stops/22482

Retrieves a specific stop by its `stop_id`, processes the stop data, and returns the corresponding stop object. If no stop is found, it responds with a 404 error message.

GET Get Stop times by trip_id

http://localhost:3000/api/stop-times/49765808

Reads data from the `stop_times` file, processes each line, and checks if the `trip_id` matches the requested `trip_id` parameter. For matching trip IDs, it collects the stop times (arrival time, departure time, stop ID, and stop sequence) into an array. If no stop times are found for the provided trip ID, it returns a 404 error message. Otherwise, it sends the list of stop times as a JSON response.

GET Get Shortest path for a route_id

http://localhost:3000/api/shortest-path/1657

This file defines the controller for fetching the shortest path for a given route ID. It leverages Axios to make API calls to retrieve necessary data about trips, stop times, and stops. The core function is to calculate and return the stops involved in the shortest path for a specified route. Fetches all stops for a specific route based on the `routeId`. It queries multiple endpoints to get the relevant data and returns a list of stops for the given route.