Musixmatch Software Engineer - Backend Test

API Documentation

Tech specification

I've developed the test with PHP Version 8.0.3 and MariaDB v. 10.4.18.

I've decided not to use a framework like Laravel or Symphony because it's not part of my everyday stack and it would take too long according to the assignment deadline.

I haven't developed any auth/tokenization for travel log, assuming that API will work in Virtual Private Network API (entering and exiting use GET method).

API Endpoint

This is the structure of API Endpoint:

http://{HOST_NAME}/{PATH}/index.php/{MODULE_NAME}/{METHOD_NAME}?{PARAM_NAME}={VALUE}

where:

- {HOST_NAME}: represents the host of the application (ex. http://localost/)
- {PATH}: represents the sub path of the application (ex. /musixmatch/api/)
- {MODULE_NAME}: represents the module called by user; the methods supported are
  - /customer/
  - /travel/
- {METHOD_NAME}: is the module called action:
  - The /customer/ method supported:
    - amountDue: {PARAM_NAME} list (month *required,* customer_id)
  - The /travel/ methods supported:
    - enter: {PARAM_NAME} required (enter_station_id, vehicles_license_plate)
    - exit: {PARAM_NAME} required (exit_station_id, vehicles_license_plate).

Here you can find three calls, as example, of each supported method:

- http://localhost:8080/musixmatch/API/index.php/customer/amountDue?customer_id=1&month=3
- http://localhost:8080/musixmatch/API/index.php/travel/enter?enter_station_id=1&vehicles_license_plate=AA001BB
- http://localhost:8080/musixmatch/API/index.php/travel/exit?exit_station_id=1&vehicles_license_plate=AA001BB

Introduction

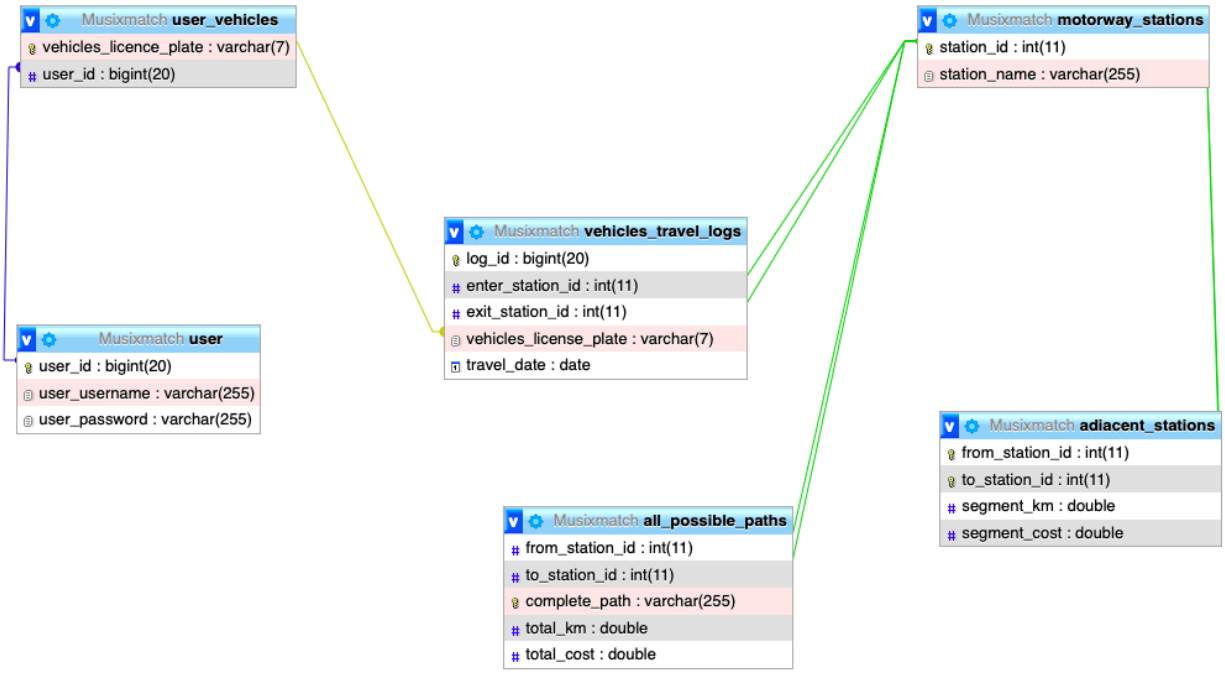
To develop this assignment I've assumed, as indicated in the document assignment, that all data (except travel logs and all possible paths) are collected as a given dataset;
Clearly I've developed the entire structure about the database and how data has to be stored but, the data itself, is considered a reliable dataset with correct data.

I've created an association in database between registered user and its vehicles (registered by a license plate as a primary key) so that each costumer could have more devices installed in their vehicles but all reconducted to the same customer.

The motorway company provides a list of stations of their network.
To elaborate all possible paths, I've chosen to use a graph.
I've created a matrix for each adjacent station and assigned to it two properties: the distance in kms between the two stations and the cost for that specific segment.

I've assumed that you can reach any destination from each station, as indicated in the rules of the assignment.
In addition, I've assumed that each station is reachable only from the adjacent one so, theoretically, Station 1 and Station 2 are connected by one specific path (according to the

adjacency matrix); in case of an error, the shortest path (given by the sum of kms) between them will be taken into account.
Here is a snapshot of the database scheme (see DBScheme and Relations.pdf for more details)



Project Scheme

The API project is provided by:

- API
  - Class
    - graph.php
    - travel_logs.php
  - Config
    - bootstrap.php
    - mysql_database.php
  - Controller
    - base_controller.php
    - customer_controller.php
    - travel_controller.php
  - Database
    - Musixmatch.sql
  - index.php
  - unit_test.php
  - visual_test.php

Class

This file represents the object used to handle the data.

db_connection.php

Simple object class that allows to set and open a new mysql connection.

graph.php

Contains the properties and the methods to create a Graph object allowing to recover the data from “adiacent_stations” and create a directed weighted graph.
The method used to do this is “buildGraph()”;

Once obtained the graph with the fetched data we can launch the method “buildAllPaths()”;
This method combines the possible from-to tuple and pass it to the function “findAllPath()”;
This function performs an DFS (Depth First Search algorithm) across the graph to find all possible paths between two nodes and, for each path found, it will be put into the database table “all_possible_path”.

N.b. I know the existence of a graph database but I don’t have much familiarity so I’ve opted for a more conventional solution.

travel_logs.php

This file contains the properties and methods to create a TravelLogs object used to provide inserting an ENTER/EXIT from the motorway.

The ENTER method is a simple insert into the correct table of the enter; it only requires the enter station id and the license plate of the car.
The date is automatically set to today.

The EXIT method performs an update on the database table of the most recent row relative to a specific customer license plate;

I’ve assumed the device is like “telepass”, so it is not possible to enter/exit from the motorway without a correct collect data (device or manually operator).

Config

Bootstrap.php:ì is a simple utility file that provides to require and set global variables included in the main page.

Controller

This contain the controller that handles the correct request by the user.

base_controller.php

Utility used to get the correct structure of the url and return query string parameters;

customer_controller.php

This controller allows to recover the amount due from a specific customer in a given month;
If the customer is not specified, it returns the amount due from all customers in a given month.

It allows GET method request.

Data is returned in json format.

travel_controller.php

This controller also uses GET method.

It enables to set ENTER or EXIT log by passing the correct station id, and the license plate.

Database

It contains Musixmatch.sql that is the export of the sample database scheme + dummy data used to perform this test.

API

index.php

It is the main page of the API and is responsible to understand what the user asked and call the correct controller required, if method is not supported, it returns an error;

unit_test.php

Contains some call to test if the main functions work properly.

visual_test.php

Calls the Customer amount due API and visualize into table.