Julian Cuevas
Gabriel Gonzalez

CAAMBus

A. Domain Description - *University Transportation*

- The transportation services of the university, specifically the trolley service, takes passengers (primarily students and staff) throughout the different buildings inside and outside the campus.
- The Department of Transportation is in charge of authorizing and dispatching each trolley trip.
    - Specifically, the department of transportation creates a new itinerary ticket with basic details pertaining to the driver, vehicle and route.
    - Each time a new itinerary is created, it is saved to the database.
- While on the road, the trolley follows the route that was specifically assigned. The only exception is when there are obstacles (traffic jams, closed roads, etc.). The driver documents when and where they were "off-route".
- Driver aren't assigned a specific vehicle, but rather are tied to one on a day-by-day basis through the itinerary assigned to them,

B. User Stories

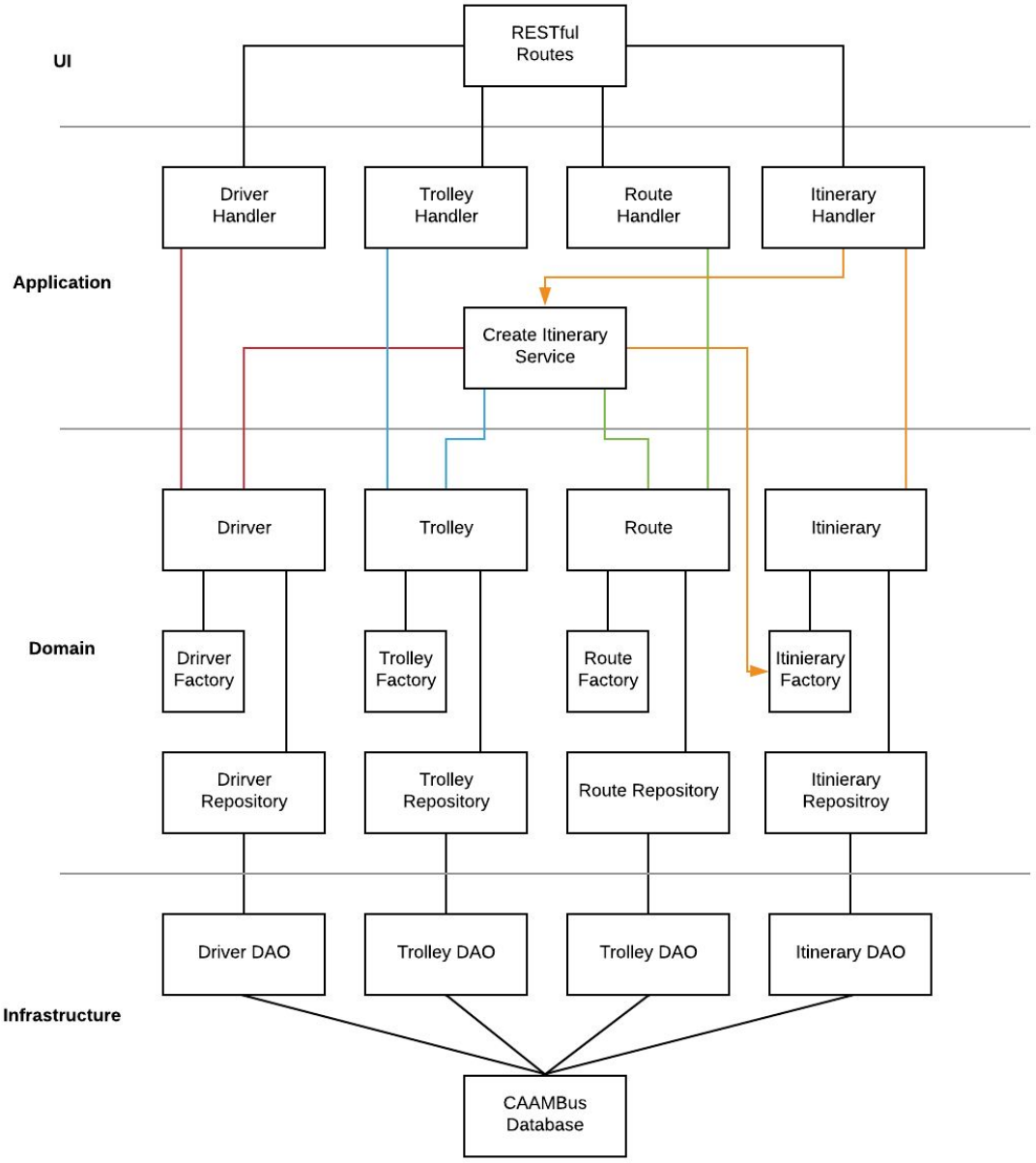
- **<u>CHANGES:</u>**
    i. Removed tracker related stories. Trackers will not be a part of this project. Outside scope.
- As an administrator, I want to be able to assign a driver a vehicle so that they can carry out trolley routes.
- As an administrator, I want to be able to issue an itinerary for a specific route so that the system can keep track of the drivers and vehicle in each route.
- As an administrator, I want to be able to create, edit or delete routes in the system.
- As a driver, I want to be able to view the itinerary I've been assigned so that I may carry out my day's work.
- As an administrator, I want to be able to manage itineraries so that I can deal with unexpected circumstances.

C. Requirements

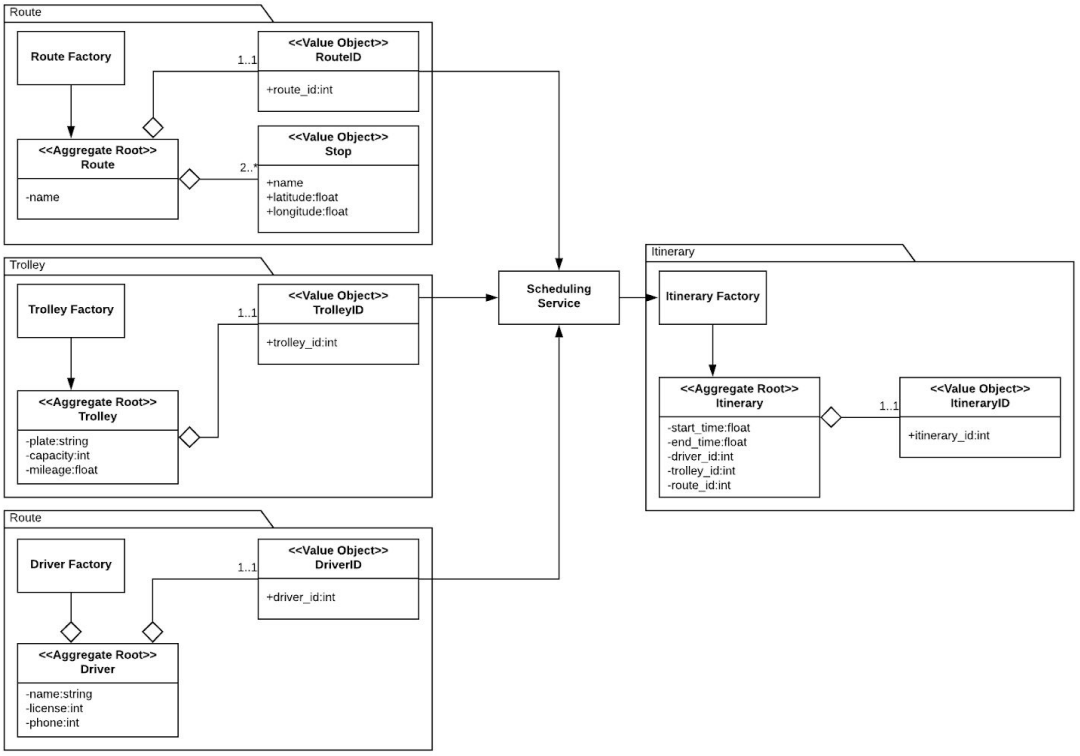
- The system has to aid the workers of the Department of Transportation to create an itinerary.
    - The itinerary must be associated with a specific driver, vehicle, & route. Within it, there is also the start time and end time.
- Once the trip ends, the system must allow:
    - The mileage of the associated vehicle be updated.
- Whenever the administration needs to create a monthly report, the system must gather all the trips taken in a certain month and generate a report that calculates

important statistics like: mileage driven for each driver and vehicle, passenger count per route, etc.

D. Layered Architecture



E. Diagram



F. Implementation

- Infrastructure Layer

The infrastructure layer is the part of the software that communicates with the database. Each of the different models have their own “data access object” class associated with it. The DAO’s communicate with the database to fetch the data so that the repositories can populate the objects in the domain layer and then the application layer can distribute the data to the UI. Each DAO has a class method pertaining to the different queries needed for the main functionalities to occur. For example (in regards to a driver), there is a query for getting all the drivers that the university currently employs. There are also queries to create, update and delete drivers. Specialized queries are built for getting drivers by a certain attribute, or for getting all the trolley trips a driver has driven.

- Domain Layer

The domain layer is the part of the software that stores the models, the repositories and the factories that creates new instances of the model that get sent to the DAO for persistence. Most models have their implementation complete as most of them do not have complex actions in the scope of this system. The exception is the itinerary, which requires that the actions (such as starting and ending the ticket, getting real-time information, etc.) be designed further.

- Application Layer

The application layer handles all the incoming API requests that the program uses to populate fields in the UI Layer. The handlers of the different requests call the DAO methods for each specific request. Each request is mapped one to one for each query that the DAO has. Once the response arrives, the data is used to populate a domain object through its respective repository, and the its information is jsonified and passed to the UI Layer.

- User Interface Layer

The user interface layer is the frontend of the application, which houses all the necessary actions in the models and other components of the system. For now, it currently displays all the models in their different tables. A form is being built for each of the model that will help the factory create a new instance.

Drivers

ID	First Name	Last Name	License	Phone Number
1	Julian	Cuevas	6523031	7876074678
2	Raul	Mojica	6523231	7875434237

Trolleys

ID	License Plate	Capacity Limit	mileage
1	GFN091	24	60000
2	ASL045	32	60000

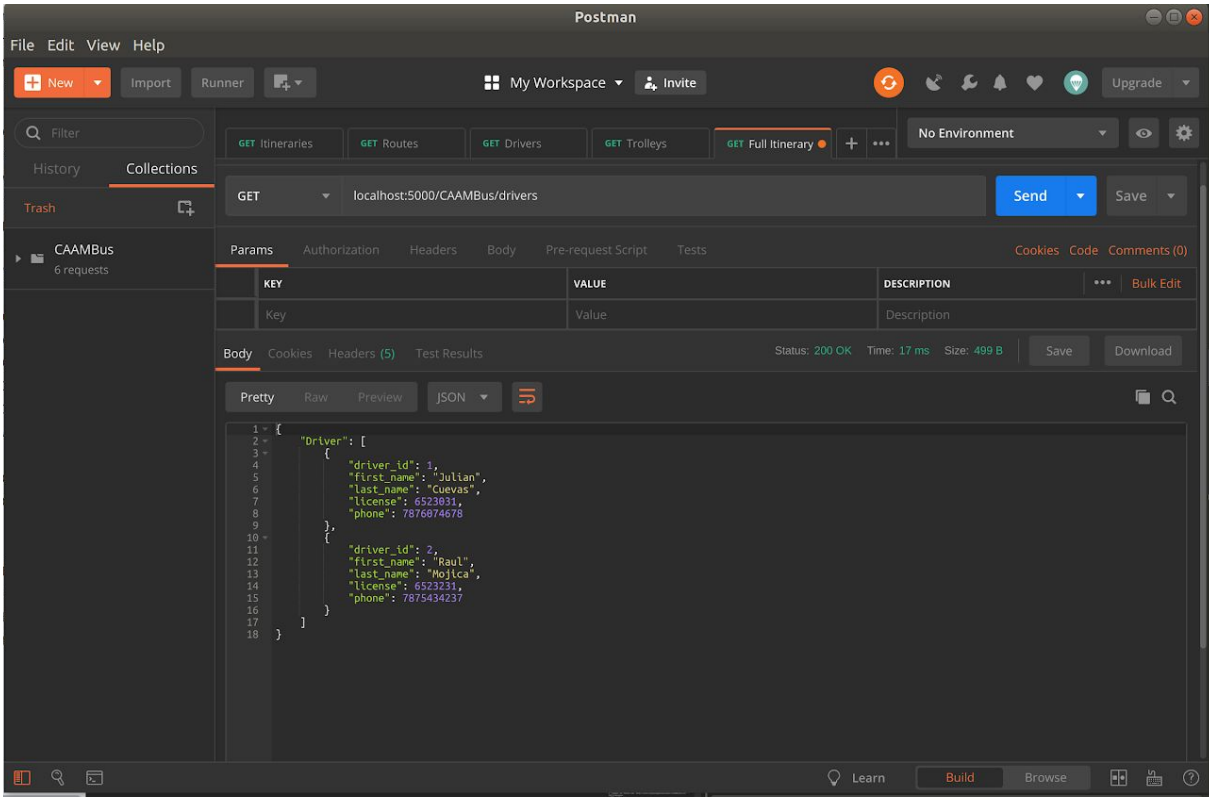
Routes

ID	Route Name
1	Dummy Route #1
2	Dummy Route #2

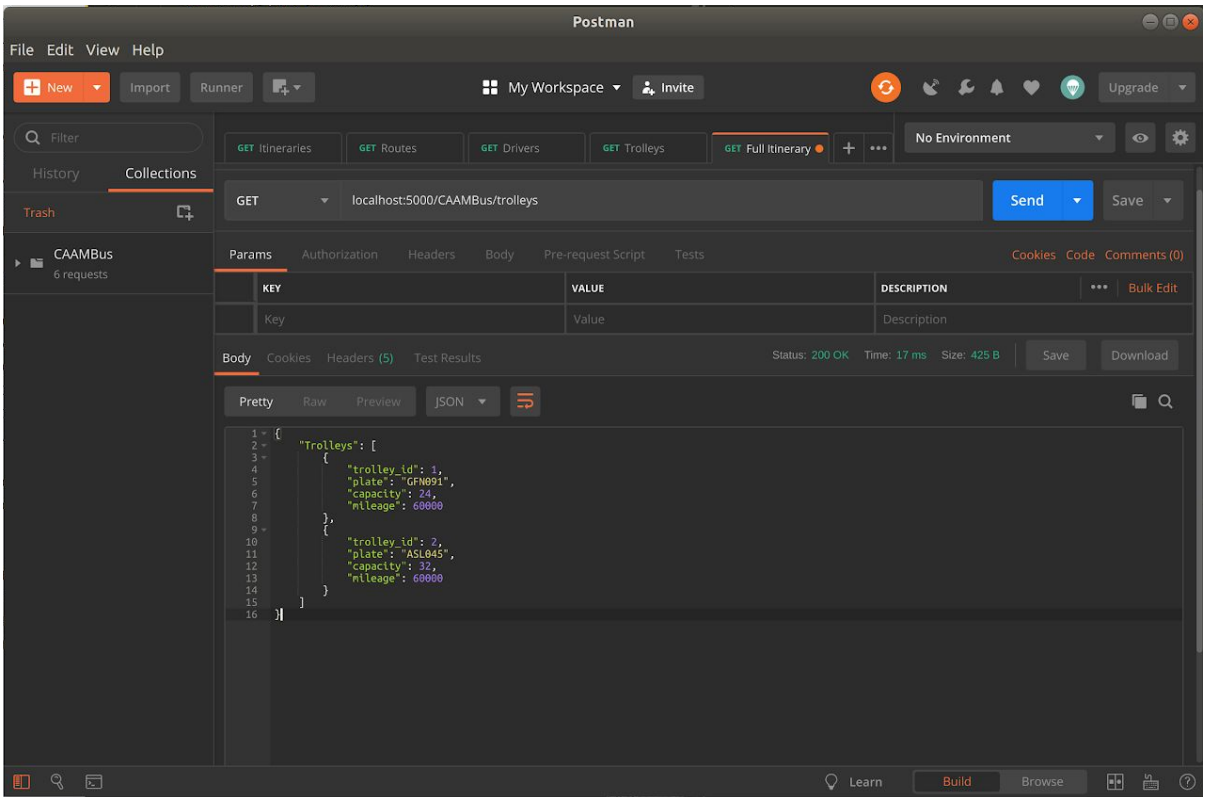
G. Testing

- Testing was carried out using Postman primarily to do integration testing as unit test, at least at this point, isn't the right way to go.
    i. Why not Unit Testing?
    The methods currently implemented in our app require a connection to the project's database for all functionality. As such, there is no way to test the methods independently from the database. Integration testing is more apt for this type of methods.

- Examples:
    i. Testing route to get all drivers' information:



    ii. Testing route to get all trolleys' information:



iii. Testing route to find driver by ID