Loadsmart - Back End Challenge
Fernando Paladini
January 2019

Vehicle Routing Problem

Challenge Report

Solving the problem

First thing I thought when I read the challenge was how similar the problem is to the "Travelling Salesman Problem", then I started researching for similar problems. The given problem is a classical graph theory problem in Computer Science field. After some research, I've found a kind of special problem derived from Travelling Salesman Problem called "Vehicle Routing Problem", that's almost what we need. In this specific problem, multiple trucks must transport loads for customers starting from it's own depot and finishing at the same depot, but always looking for optimising the global transportation cost. This cost may be monetary, distance, etc. In our case, the global transportation cost will be **distance**. It's known that "Vehicle Routing Problem" is classified as a NP-Hard problem, so our solution must use heuristics in order to solve the problem. That way, we're going to solve the problem always in a very good route, but not always with the best route. For reference, the "Vehicle Routing Problem" can be found on Wikipedia and some Computer Science classes on Internet. Other references used for this problem solving can be found on the bottom of this report.

It was not clear if the transportation of the loads should be:

A. A starting address that must be traveled street-by-street until the destination address, taking the best streets and routes across the city streets; or
B. A starting city that must be traveled in a straight line to the destination city, without giving specific directions about the streets that must be traveled.

Since it's a simplified problem that doesn't have much variables like gas availability and human resources associated with the travels, I'm going to assume the hypothesis B to this problem. If needed, I can go deeper in the problem and use more complex variables and algorithms to solve it.

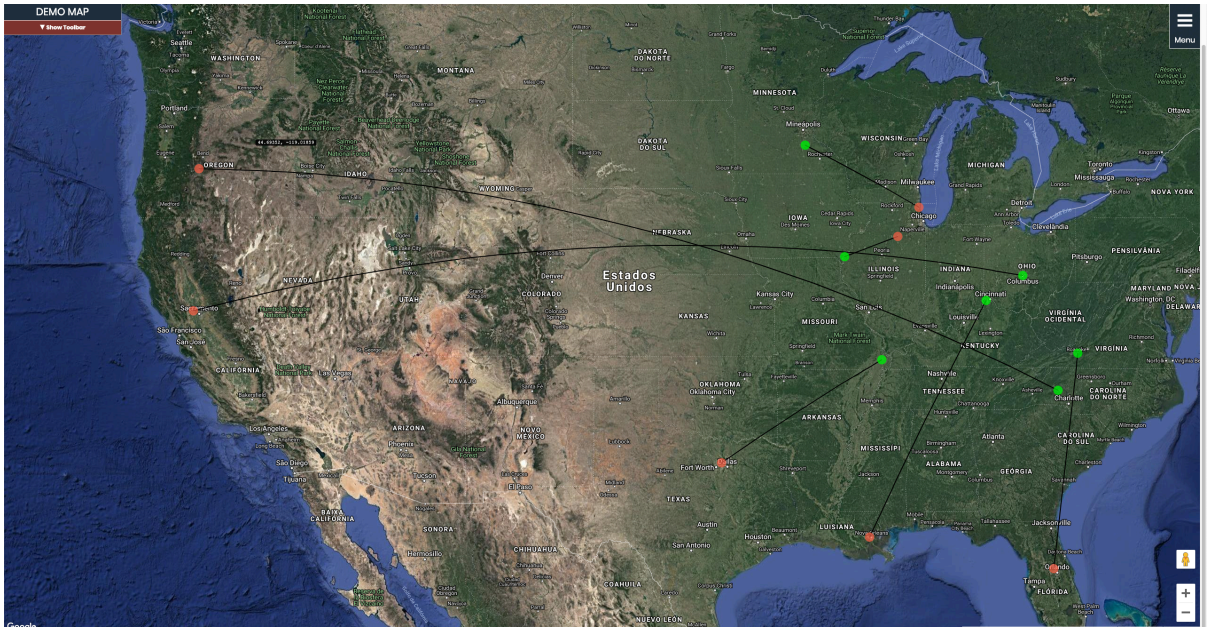
At the first time, I though that I should use Google Maps API in order to get the distance in kilometres/meters between two coordinates, but that would be slower and in a large scale /production environment it will be expensive too. So I remembered some aviation videos that I watched in the past and in these videos was explained that geographical coordinates can represent angles or distances through some math computations. That I way, I just got the insight that put me to find an equation that calculates a distance between two locations using only it's geographical coordinates - without any API call or similar. That was a major breakthrough when trying to find a solution to that problem. There's a small error rate related to these calculations (because the calculations uses a spherical earth as it's basis, while the Earth is actually slightly ellipsoidal), but the error rate is something like 0.3%. Doesn't seem to be a huge error rate for an estimative distance, so I'm going to follow this approach and calculate the best routes using only geographical coordinates. It's important to note that using Google Maps API to calculate distance is much easier and more precise than the chosen method, but it's also more expansive and slower (computationally speaking).

Exploration

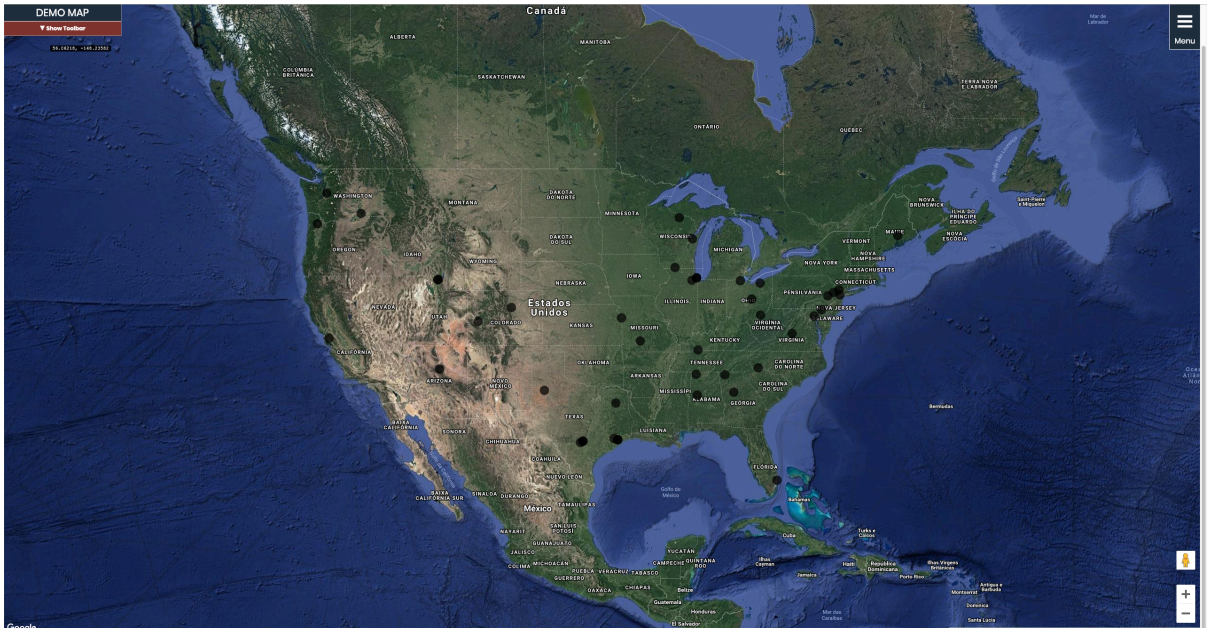
Before starting the development of the solution, it's always a good thing explore the dataset related to the problem. The datasets given are *trucks.csv* and *cargos.csv* and I'm going to plot everything on a map using their geographical coordinates, so I can have a wide view of this specific problem. In order to do that, I'm going to adapt the dataset to a specific format and see all trucks and loads on a map using a tool from Darrinward [1]. The formatted datasets will be available at the folder "*exploration*", so you can test it by your own too.

Below you can see all the cargos plotted in the map:

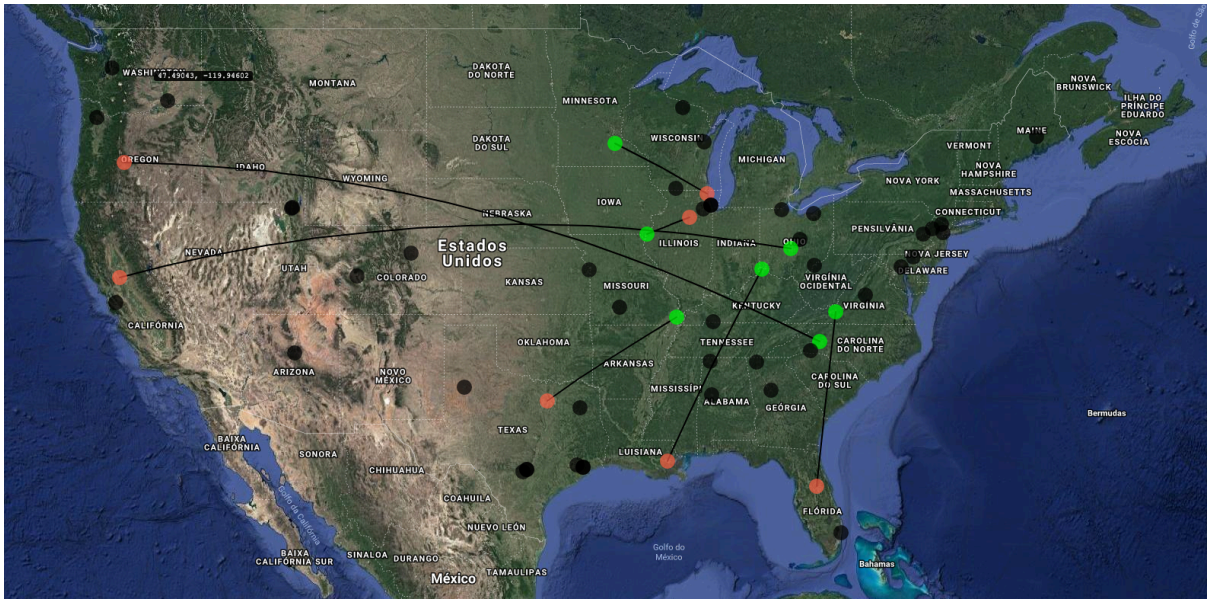
The green circles are the origin point for a specific cargo while the red circles are the destination of that same cargo. The line between these circles on the map are a simplified route in order to deliver the cargo.



In the following image you can see the location of all the trucks available for moving the cargos over US (the black circles on the map):



Now you can see all the trucks and all the cargos on the same map:



Development

I've seen the stack used by Loadsmart and since it uses Python, Django and PostgreSQL, I've decided to use exactly these technologies on my solution, so I can have play with part of your stack and also ease the task for who's going to check my solution and report. The development of the solution would use TDD (Test-Driven Development), so I'm going to first write the tests needed for CRUD operations related to the trucks and cargos. After that, I'm going to write the code to find the best mapping of trucks to cargos in order to minimize the overall distance the trucks must travel.

Below you can see the RESTful structure that will be created for this project:

Trucks REST API

Endpoint	HTTP Method	CRUD Method	Result
api/v1/trucks	GET	READ	Retrieve all trucks
api/v1/trucks/:id_truck	GET	READ	Retrieve a single truck
api/v1/trucks	POST	CREATE	Insert a single truck
api/v1/trucks/:id_truck	PUT	UPDATE	Update a single truck
api/v1/trucks/:id_truck	DELETE	DELETE	Delete a single truck

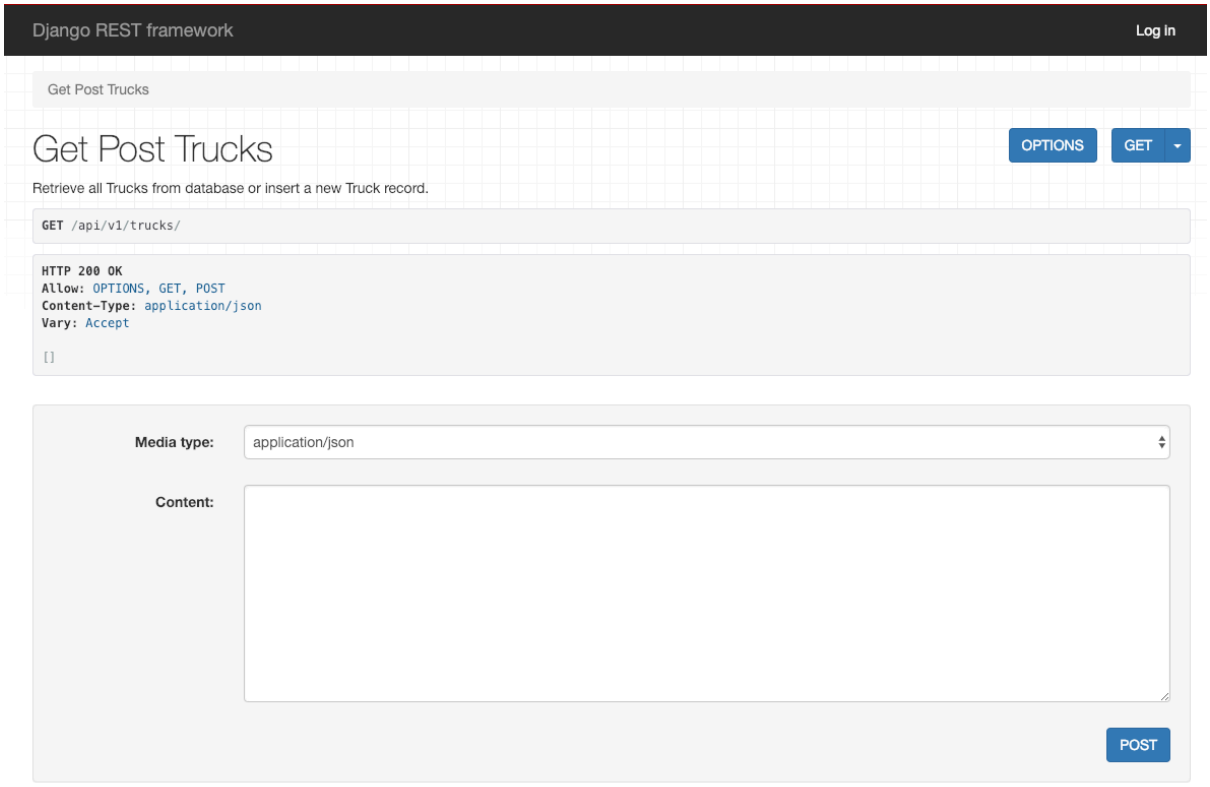
Loads REST API

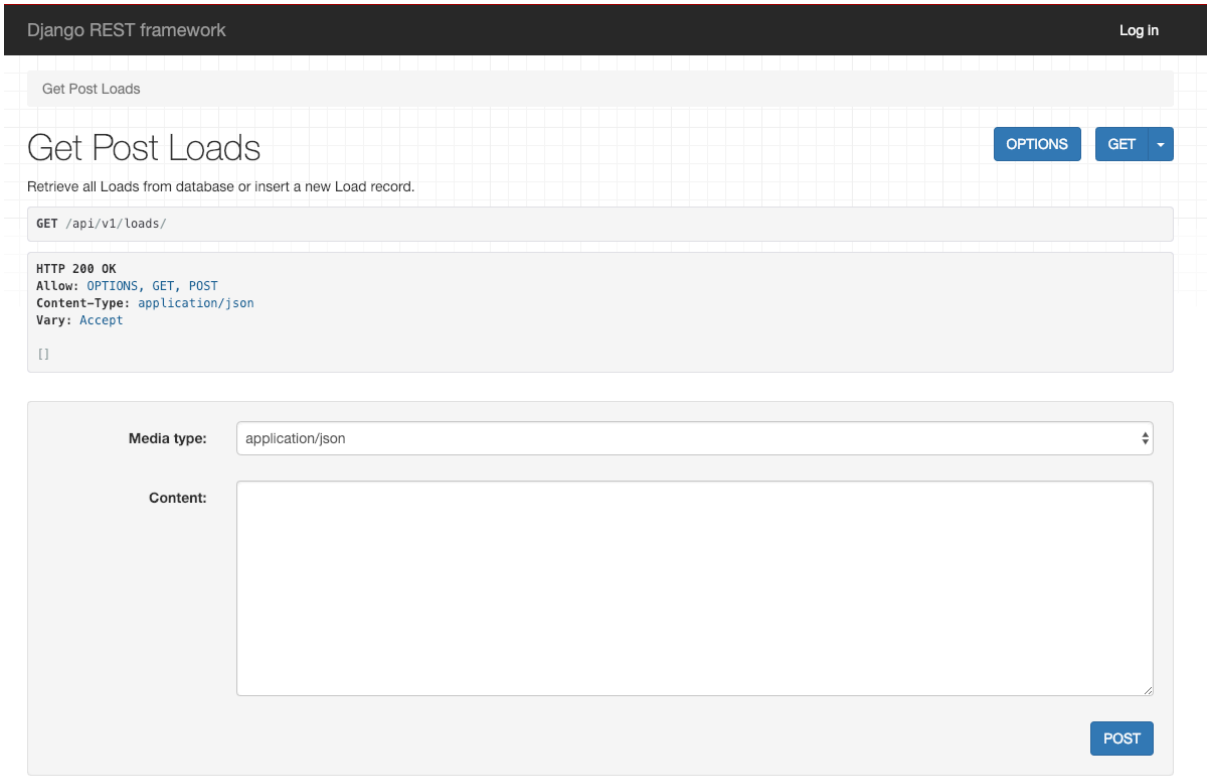
Endpoint	HTTP Method	CRUD Method	Result
api/v1/loads	GET	READ	Retrieve all loads
api/v1/loads/:id_load	GET	READ	Retrieve a single load
api/v1/loads	POST	CREATE	Insert a single load
api/v1/loads/:id_load	PUT	UPDATE	Update a single load
api/v1/loads/:id_load	DELETE	DELETE	Delete a single load

After that, I'm going to create a specific endpoint in the API to map the loads to trucks in order to optimize the entire route. This last endpoint will not respect the REST concept of making APIs but since it's just for creating a start point in the script (and later showing the results), I think that should not be a problem. It's just like running a Python script from the Terminal in order to get the expected results, but from a web server.

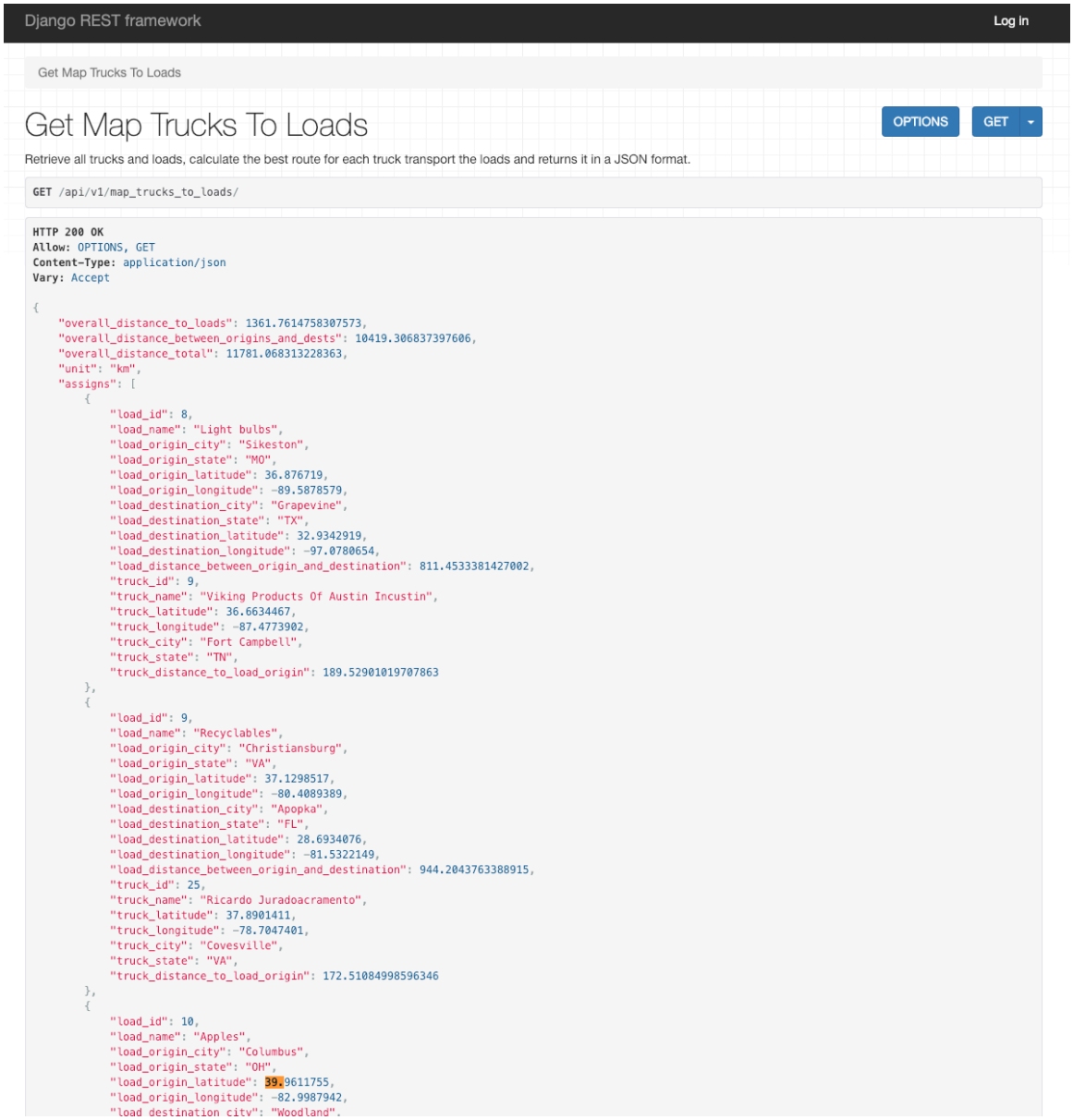
Solution

I've created the RESTful API for trucks and loads(cargos) and you can find it inside the "*django-truck-load*" folder. There's a readme file called readme.md inside of each important folder of the project. After following these instructions in order to start, test and run the server, you can access it through your web browser going to http://127.0.0.1:8000/api/v1/loads/ or http://127.0.0.1:8000/api/v1/trucks/ . In these pages you can find a small documentation of the API related to each endpoint. Just for reference, these are the documentation pages:





In order to check the final solution (available in *views.py* file), just make a single request for http://127.0.0.1:8000/api/v1/map_trucks_to_loads/ and you will see something like that:



After a lot of tests, I've seen that the method used to calculate the distance between two geographical coordinates is not quite accurate compared to the distances shown through Google Maps / Google Maps API. In a future version, that can be made in order to get better and more accurate results.

References

- https://mitsloan.mit.edu/ideas-made-to-matter/creating-better-bus-routes-algorithms
- http://depts.washington.edu/ddi/publications/ictd_2009_starbus_final.pdf
- http://andromeda.rutgers.edu/~jy380/research/trb/trb.pdf

- https://stackoverflow.com/questions/30706799/which-model-field-to-use-in-django-to-store-longitude-and-latitude-values
- [1] https://www.darrinward.com/lat-long/