**Cargo Logistics Project**

**By- Gunasekhar and Radhika Jade**

**Date: May 18, 2021 17:37 IST**

Task 1: Loading Data Set – from the given world cities.csv and generate planes and cities collection

Task 2: Developed all the endpoints (such as /cities, /planes, /cargo) and few endpoints for plane/cargo metrics. Following historic records are captured:

* Average time taken for the parcel/cargo drop,
* Total number of drops/pick-ups by each plane,
* Number of route cities that the parcel/ cargo has traversed before reaching its destination,

Task 3: Implemented an approach for aircraft(plane) to collect parcels(cargoes) and deliver them to their destination.

-> Shortest distance approach (routing.py)

1) Shortest Distance:

1. We have change\_stream.py script will be running continuously and watching over Cargos collection.
2. Whenever a parcel gets created ( insert operation performed in cargos collection), change\_stream.py script will call add\_route() function.
3. It will calculate the distance (RD) of each plane that will travel by adding distances between each city in route field of plane. Distance calculation is done by using coordinates (longitudes, latitudes)

RD = sum of distances between all cities in plane

1. Add plane to parcel (cargo) distance (PTP- parcel to plane distance) to above distance (RD)

Total\_Distance = RD + PTP

1. Sort the total distances (Total\_Distance) in ascending order.
2. Identify and assign the plane (ID) that is nearer to the parcel /cargo and this will take the shortest distance taking less time.
3. Along with adding destination to plane will add courier to a parcel so that whenever a plane landed will load all the parcels that are assigned to plane.

Here ‘logistics database’ contains following collections and schema.

Plane

{

"\_id": "CARGO1",

"currentLocation": [ 10.2, -18.3],

"heading": 165,

"route": ["London","Paris"],

"landed": "Madrid",

}

City

{

"\_id": "Paris",

"country": "United Kingdom",

"location": [ 52.0, -0.1 ],

}

Cargo

{

"id": "ab02b4029392ncalksjfo",

"destination": "Paris",

"location": "Berlin",

"courier": "CARGO1",

"received": "2020-06-15T11:34:05Z,

"status": "in process" OR "delivered",

"delivered\_time":time,

"Origin": "Harare"

"history": [

]

}

Make sure you create required indexes in Cargo collection.

db.cargos.createIndex({'location':1,'status':1})

PlaneHistory

{

"\_id": "60a012288e34fda6cc92c13e",

"plane": "CARGO17",

"location": "Madrid",

"status": "in process",

"cargo\_id": "609ffbdf87073877450b6a1d",

"operation": "drop",

"date": "2021-05-15T23:55:44.862Z"

}

Make sure you create required indexes for PlaneHistory collection

db.planeHistory.createIndex({"plane":1,"date":1})

`Deltime` collection contain following structure.

{

"\_id" : 7,

"delivery\_time" : ISODate("2021-05-18T15:14:08.101Z"),

"delivery\_sum" : 0.283479212,

"cargo\_id" : "60a38ae6e704c461fa86846f"

}

“ID” itself delivered parcels count.

"""

**Task 3: Additional stats – Historical records**

Following are the historical records we are extracting from the available plane/cargo/city information:

**Parcel / Cargoes Stats:**

1. Average time taken by each parcel.

**URL: /cargoes/avg\_delivery**  **method: ‘GET’**

**>>> curl** [**http://localhost:5000/avg\_delivery/**](http://localhost:5000/avg_delivery/)

1. Average time taken by each parcel from origin to destination.

**URL: /cargoes/avg\_time** **method: ‘GET’**

**>>> curl** [**http://localhost:5000/cargoes/avg\_time**](http://localhost:5000/cargoes/avg_time)

**Plane / career related stats:**

1. Number of drops done by the plane each day.

**URL: /planes/<plane\_id>/profile/drops**   **method: ‘GET’**

**>>> curl** [**http://localhost:5000/planes/CARGO10/profile/drops**](http://localhost:5000/planes/CARGO10/profile/drops)

1. Number of pickups done by the plane each day

**URL: /planes/<plane\_id>/profile/pickups**   **method: 'GET’**

**>>> curl** [**http://localhost:5000/planes/CARGO10/profile/pickups**](http://localhost:5000/planes/CARGO10/profile/pickups)

1. Number of distinct/unique cities covered in each day.

**URL: /planes/<plane\_id>/profile/cities**  **methods: ‘GET’**

**>>> curl** [**http://localhost:5000/planes/CARGO10/profile/cities**](http://localhost:5000/planes/CARGO10/profile/cities)

1. Historic records of date/ time stamping pertaining to each planes’ profile/details (e.g., Plane ID, location, routes etc.,).

**URL: /planes/profile**  **method: ‘GET’**

**>>> curl** [**http://localhost:5000/planes/profile**](http://localhost:5000/planes/profile)