Connected World

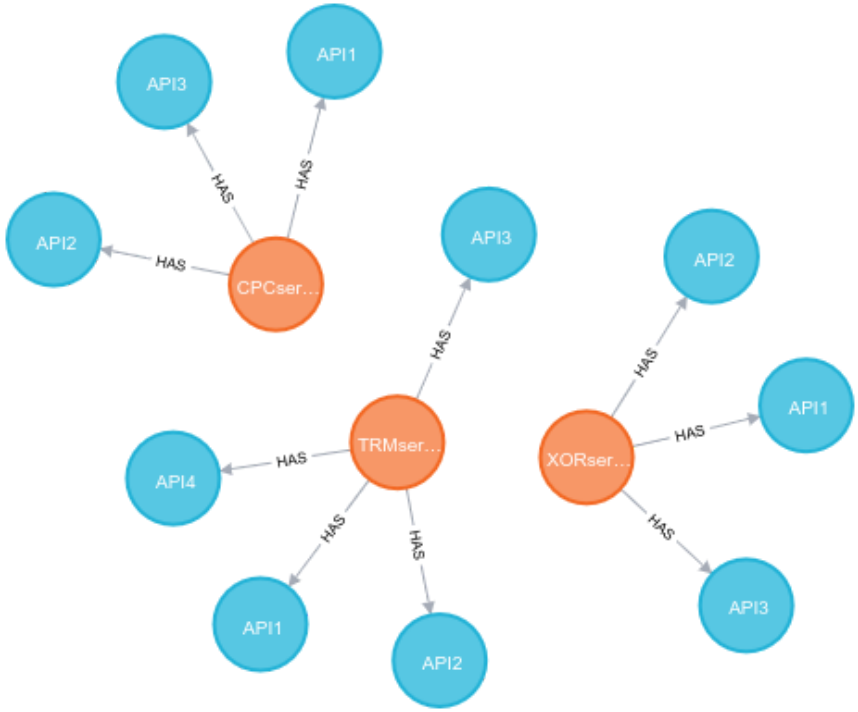
Rudrajit Dawn

Requested Reviewers: Hemant Suthar, Nitish Garg, Manu Agarwal

Description: Project during Amazon Internship (27-04-2020 to 19-06-2020)

Problem Statement:

First let's discuss the problem statement. So, in a system, there are multiple services and each service consists of multiple APIs like the picture below.



Here TRMservice has 4 APIs, XORservice has 3 APIs, CPCservice has 3 APIs. Also there are some relations between the nodes, like one call to one API will result in one or multiple calls to other APIs. We will call this kind of relation "DEPENDENCY", and this kind of relation has an attribute "call_rate". Let's look at the picture below.

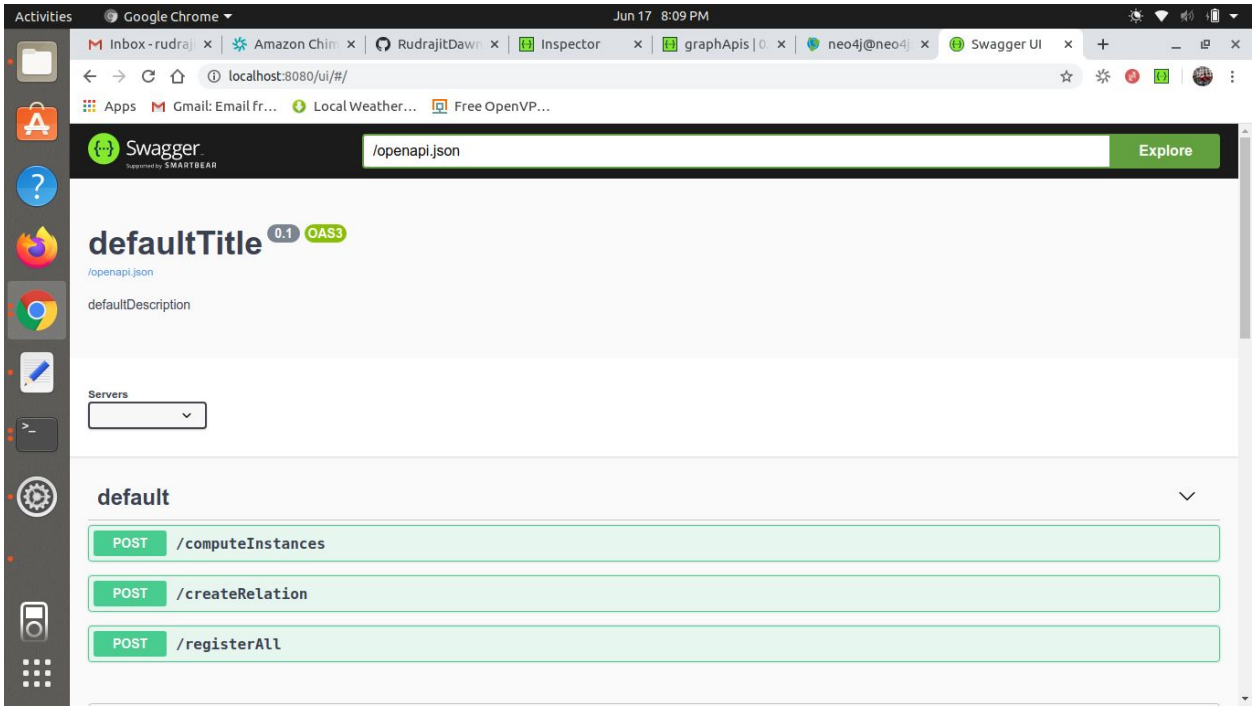
So this is how the graph will look like after creating relations successfully. After creating the relations we will provide the load or “hit_rate” to some APIs in the graph and will see the impact of it in the whole graph. Also we should have the flexibility to control how far from the starting nodes the calculator will follow the relations. One attribute is dedicated to this, named “depth”. We will explore more about these things.

Reason for selecting neo4j as Database:

Actually neo4j is more specific and optimized for graph operations, like creating new nodes of different types, and creating relations between nodes, making query and updating. It actually uses cypher query language whose working style is different from SQL. It is actually very good in matching some patterns and after matching it takes required action in those places. It can very easily fit itself in an environment where schema is dynamic.

Step by step approach and screenshots:

To use the APIs I have created, there is one interactive and easy to use platform, Swagger UI. It will look like this.



So here we can see three APIs are there, all are for POST requests.

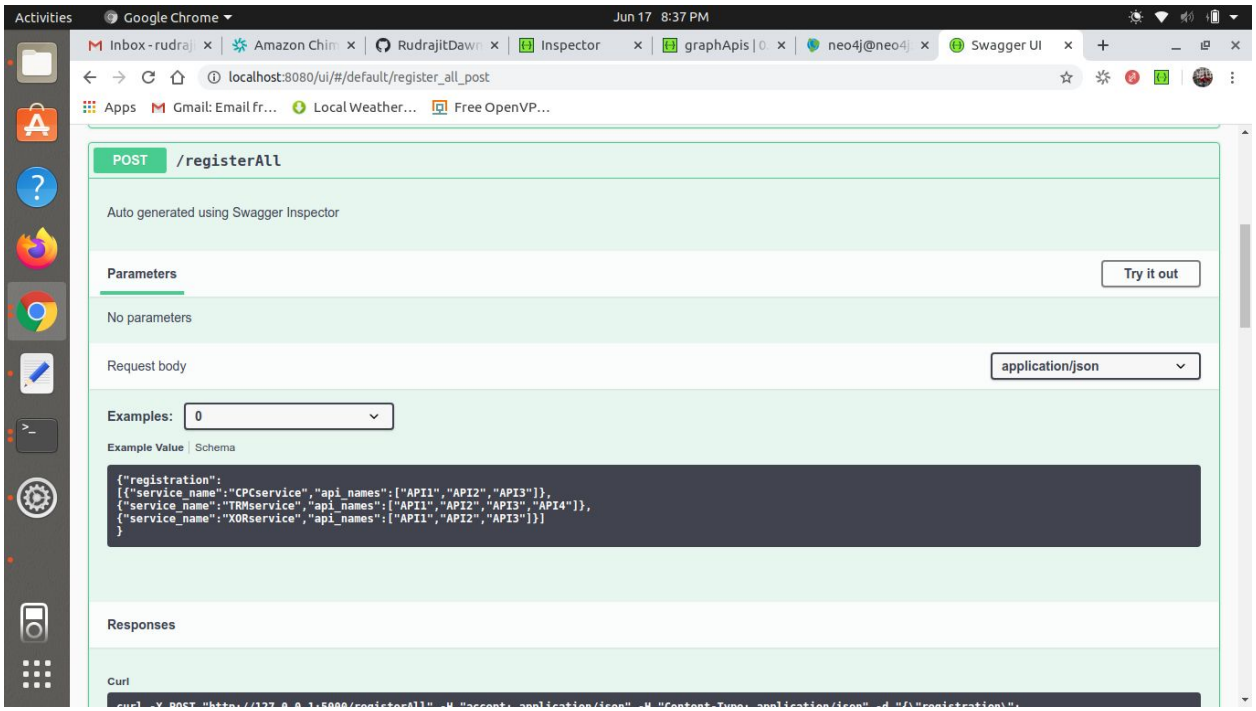
Reason for making a separate API for registration:

One thought may come in mind that everything we could do in a single shot like we could register services, APIs while creating relations. Then users could do those with less effort. But the problem is that if someone made some spelling mistake while writing service or api names in a relation (for example someone wrote “TRservice” in place of “TRMservice”) then wrong registration would happen and its impact on database would be very bad. So there is a separate API for registration and we are assuming that users will be serious enough while registering service and APIs.

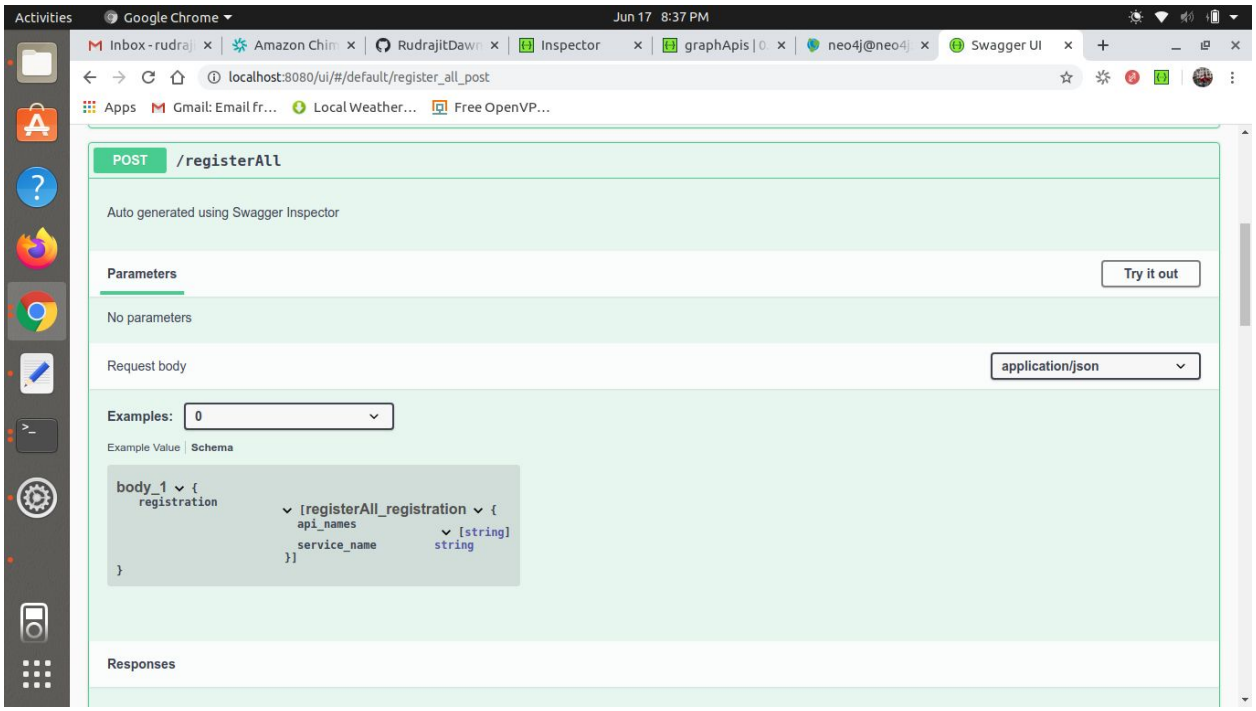
Now let's look at different APIs and their input, output schemas and examples.

API for registration of services and APIs:

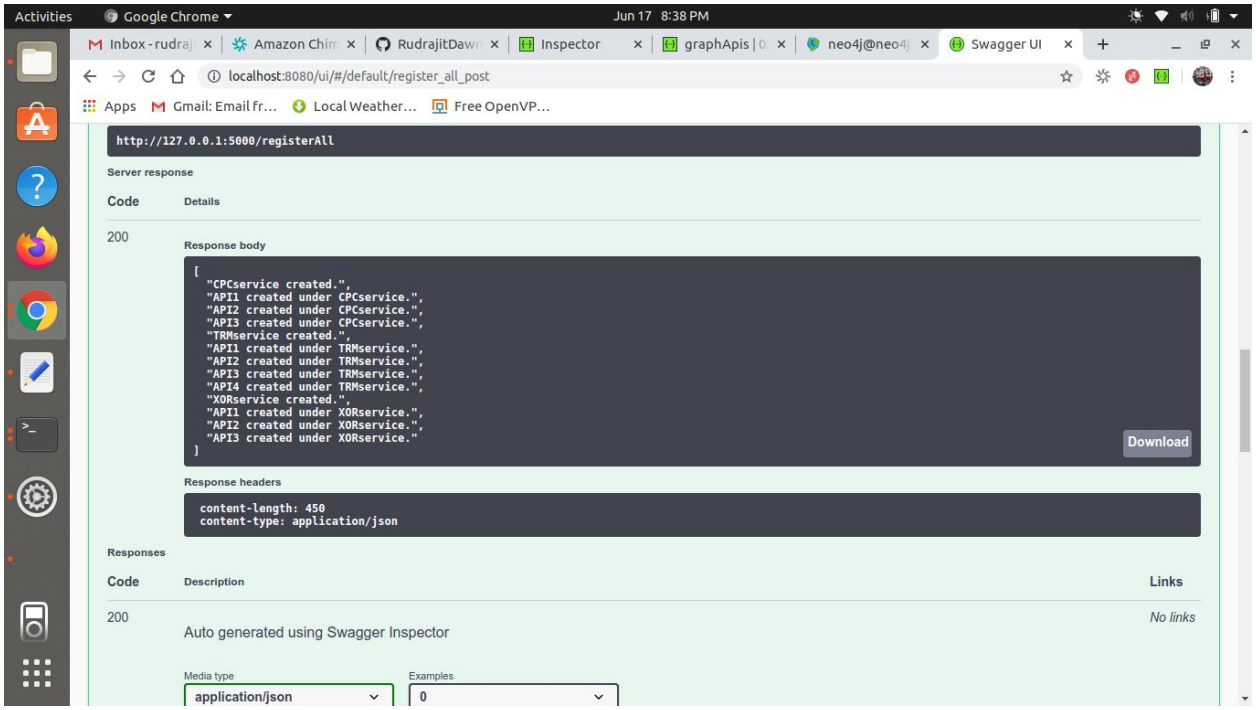
Let's look at the picture below.



This is an example input for registration API.

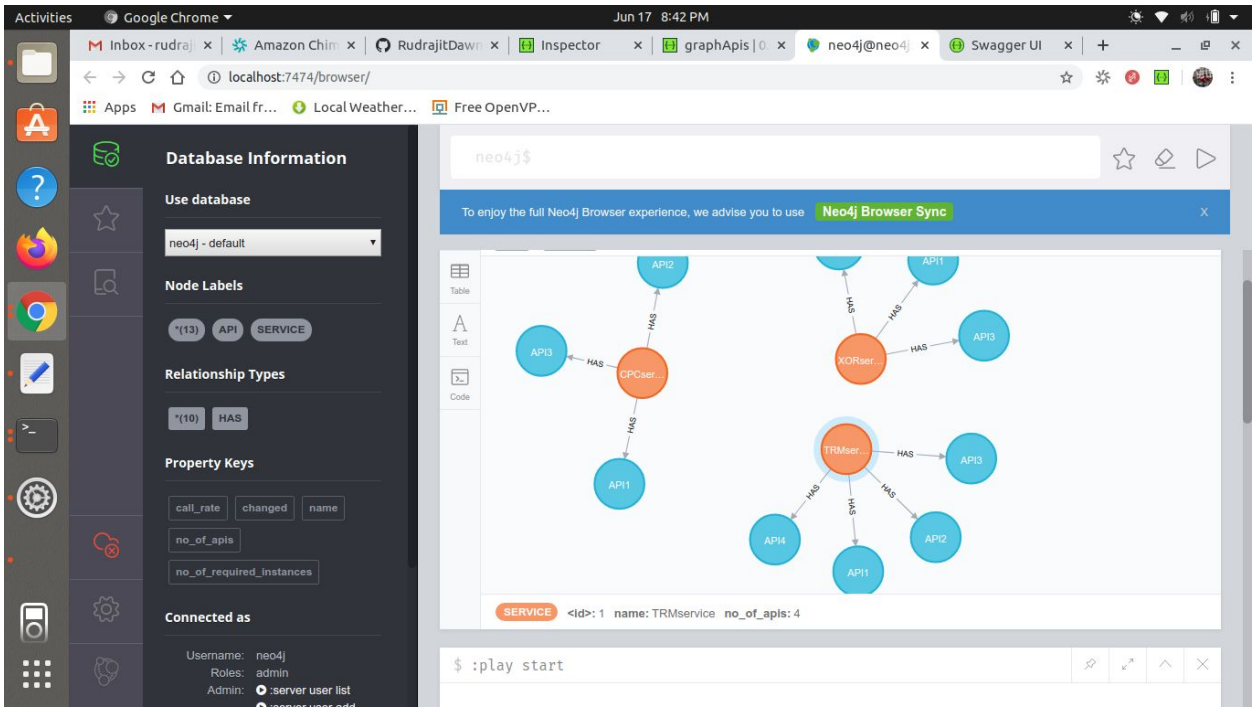


The above picture is for the schema of registration input.



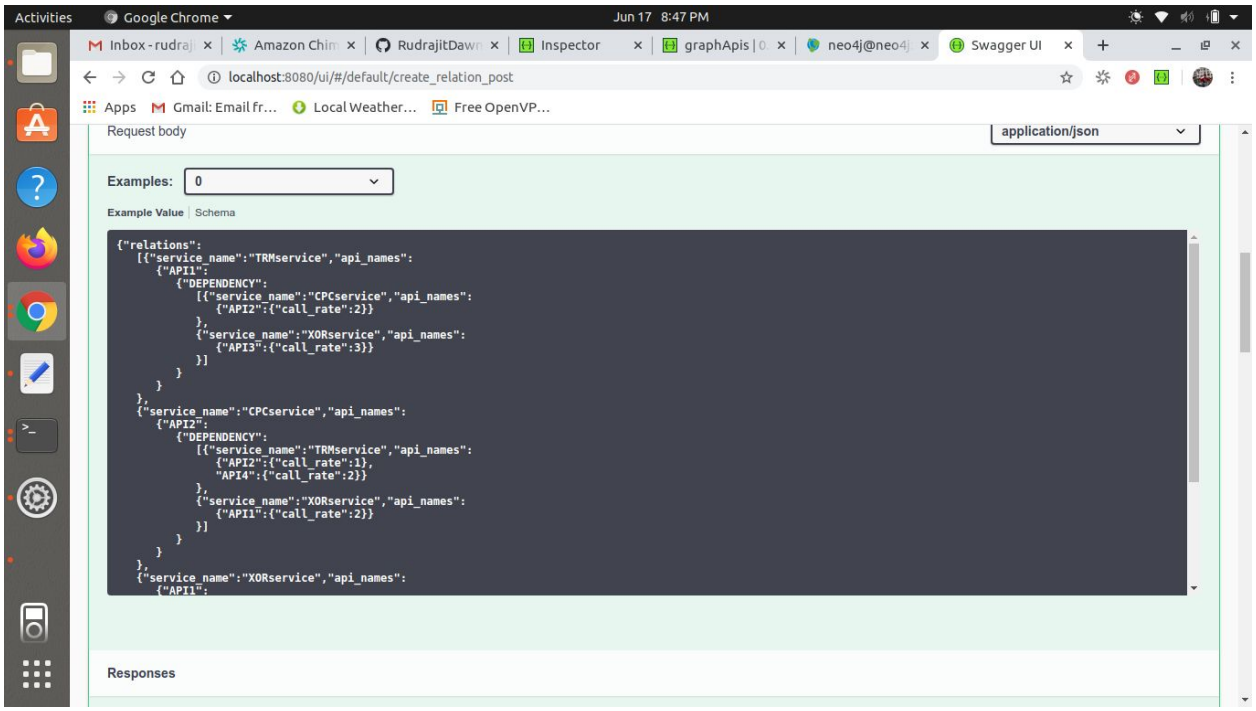
This is an example output for registration API.

We can check the database after this operation. It will look like this:

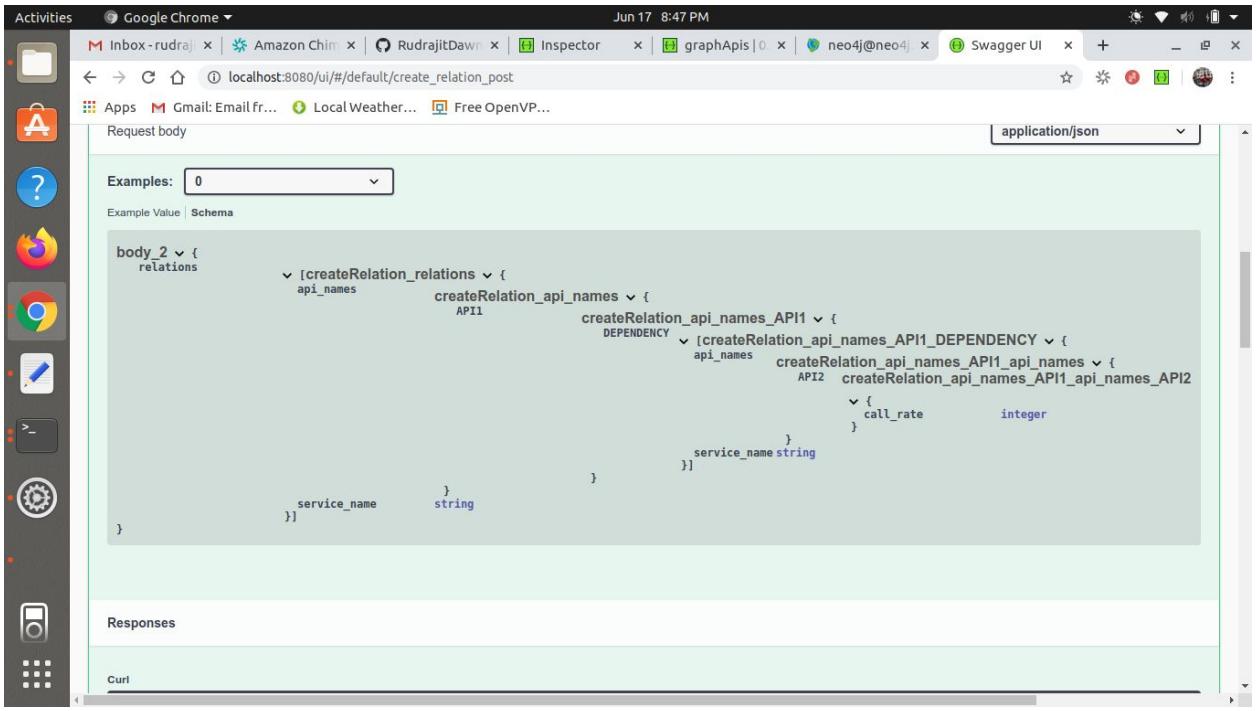


API for creating relations:

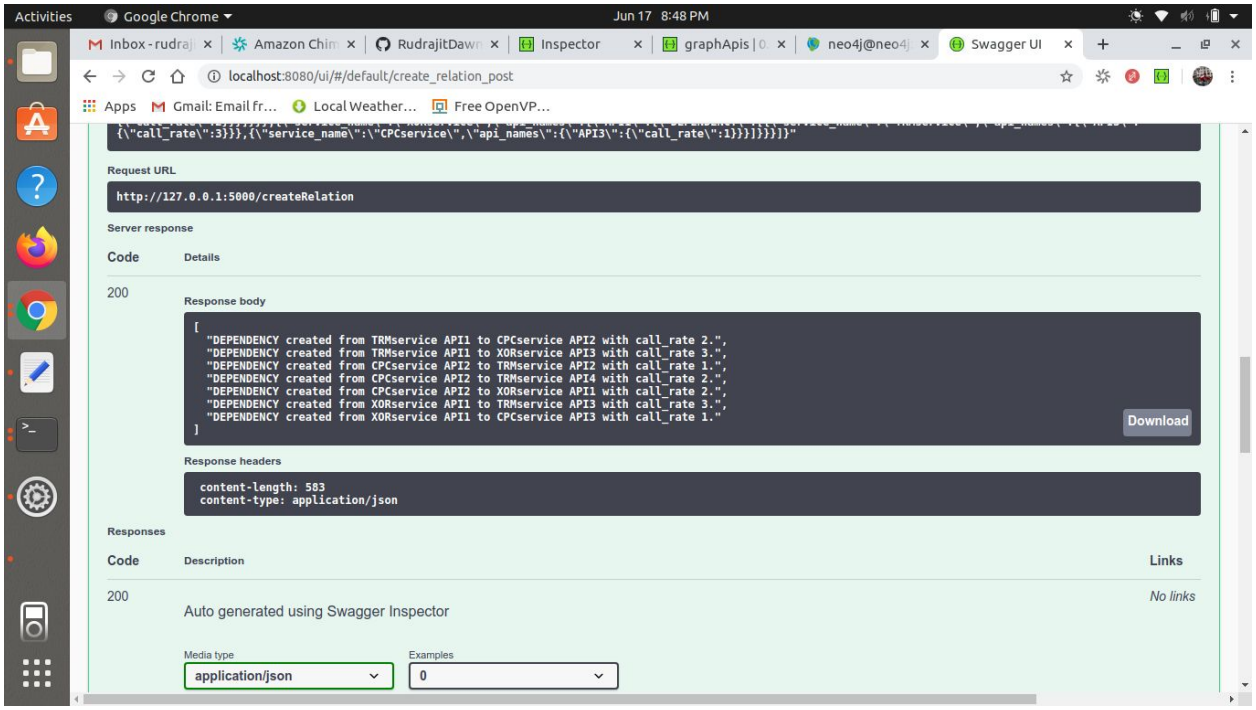
Let's look at the picture below.



This is an example input for the API responsible for creating relations.

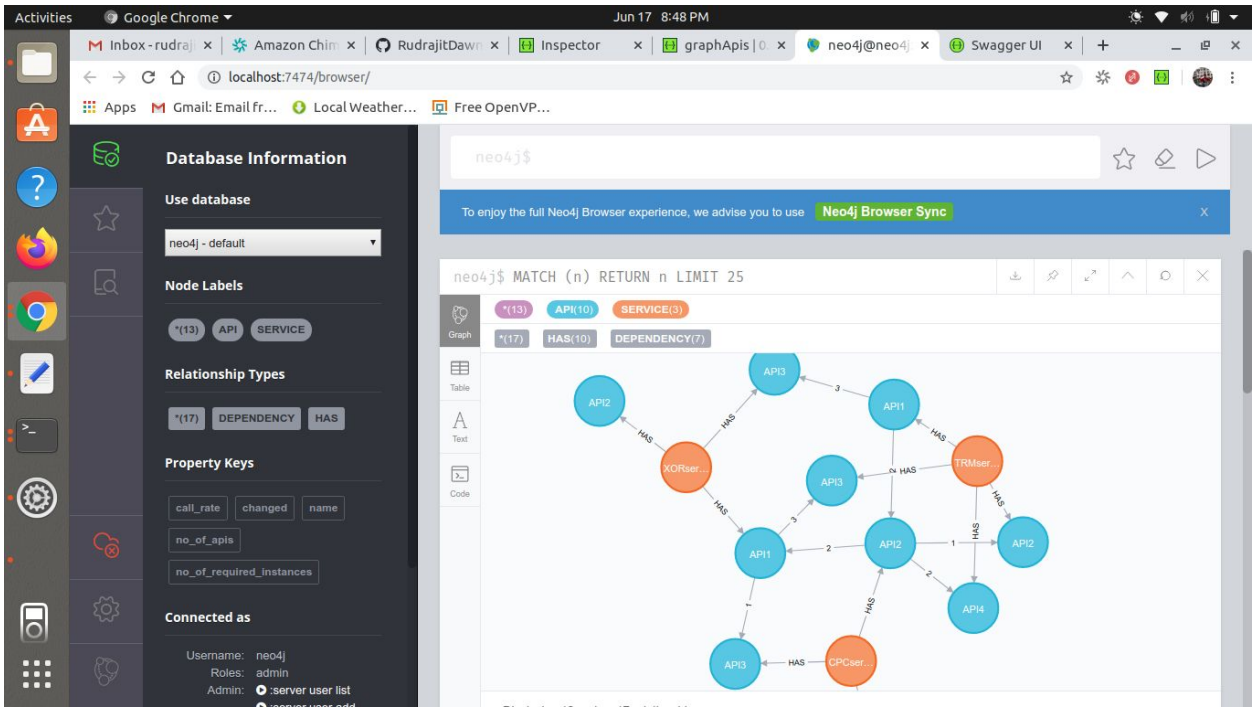


The above picture is for the input schema of the API responsible for creating relations.



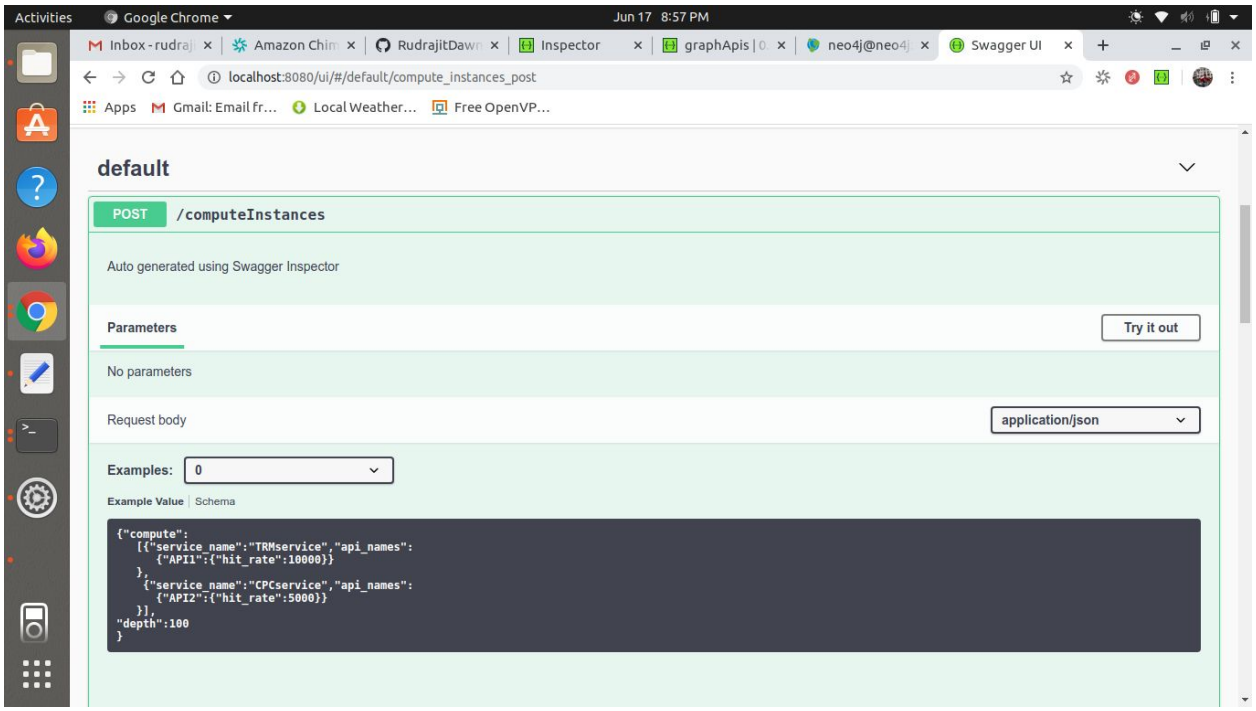
This is an example output for the API responsible for creating relations. These are just acknowledgements. In case of any error, it will give proper error messages.

We can check the database after this operation. It will look like this:

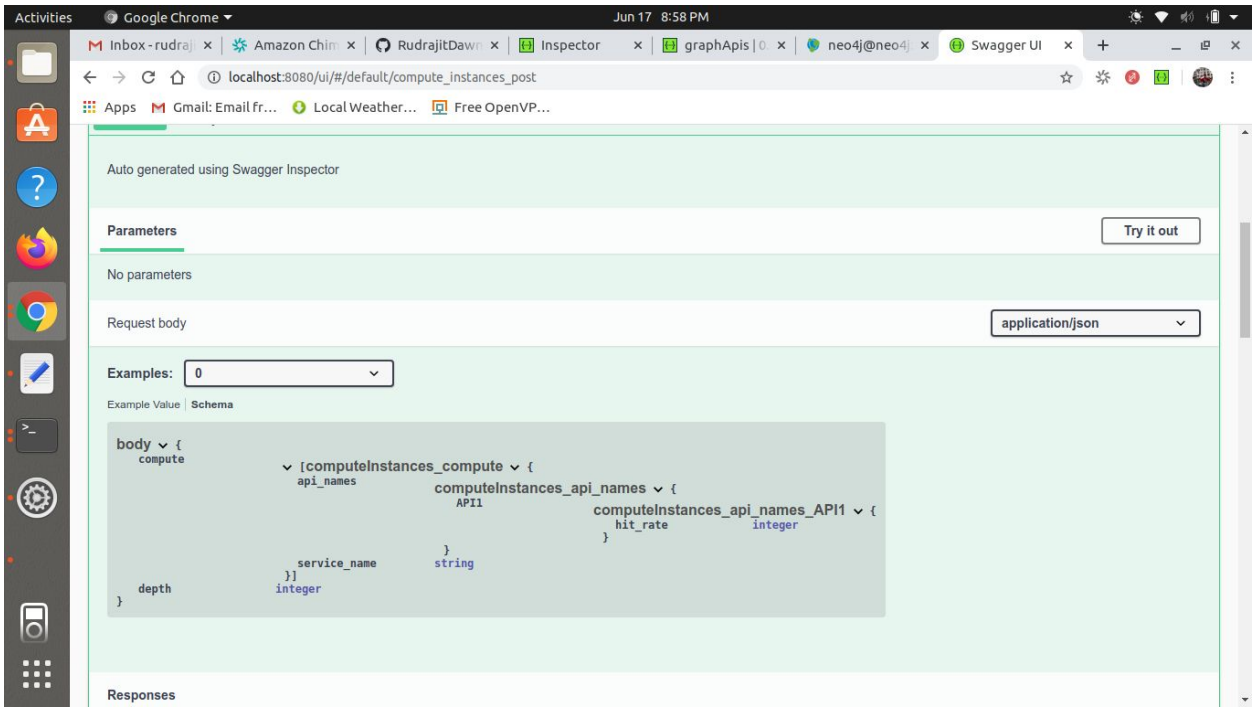


API for computing instances:

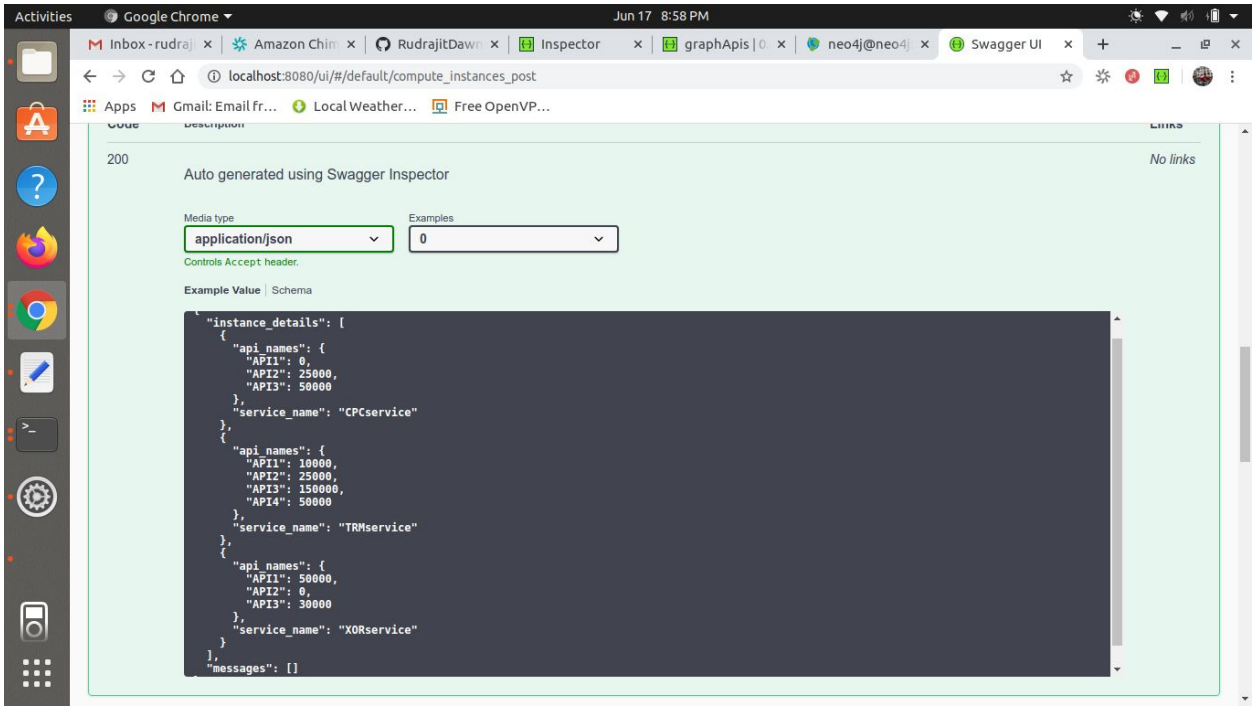
Let's look at the picture below.



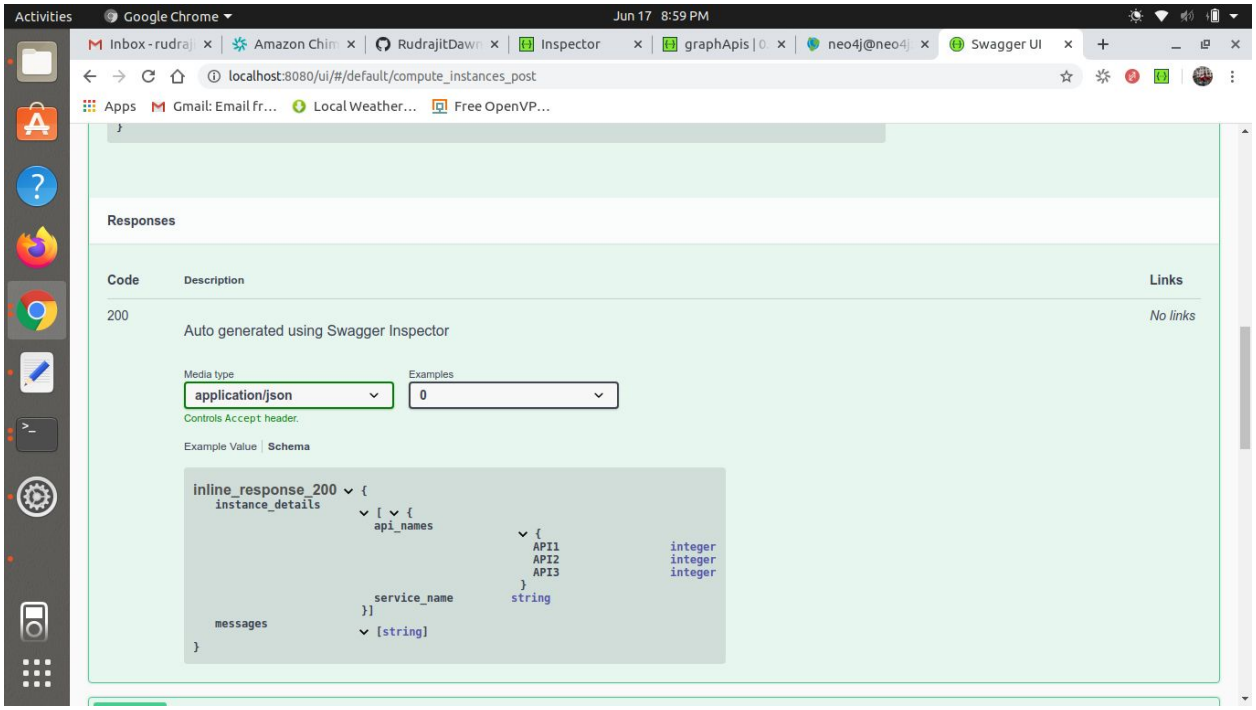
This is an example input for the compute API.



The above picture is of the schema for input in compute API.



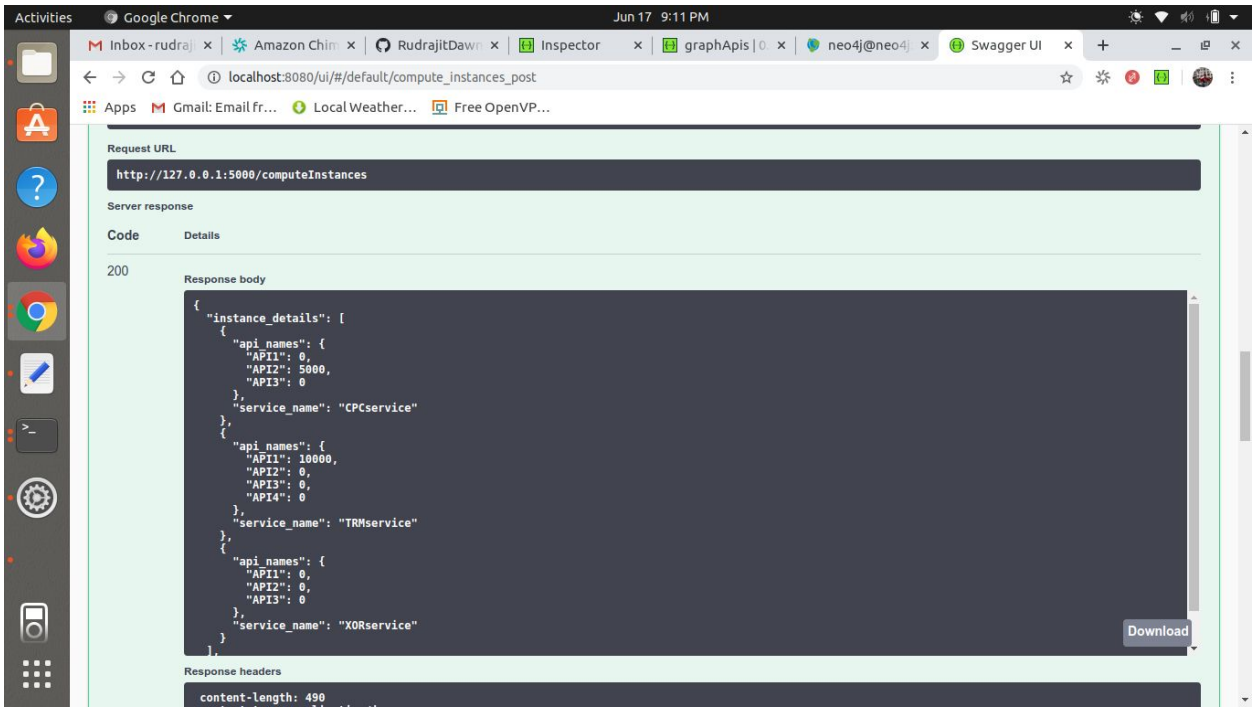
This is an example output for the compute API.



This is the output schema of the compute API.

Now there is one interesting attribute in input, “depth”. Let’s discuss a little about that attribute.

It actually controls how far the calculator will follow the relations from starting nodes. If it’s value is 0, it will follow no path. I am showing the output for that scenario.



So we can see here only two values are greater than 0 and they were the only ones present in input. So the calculator followed no path.

If one wants to traverse the whole path, till the end, he/she can give some large value like 100.

One more thing, there is no separate API for update. One can create new services, apis, relations any number of times using the existing APIs.