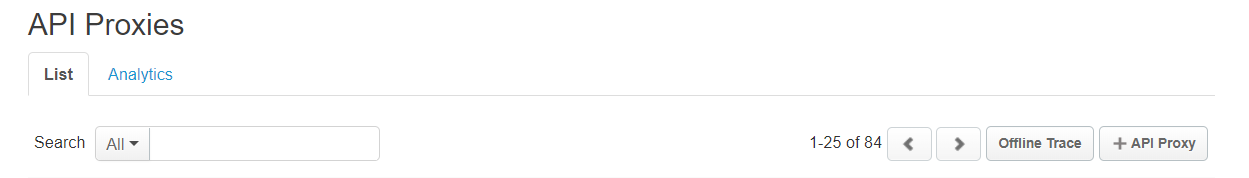
* **OBJECTIVE:**

**IN APIGEE EDGE WE ARE EXPOSINGG MICROSERVICES or REST API BY USING NODE JS AS A SERVER SIDE LANGUAGE AND MONGO DB AS A DATABASE BY GIVING THE API (DESIGN) TEMPLATE WITH SWAGGER (OPEN API) AND USE THAT SWAGGER IN DEVELOPER PORTAL TO GET THE STRUCTURE OF YOUR API WITH NODE JS BACKEND SO IT WILL GIVES THE LIVE OVERVIEW OF YOUR COMPLETE REST API.**

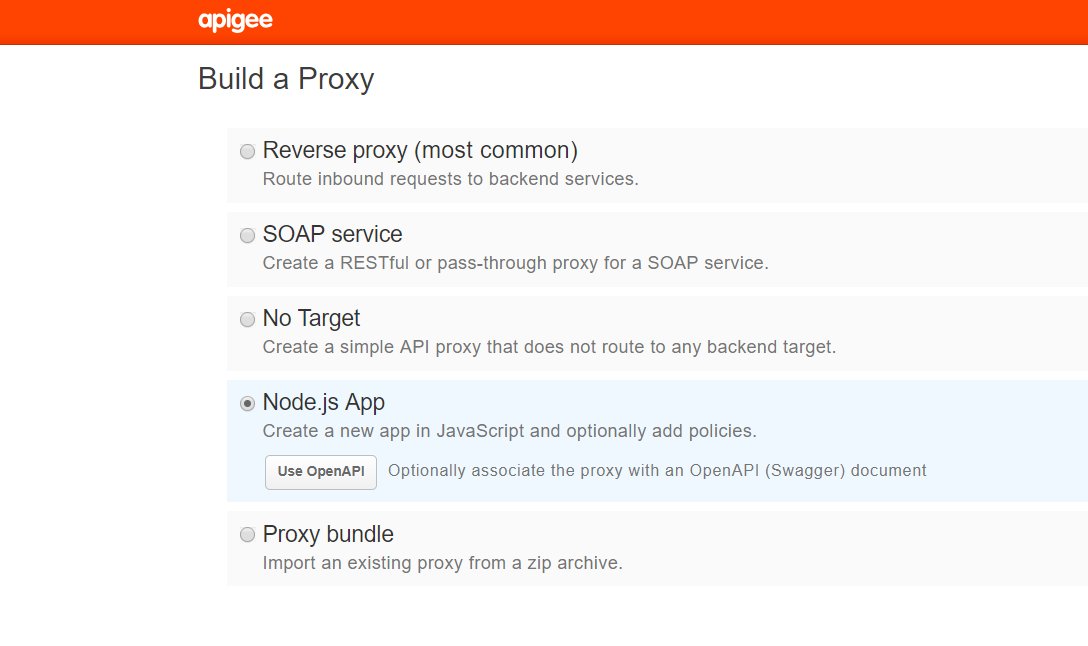
**BLOG 1:**

**EXPOSING MONGO DB AS REST API FROM APIGEE:**

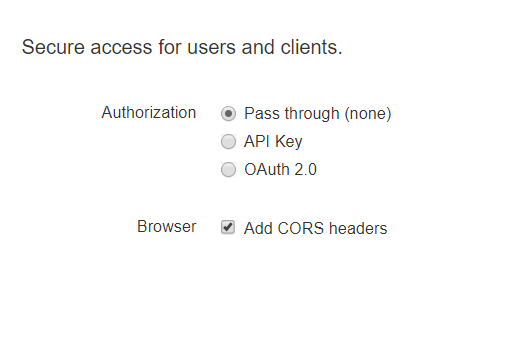
* **We are creating a simple REST API with express node js framework using HTTP methods get,post,put and delete.**
* **We are using cloud mongo db database i.e mlab with mongoose module, so we can perform our CRUD operations through apigee edge.**
* **We deployed the REST API on Apigee Edge , calling the api through apigee edge while using the cloud database.**
* **After that we’ll introduce swagger, an open source project that plays a big role in Apigee and provides tools for building APIs with Node.js and Swagger.**
* **Steps to create:**
* **Click on Add api proxy**



* **Selecting node.js app**



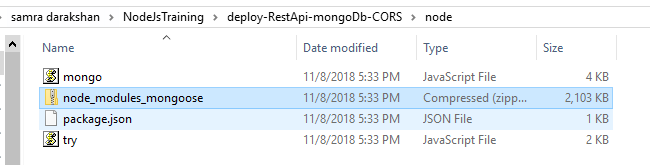
* **Selecting add CORS here for browsers headers**

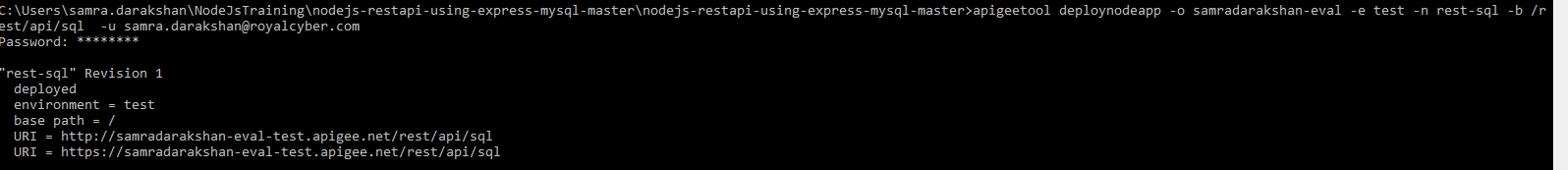


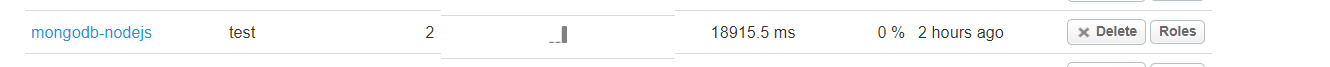
* **Enabling CORS**
* Before publishing your APIs, you'll need to enable CORS on your API proxies to support client-side cross-origin requests.
* CORS (Cross-origin resource sharing) is a standard mechanism that allows JavaScript XMLHttpRequest (XHR) calls executed in a web page to interact with resources from non-origin domains.
* **You have to install mongoose on local machine first.**



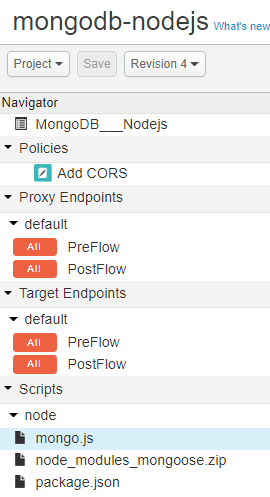
* **After doing this, mongoose added into the node modules folder.**



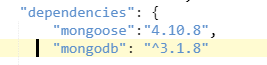
* **Deploy your api on apigee by the following command**
* apigeetool deploynodeapp -o **org** -e **env** -n **proxy-name** -b /node-app -express -u **username**
* **write the above line on command prompt by substituting your values, after giving your username and password you will see your api deployed on apigee edge with all the modules you have installed on local.**



* **Here you can see the main file mongo.js, package. json and mongoose these are the files which we can find here after deploying.**



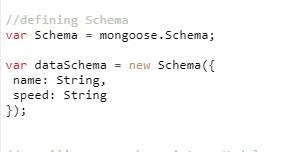
* **In Package.json file you have to give the dependencies of the modules which you are using in your api.**



* **We used mongoose module for connecting cloud database with apigee edge.**

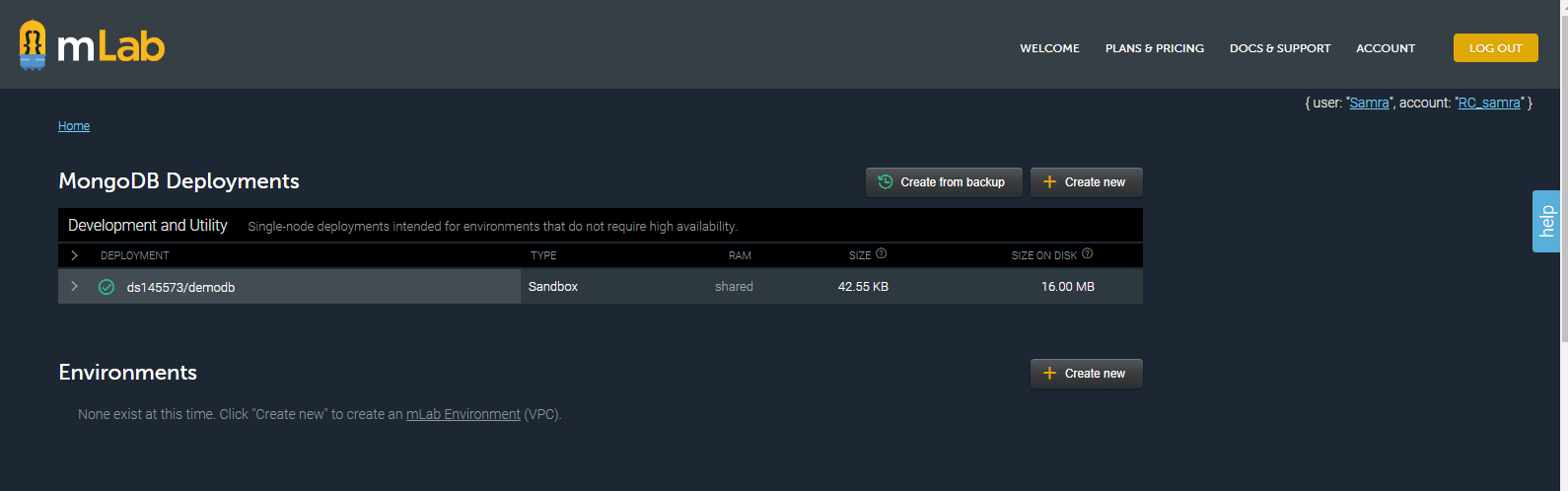
**-In mongoose create schema to fill the collections easily.**

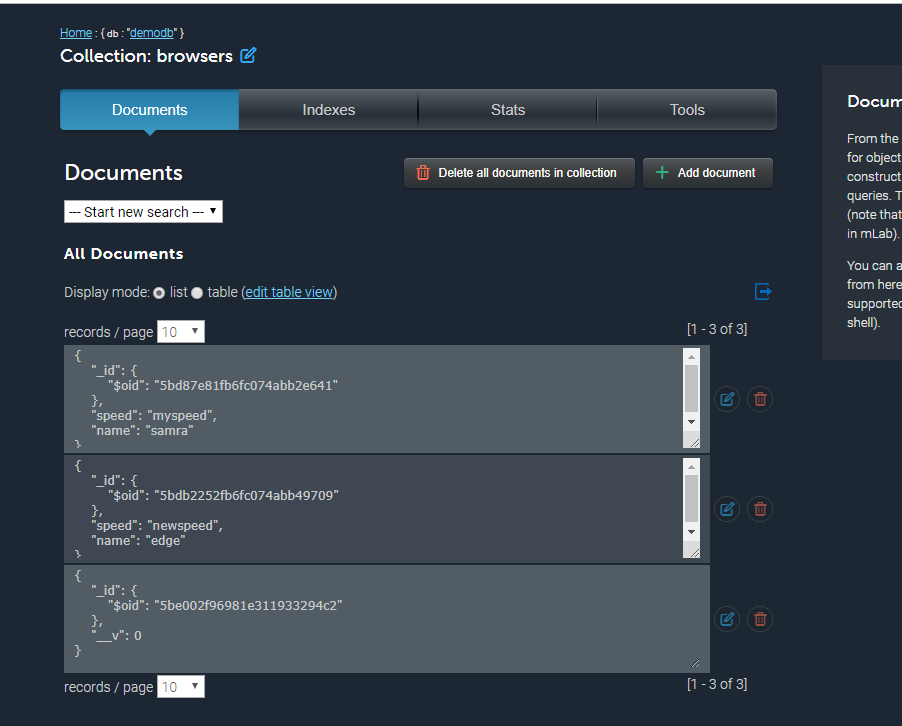
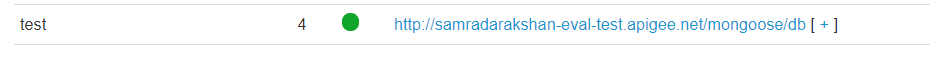




**Database:**

* **Mongoose** for **MongoDB**. **Mongoose** is an Object Data Modeling (ODM) library for **MongoDB** and Node.js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in **MongoDB**.
* **Creating database on cloud (mlab) for mongodb, it is a cloud database which is used for mongo db.**



* **Collections are just like tables in sql.**
* **While connecting your api from apigee edge you have to give the url which is on mlab**
  + **mongodb://<dbuser>:<dbpassword>@ds145573.mlab.com:45573/yourdbname**
* **By selecting your url you can perform your actions for the api.** 
* **Get our data from the database through apigee edge.**



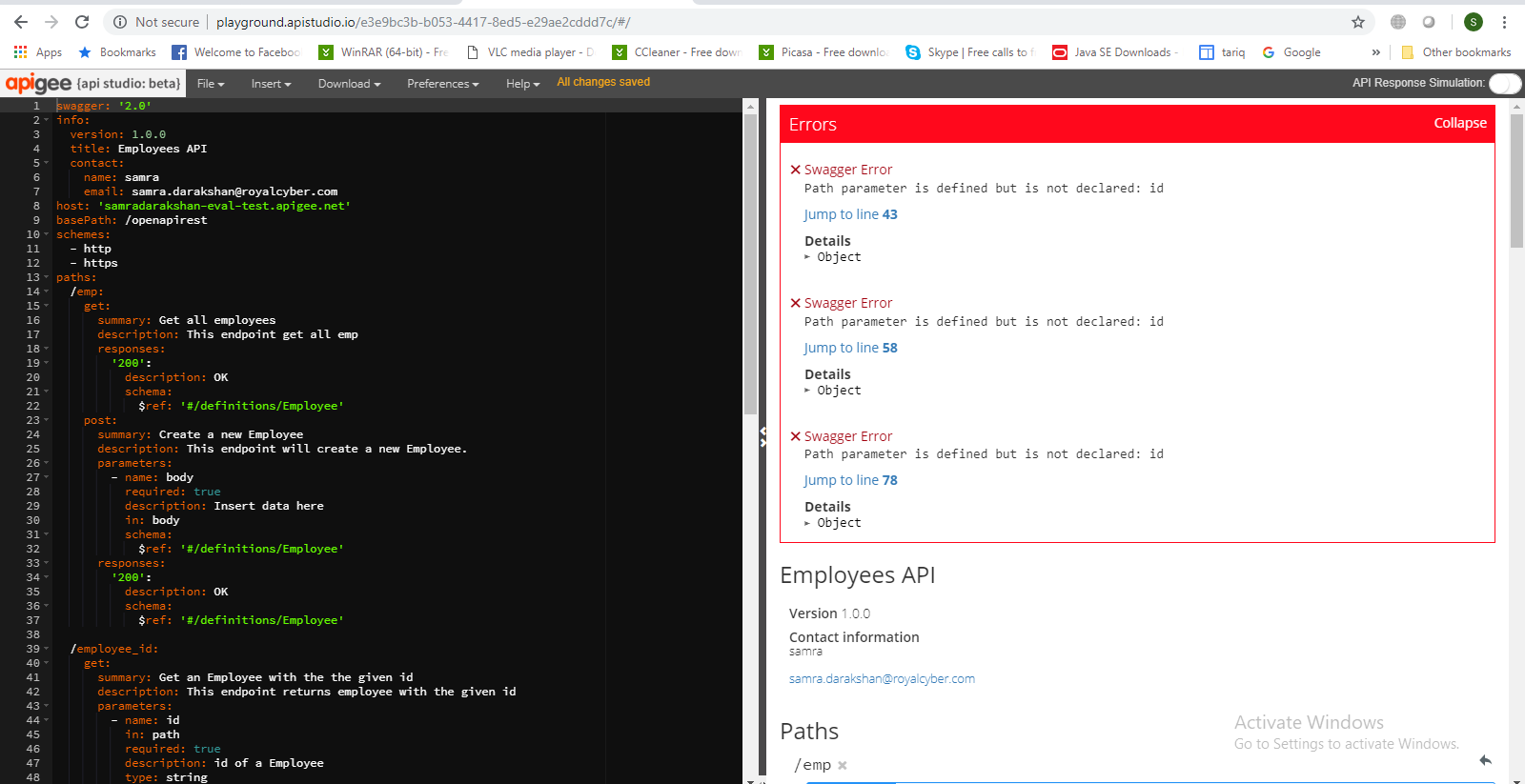
**BLOG 2:**

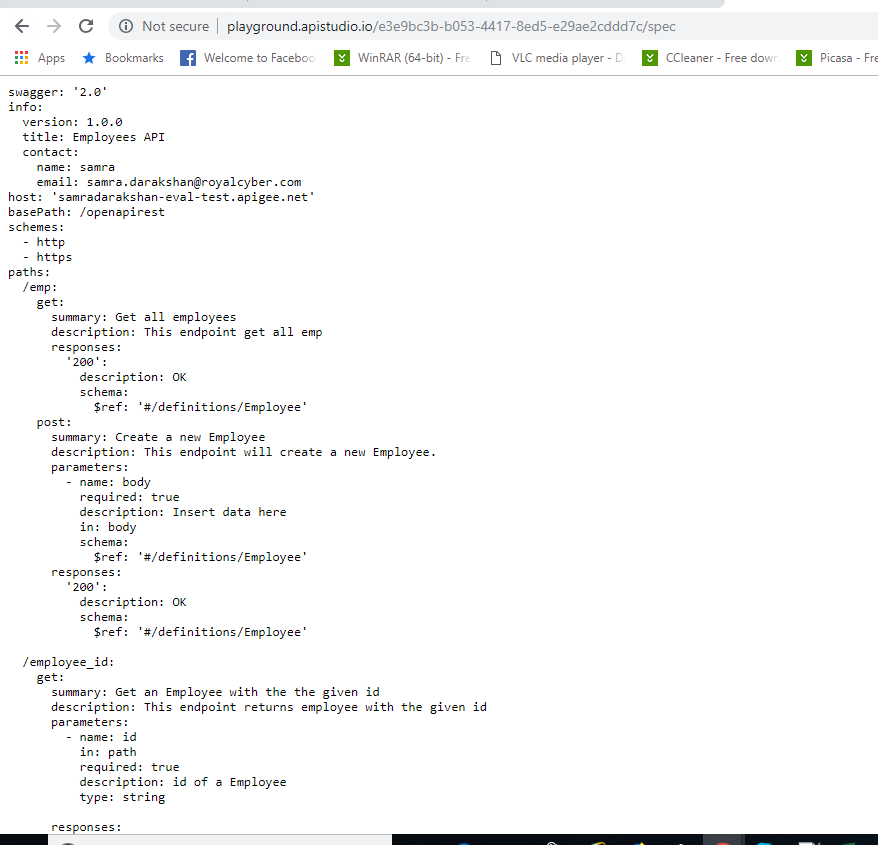
* **EXPOSING SWAGGER (OPEN API) AS REST API FROM APIGEE USING NODE JS MICROSERVICE AS A BACKEND:**

### **Swagger**

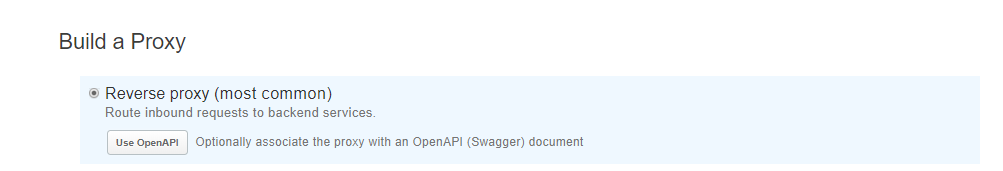
Swagger is a documentation format for REST APIs. Using Swagger, you can describe the available endpoints your API exposes, the contract/interface consuming those API endpoints, and more. In addition to providing a common format for describing REST APIs.

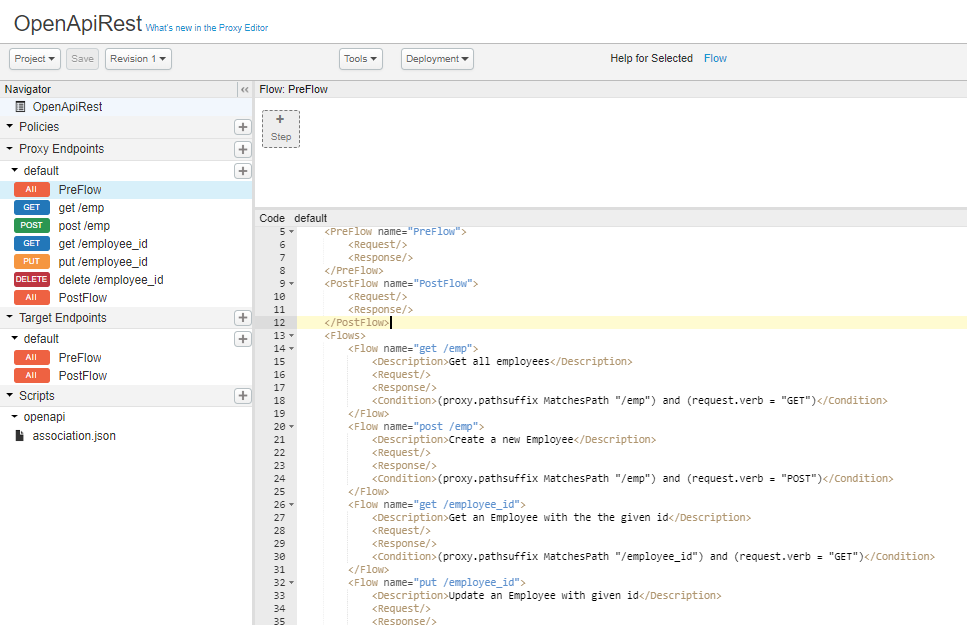
**Design your API:**

* **Now create swagger file (structure of Api) for the testing, here** [**http://playground.apistudio.io**](http://playground.apistudio.io)
* **In swagger file gives the host name of that node js micro service url that we have created before, it will use as a backend here.**
* **Make the url of the swagger file, add spec at the end of the url so it will show your complete open api, now we can use this url while importing the swagger file on apigee edge.**



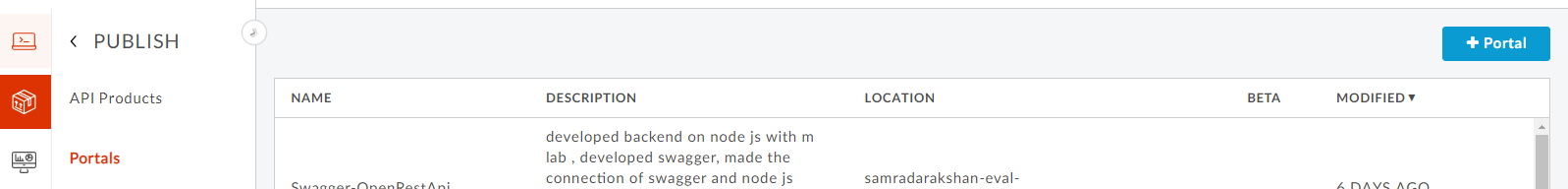
* **Creating new api proxy and use the above url of the swagger file here.**



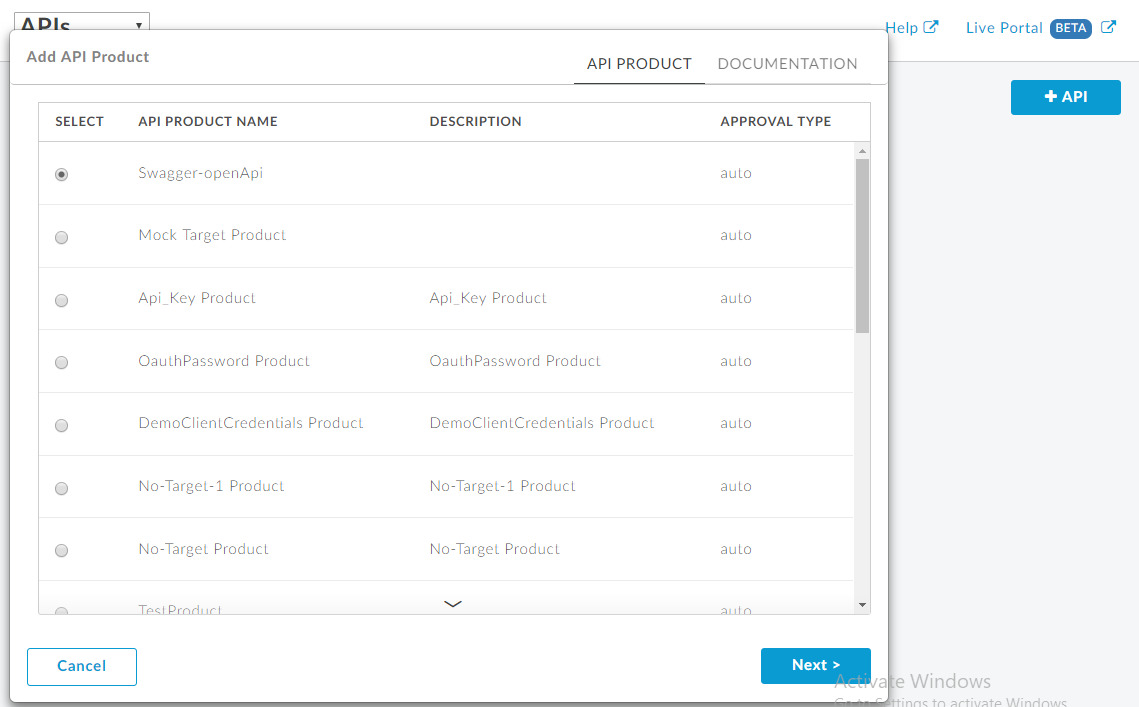
* **This proxy shows the structure of swagger file and hit the resource url as a backend.**
* **In the backend node.js micro service we used the same paths, methods that is using in swagger and save it in mongo db, data base on m lab like we have done before.**

**BLOG 3:**

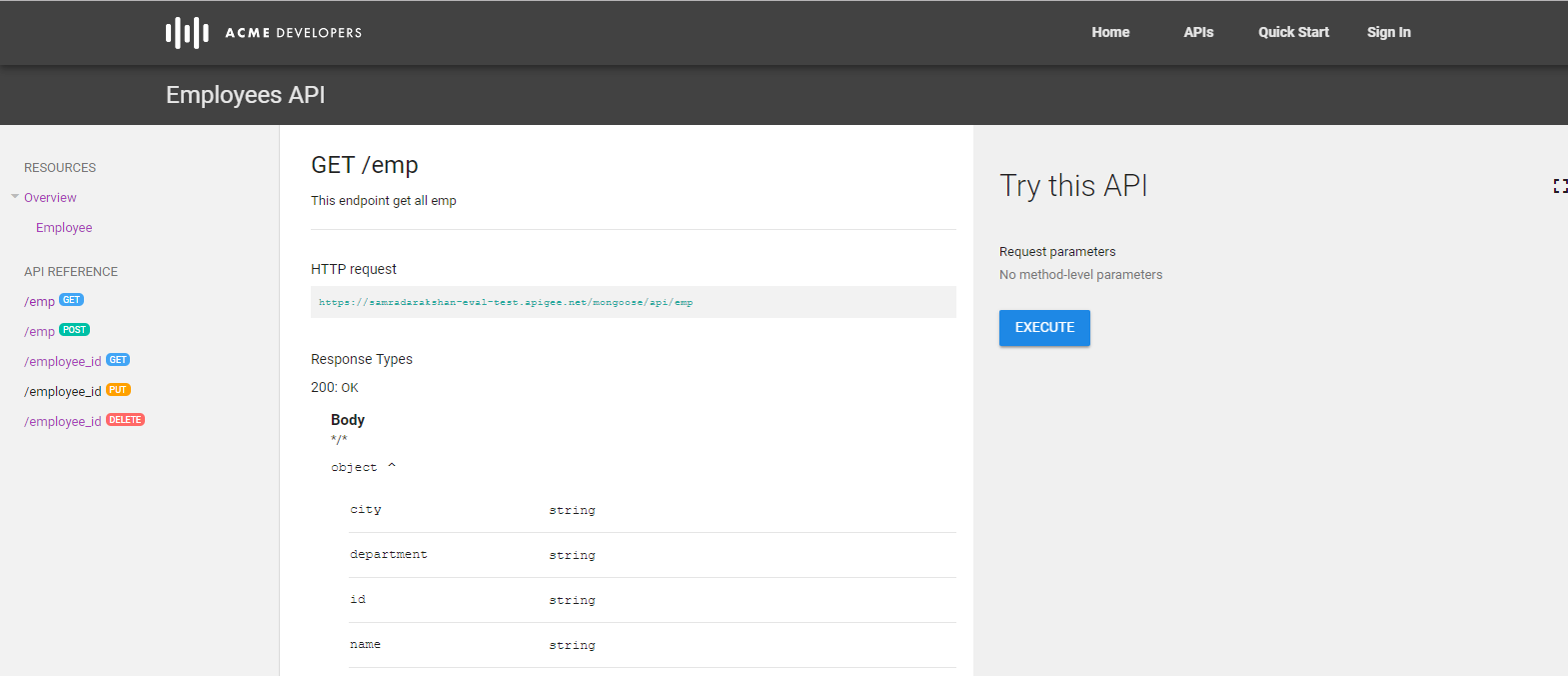
* **USING SWAGGER (OPEN API) IN DEVELOPER PORTAL FOR TESTING THE COMPLETE API:**
* **Go to the new edge, add new product and select your new api that you have created as an open api in the product.**
* **Add new portal.**

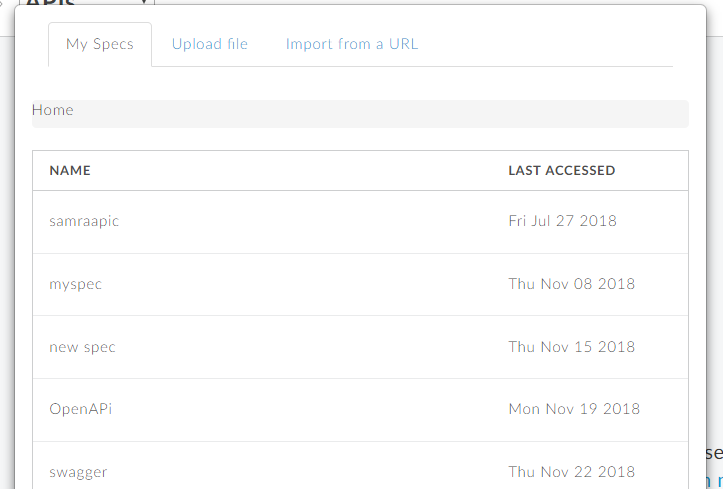


* **Now add apo in the portal and selecting the product that you have created for your api.**



* **Now you can add your spec file here or you can upload or put the url of the spec file that we have created before.**
* **Then see the live portal and click on api you can see your api and when you test it it will gives you the url of your main api so now you can test it by here.**
* **This is the structure of your complete api integrated with node js service**





* **When you gives the url while creating proxy your resource url should be the node js micro service and when you use your swagger in developer portal your http url can be change to the new url of the api proxy that have integrated the node js micro service, so it will works w.r.t your new api service.**