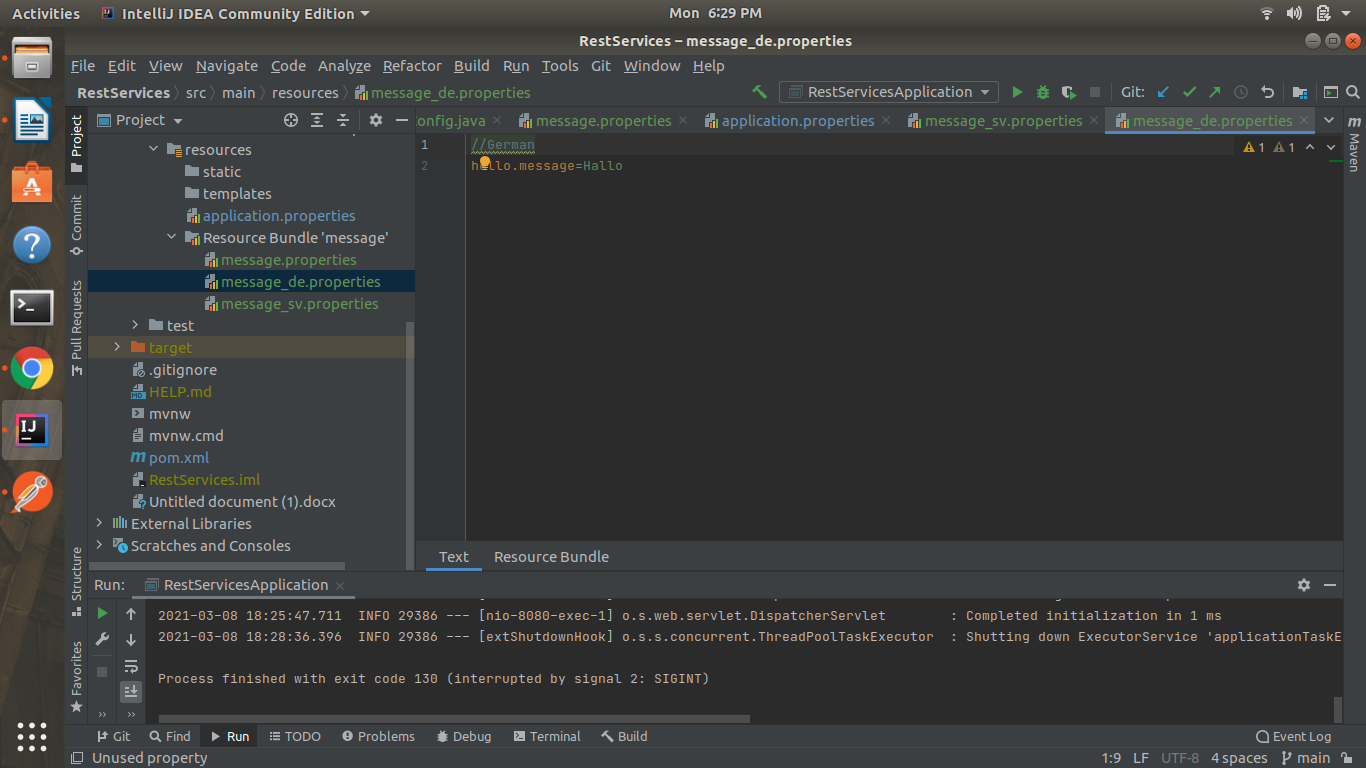
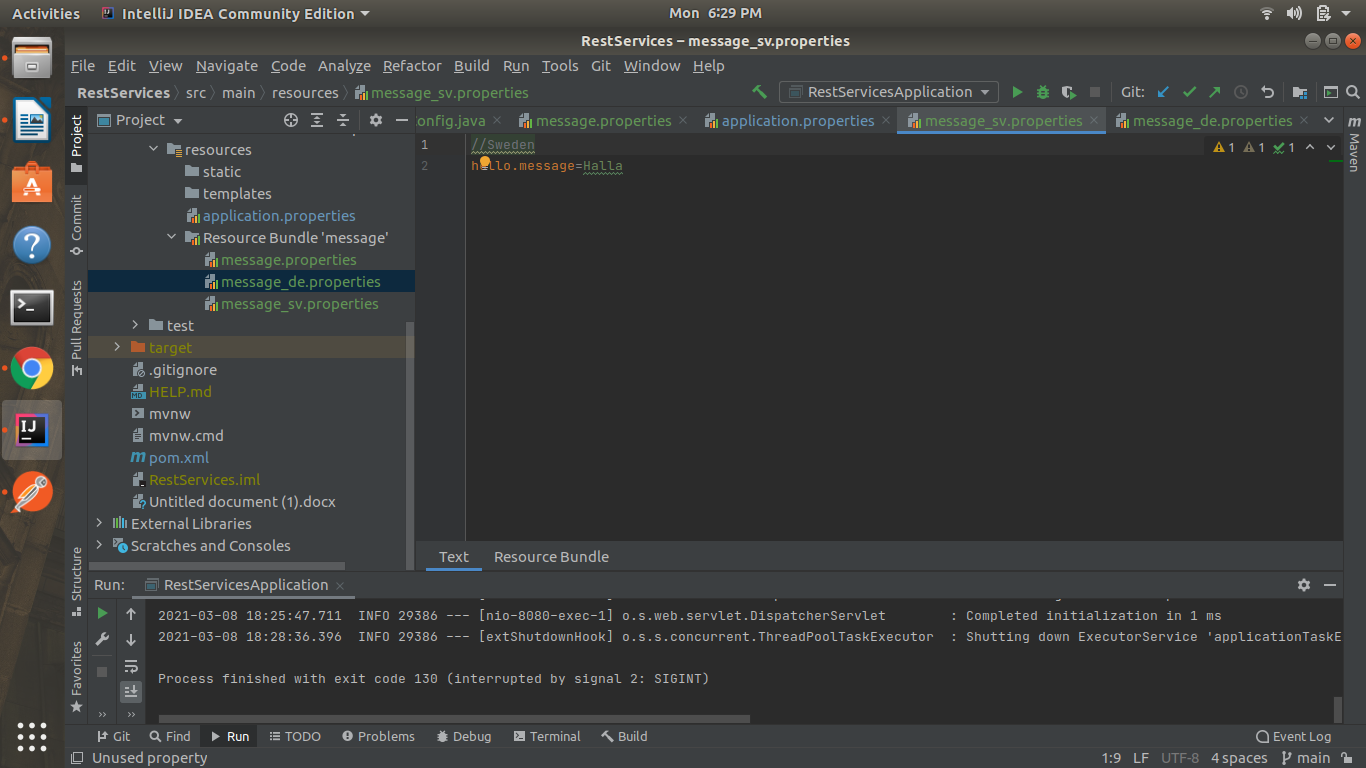
**Assignment**

**RestFul Web Service Part 2**

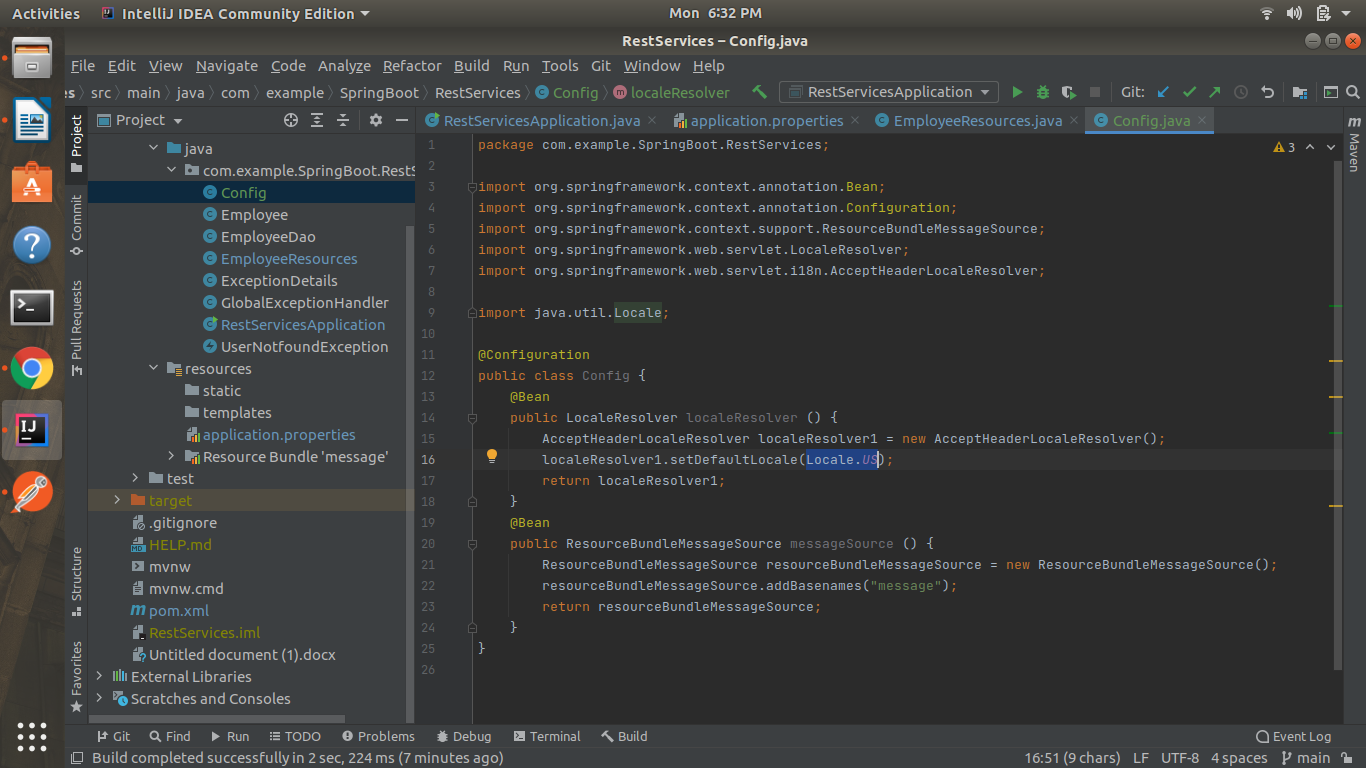
Date-08/03/21

Q1)Add support for Internationalization in your application allowing messages to be shown in English, German and Swedish, keeping English as default.





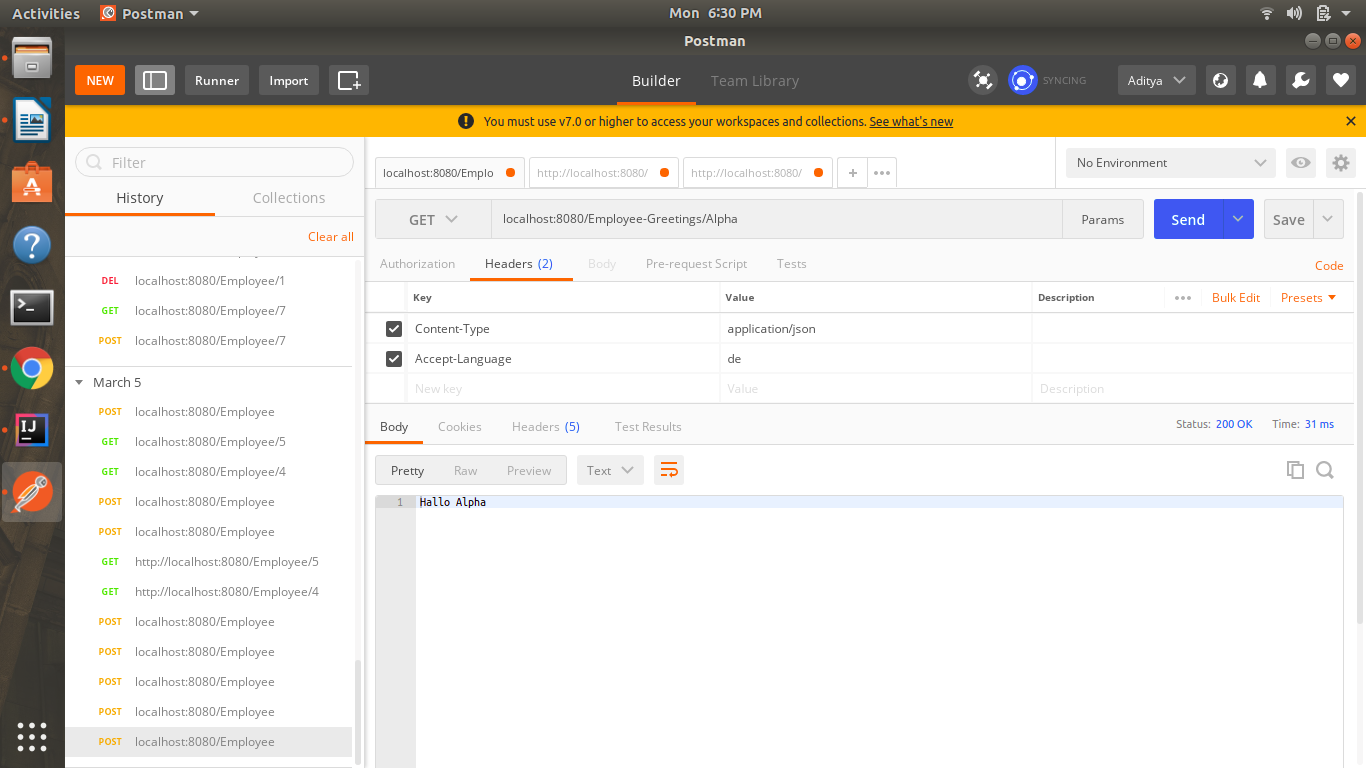
Default-English



Q2)Create a GET request which takes "username" as param and shows a localized message "Hello Username". (Use parameters in message properties)

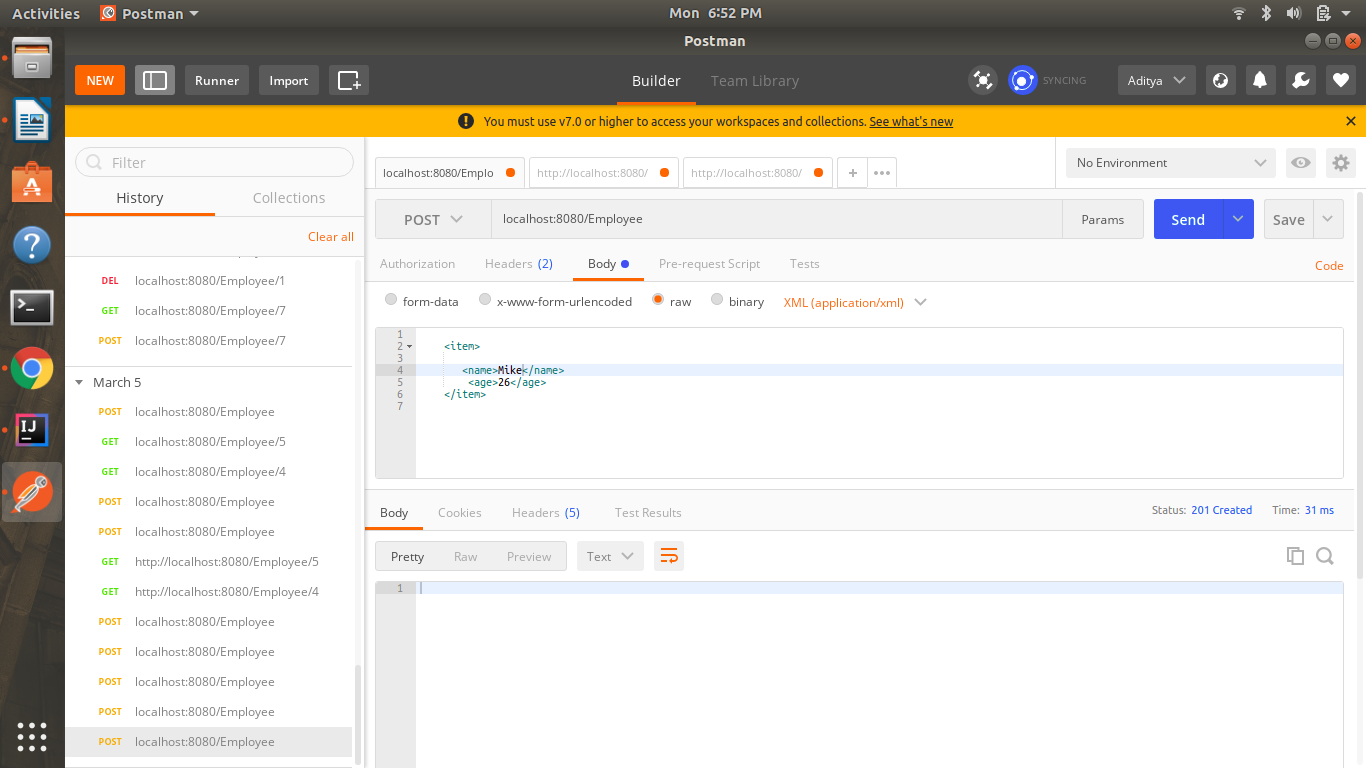
de for German

**GET-localhost:8080/employee/greetings/Alpha**



Q3)Create POST Method to create user details which can accept XML for user creation.

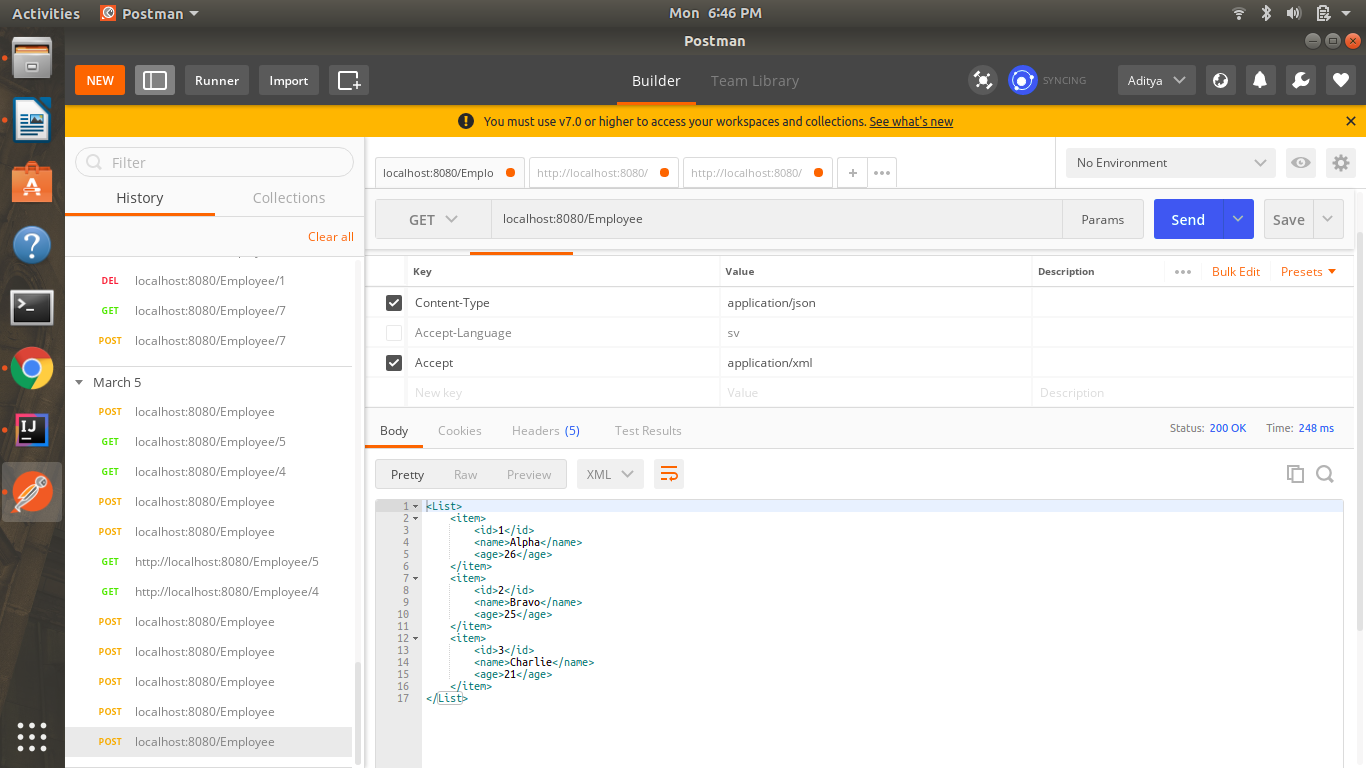
**POST-localhost:8080/employee**



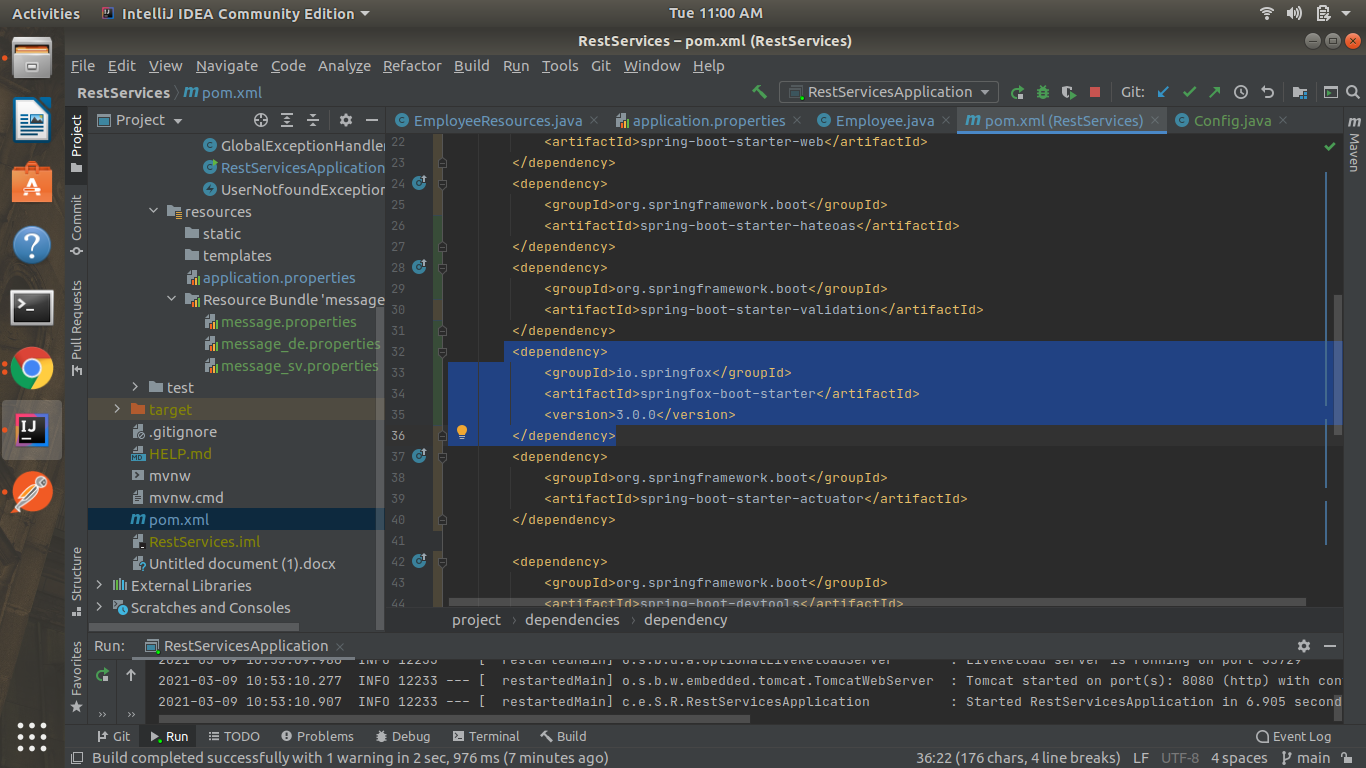
Q4)Create GET Method to fetch the list of users in XML format.

Answer-Here, I have shown the list of Employees in xml.

**GET-localhost:8080/employee**

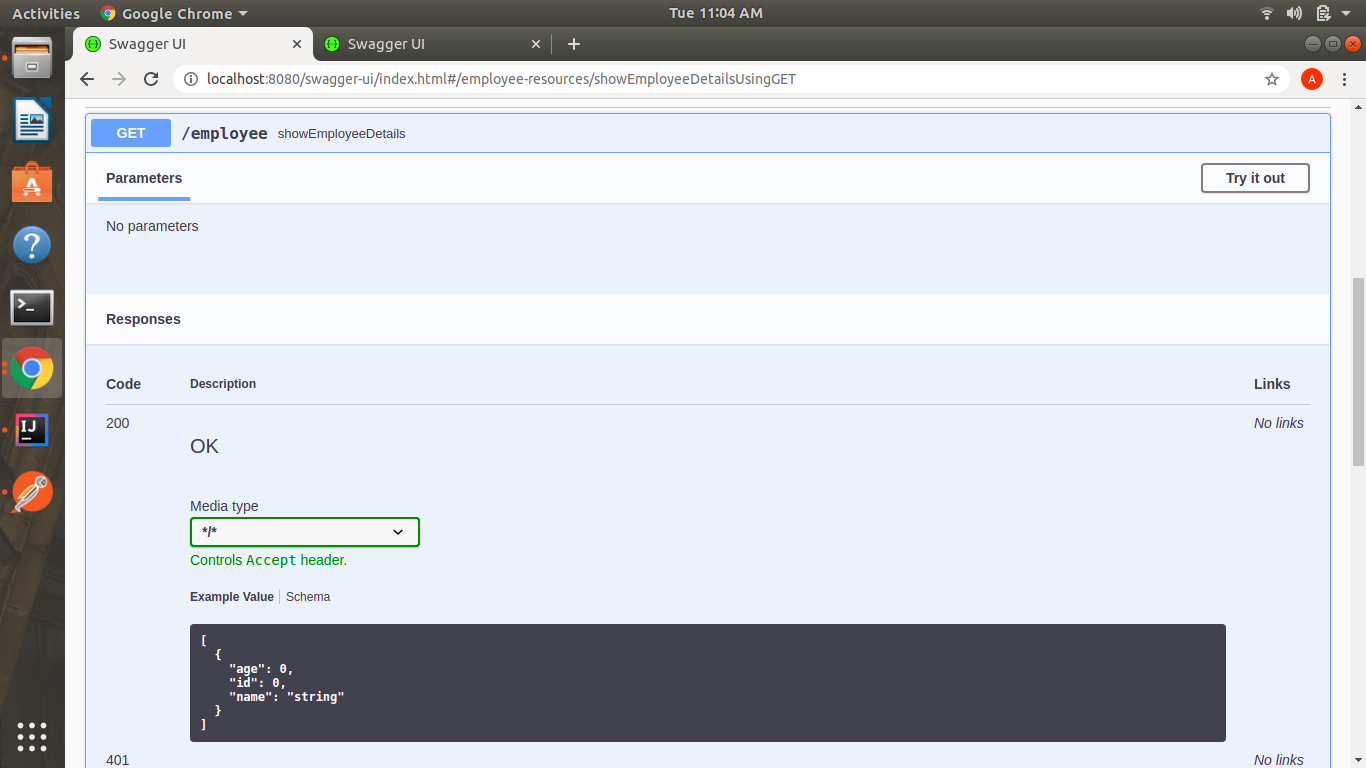


Q5) Configure swagger plugin and create document of following methods:



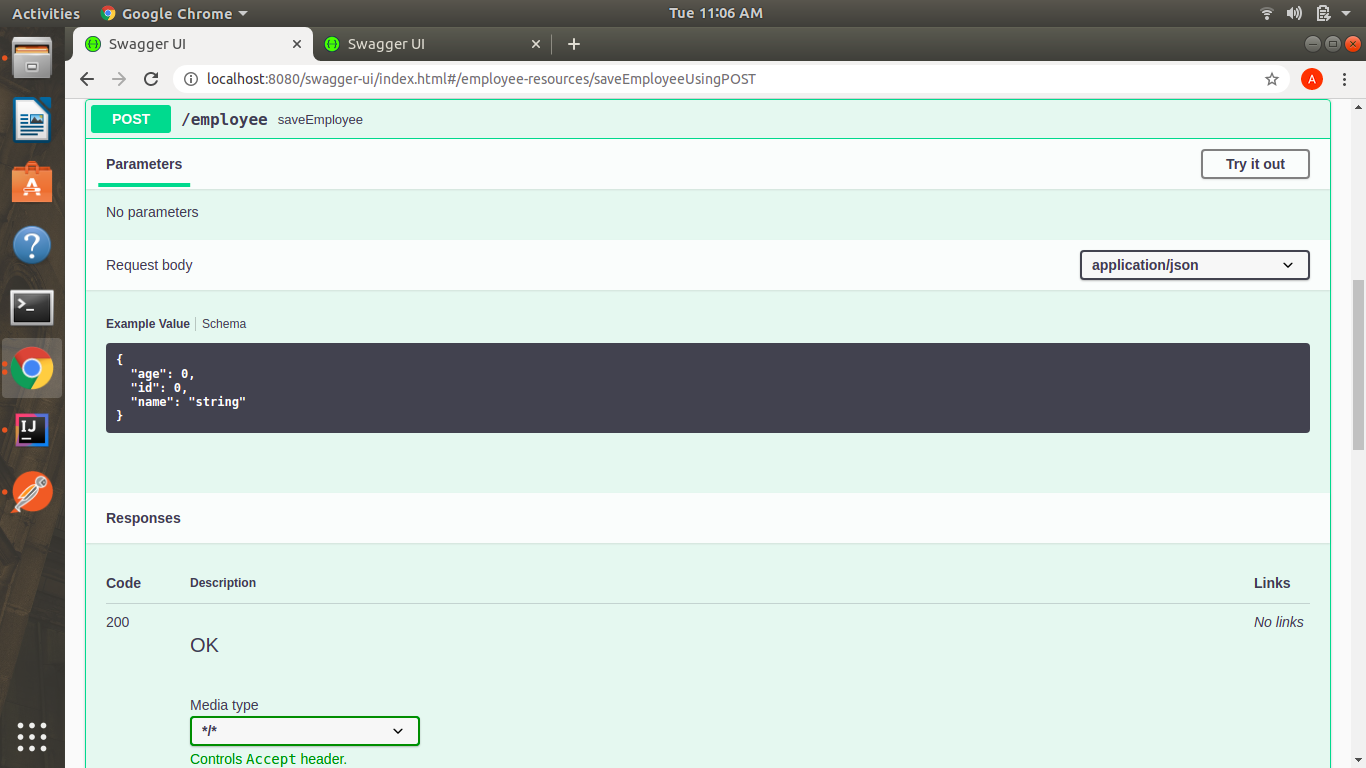
a)Get details of User using GET request.

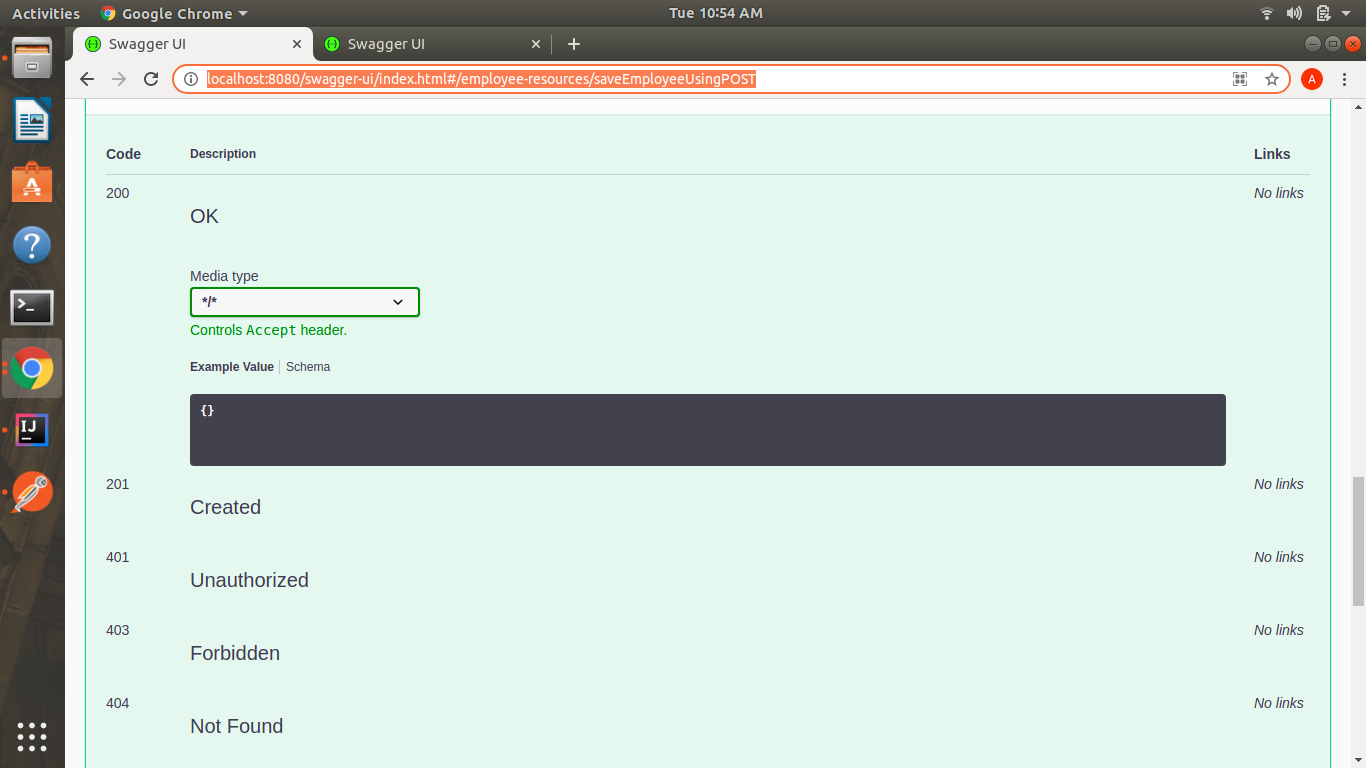
<http://localhost:8080/swagger-ui/index.html#/employee-resources/showEmployeeDetailsUsingGET>



b) Save details of the user using POST request.

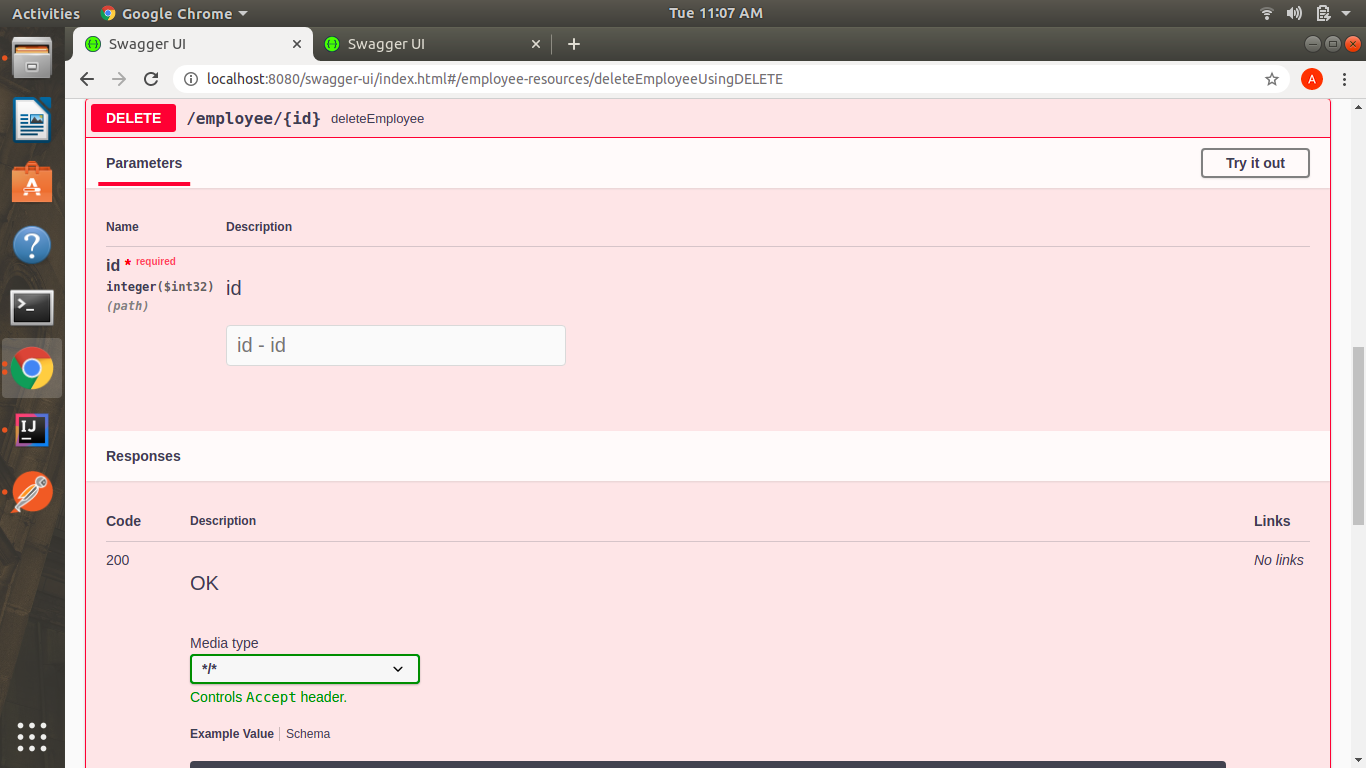
<http://localhost:8080/swagger-ui/index.html#/employee-resources/saveEmployeeUsingPOST>

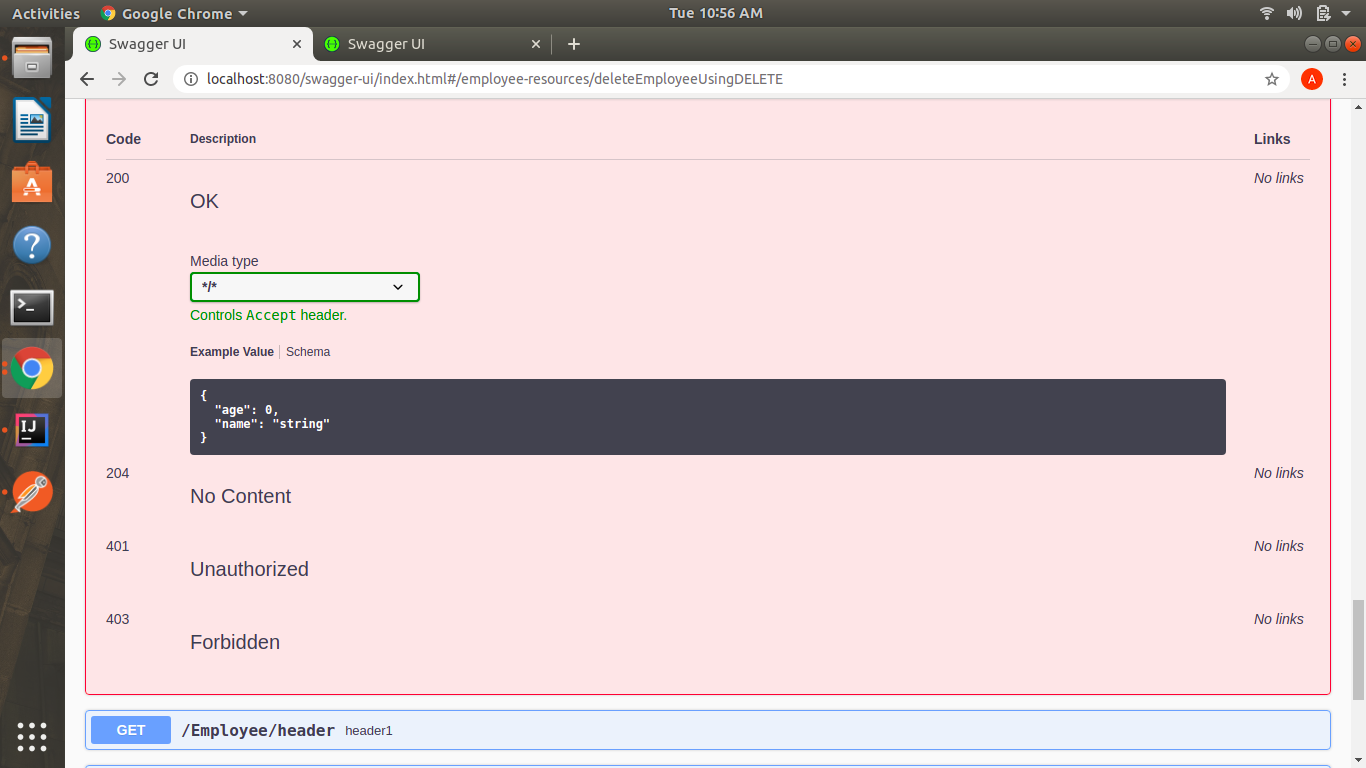




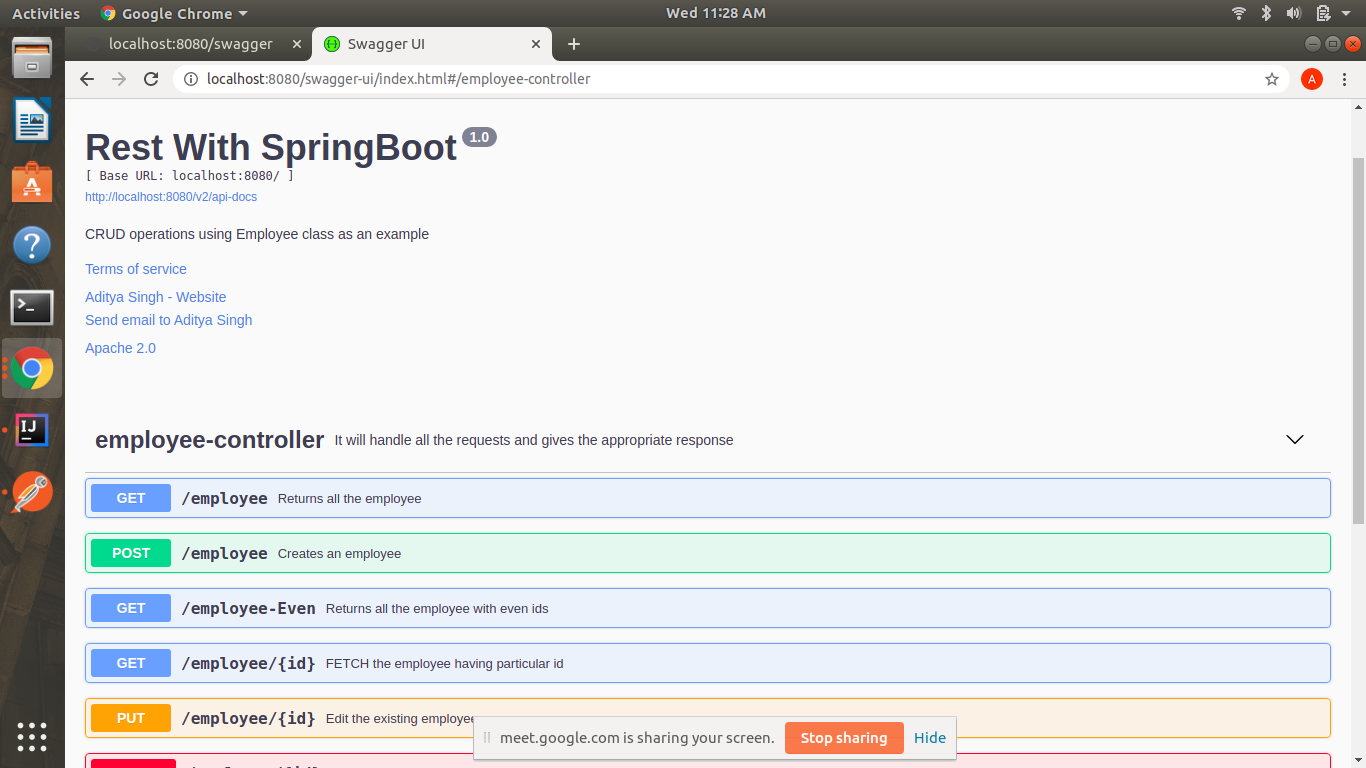
c) Delete a user using DELETE request.

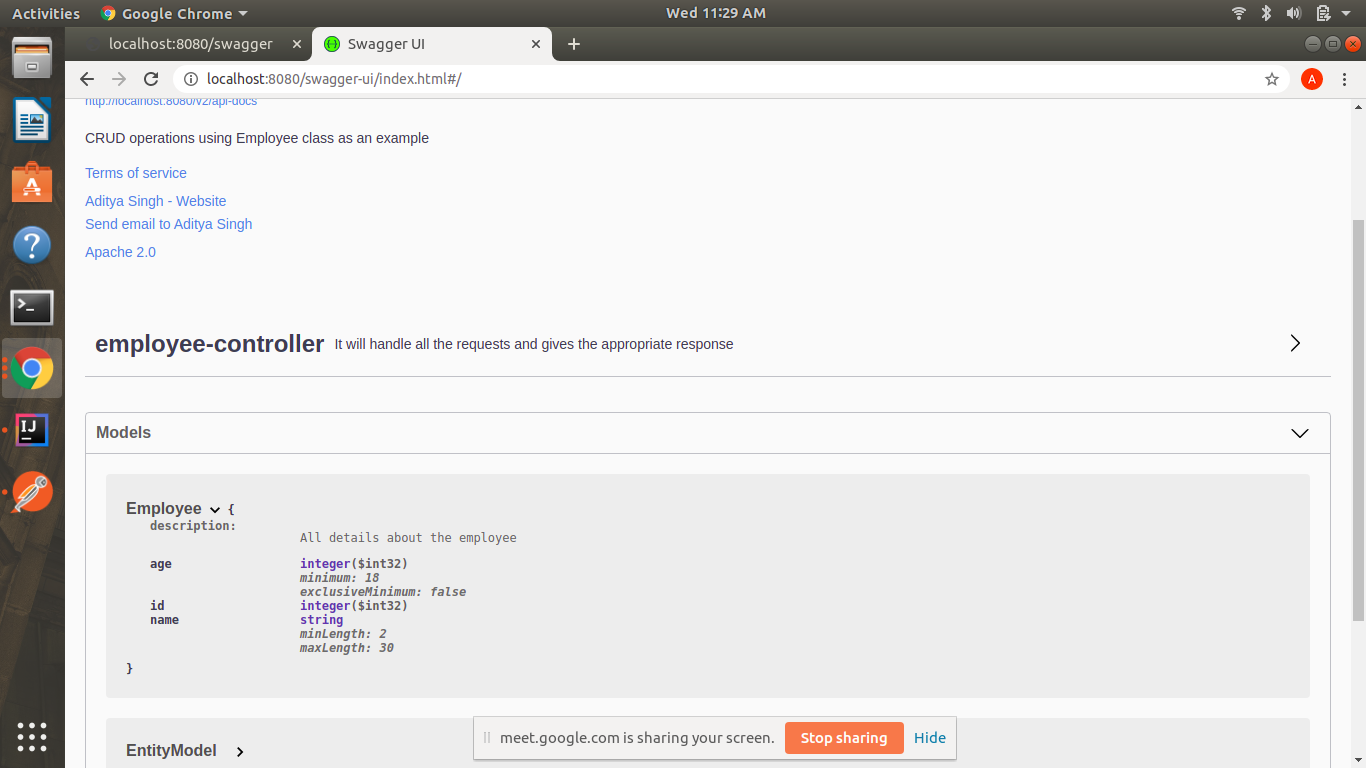
<http://localhost:8080/swagger-ui/index.html#/employee-resources/deleteEmployeeUsingDELETE>

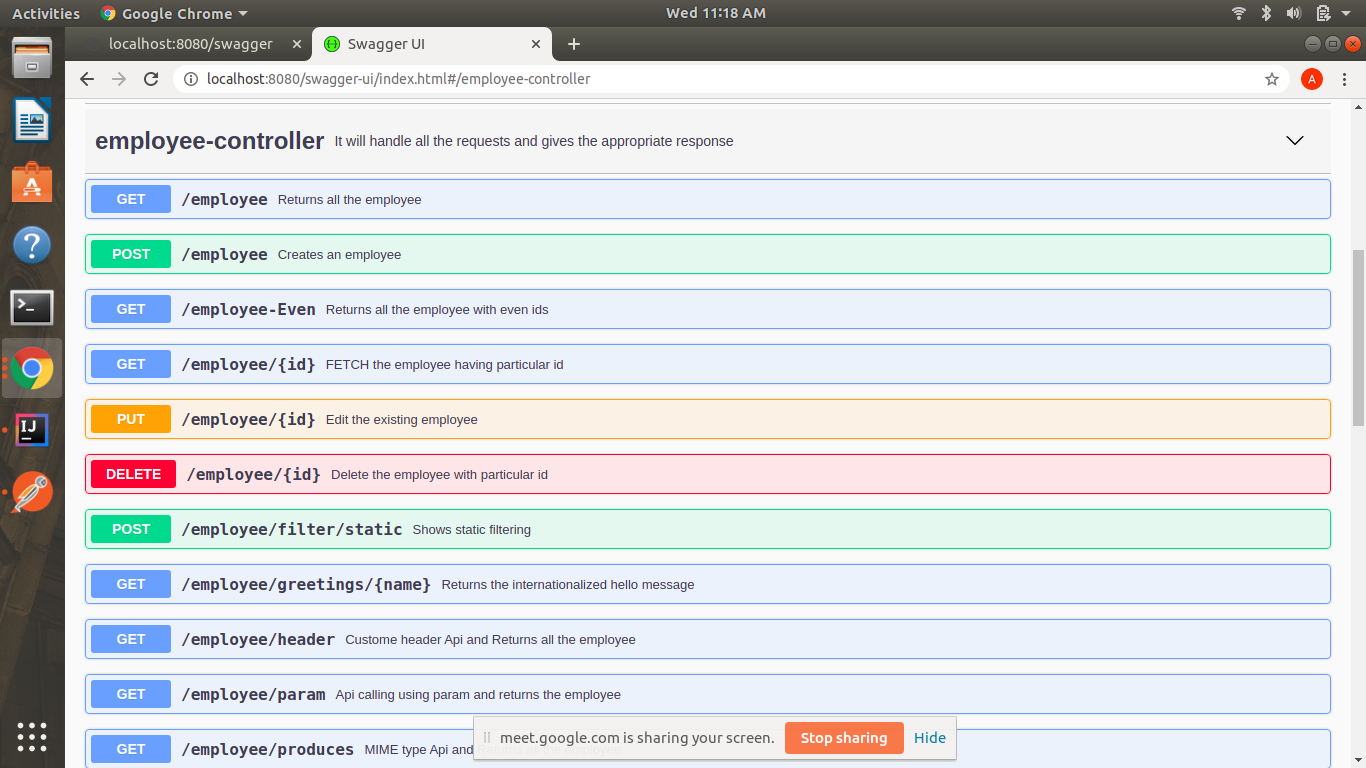




Q7)In swagger documentation, add the description of each class and URI so that in swagger UI the purpose of class and URI is clear.

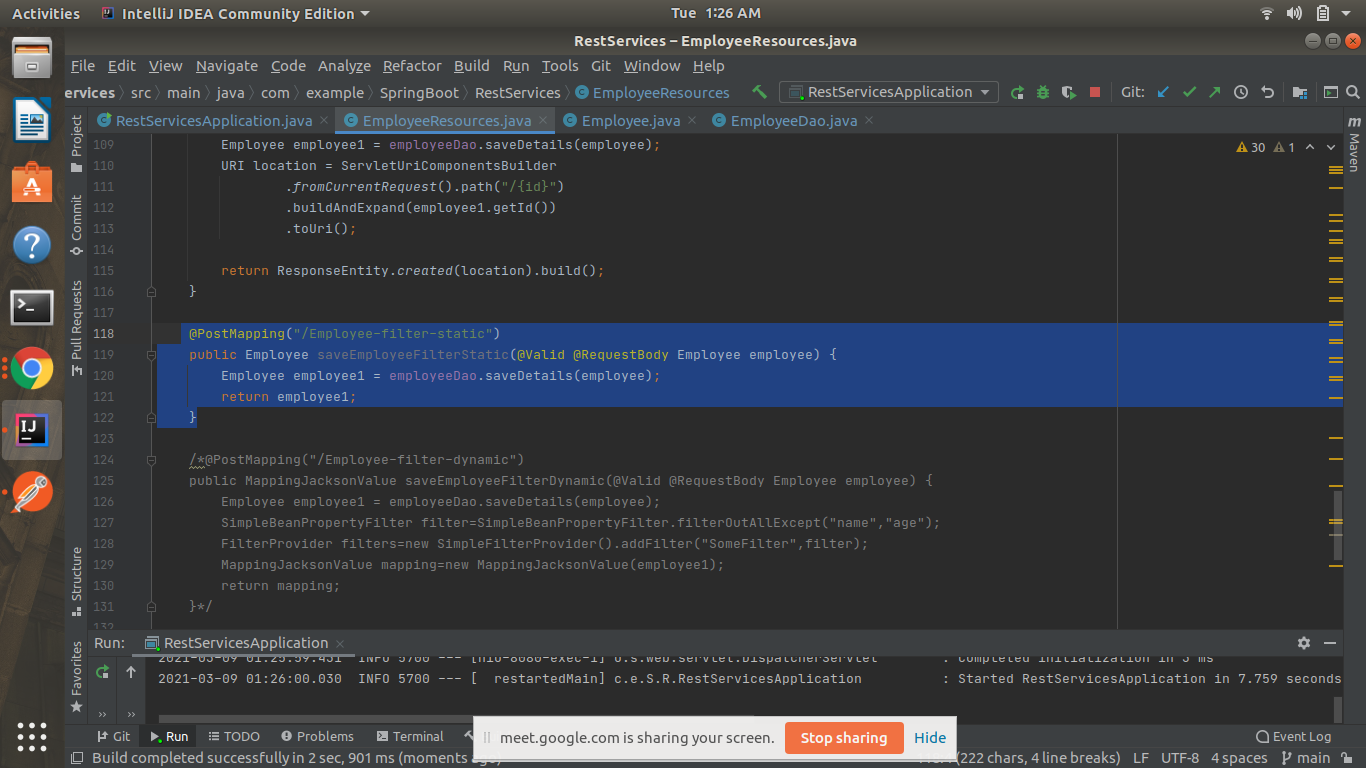






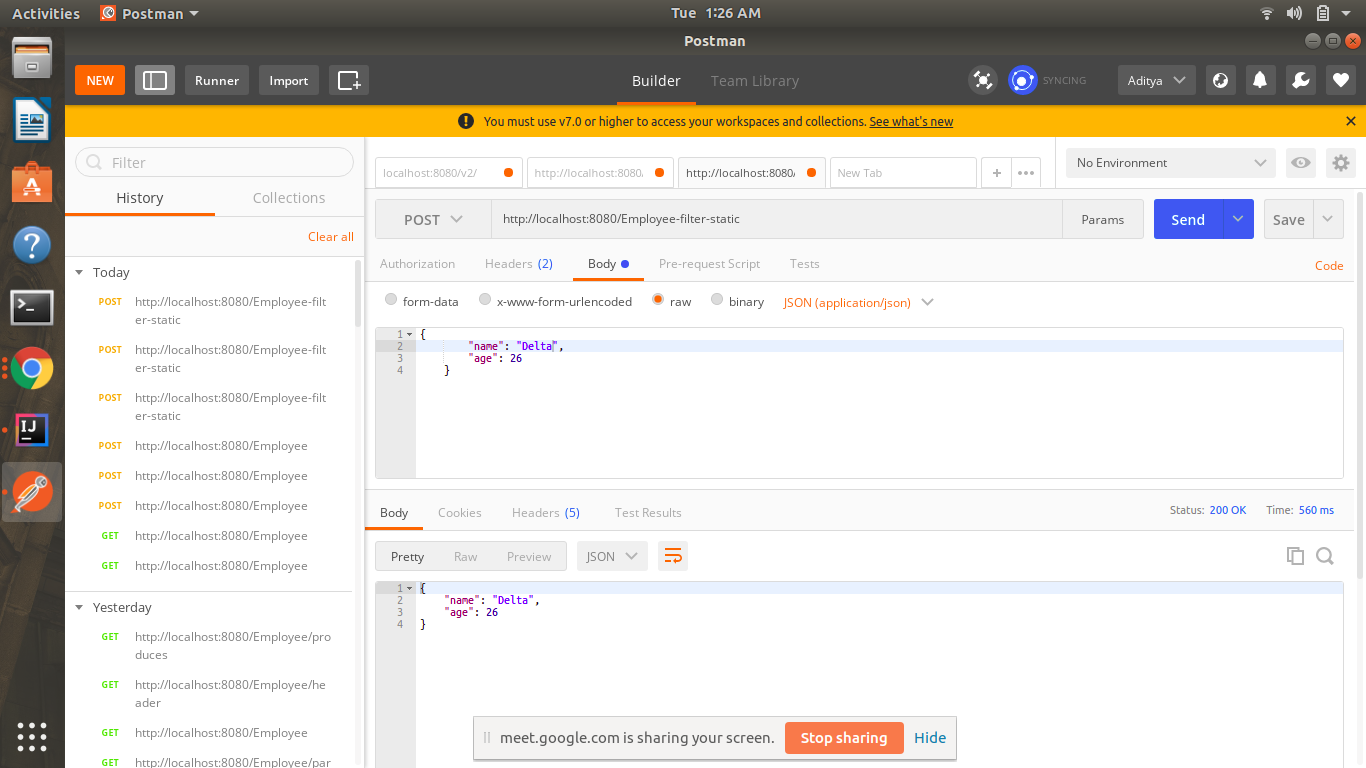
Q8) . Create API which saves details of User (along with the password) but on successfully saving returns only non-critical data. (Use static filtering)

@PostMapping(“/employee/filter/static”)

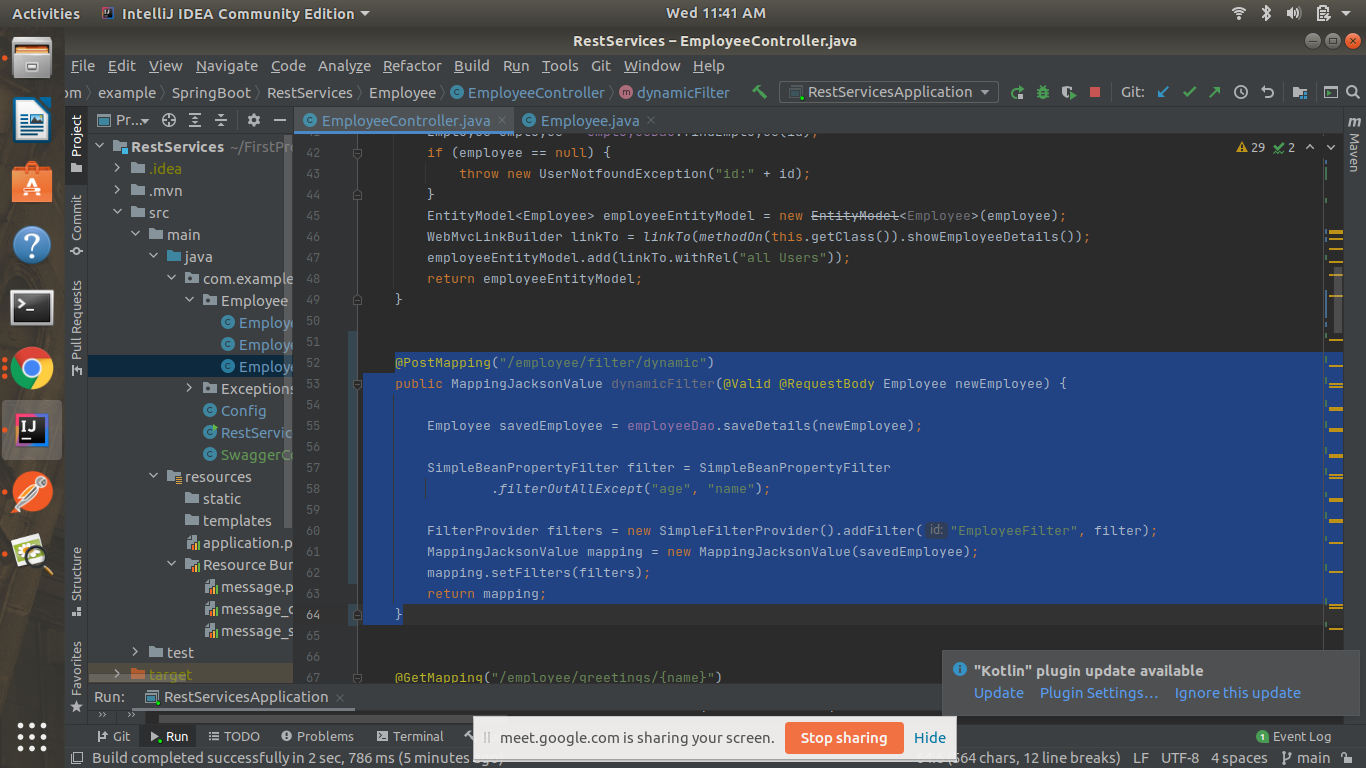


In the code, I am returning a complete employee but because of@JsonIgnore over Integer id, it’s not been shown in the response body.

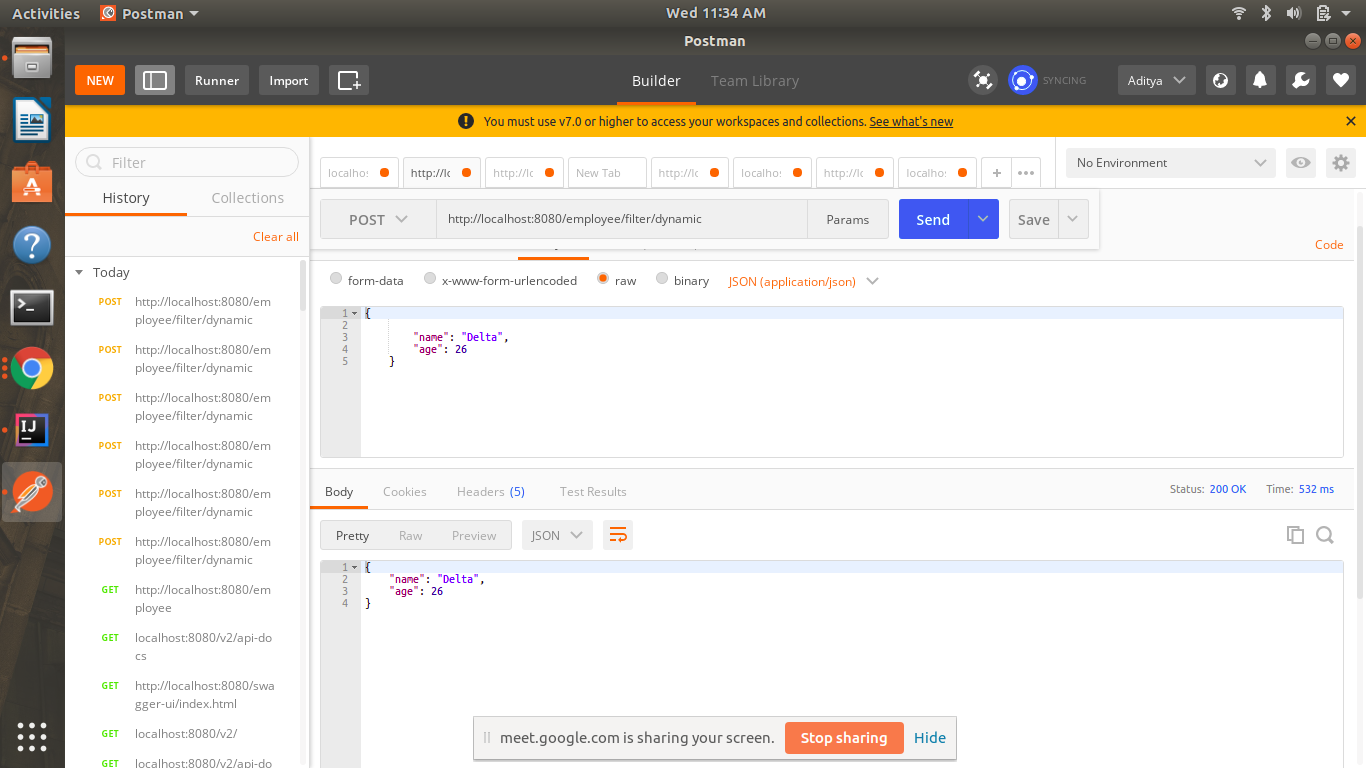
POST-localhost:8080/employee/filter/static



Q9)Create another API that does the same by using Dynamic Filtering.



Putting @JsonFilter("EmployeeFilter") over Employee class.



Q10)Create 2 API for showing user details. The first api should return only basic details of the user and the other API should return more/enhanced details of the user,

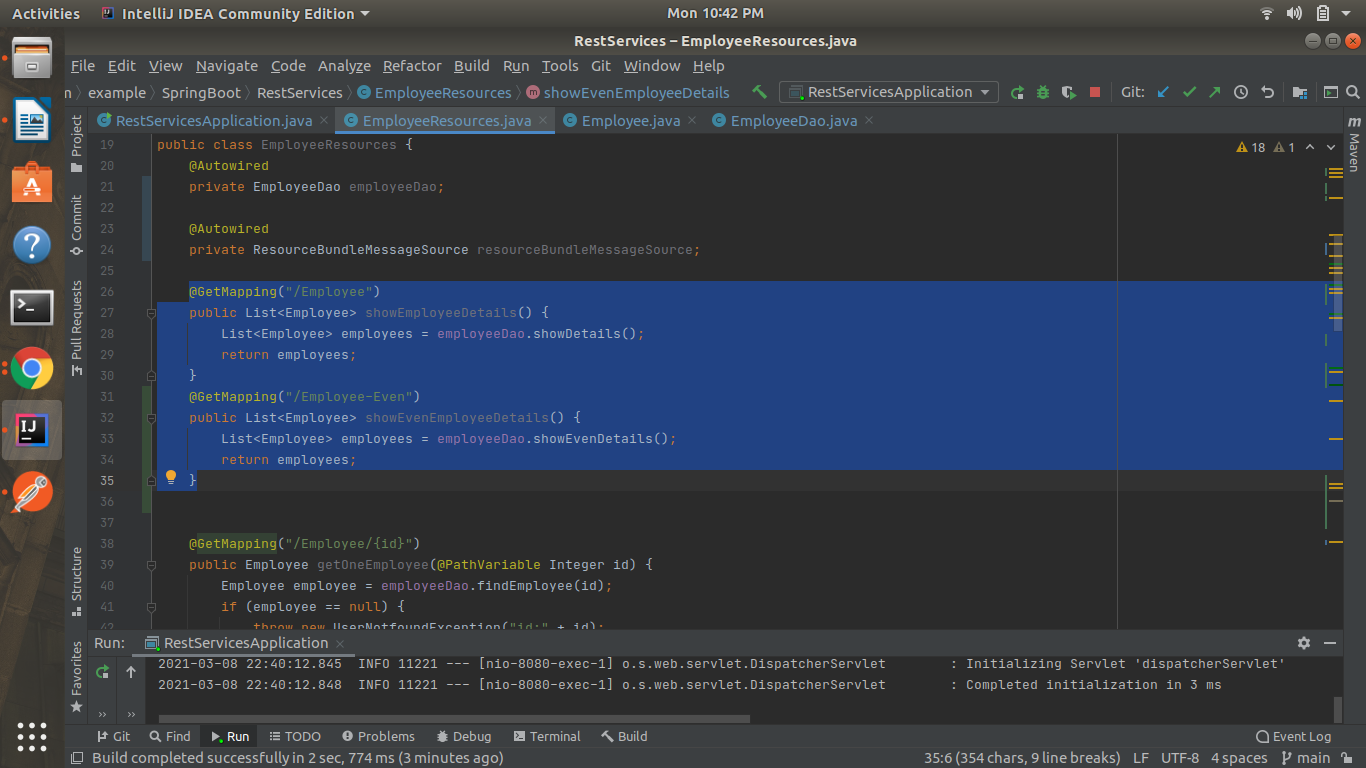
Answer-here, I have created two API, one will show the details of all the employees and another will show only the even employee’s details.

A->URI Versioning

<http://localhost:8080/Employee>

<http://localhost:8080/Employee-Even>

**@GetMapping(“/employee/even”)**

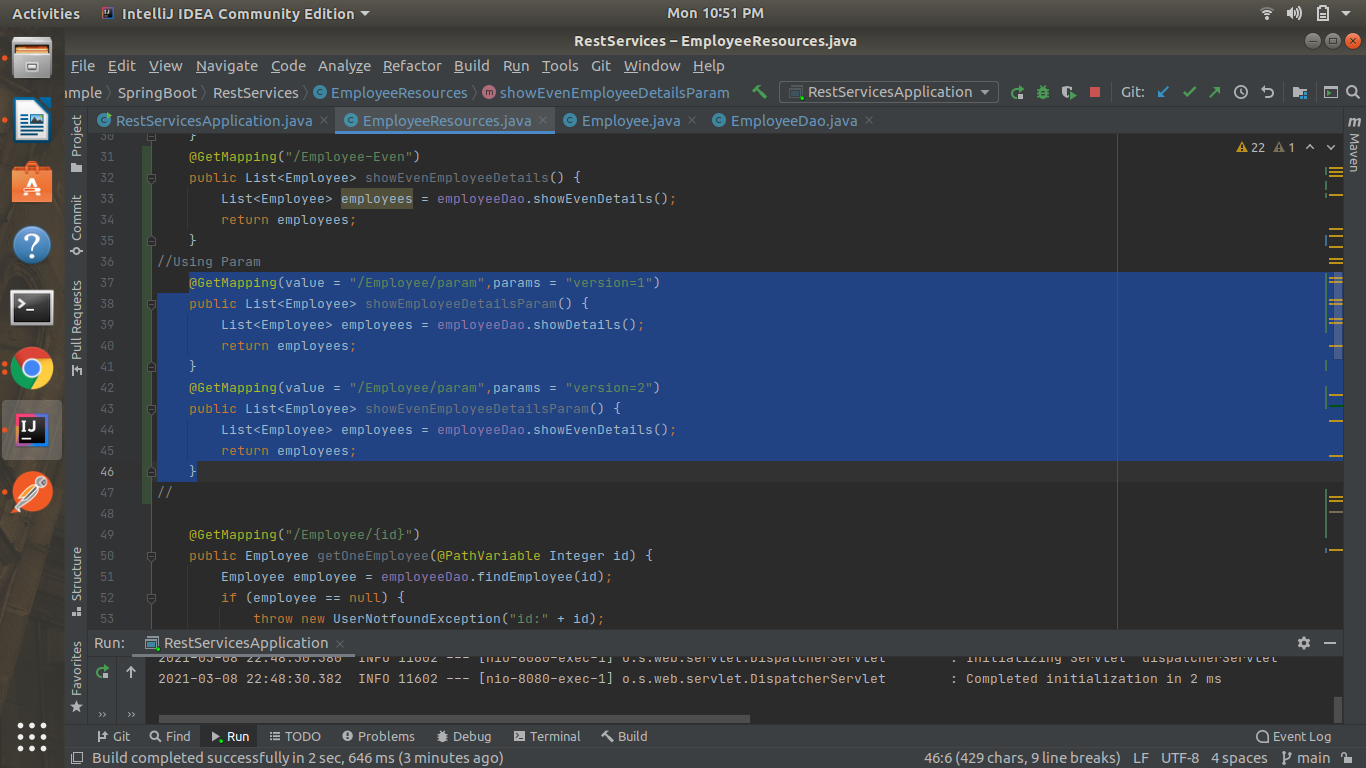


B->Request Parameter versioning

<http://localhost:8080/Employee/param?version=1>

<http://localhost:8080/Employee/param?version=2>

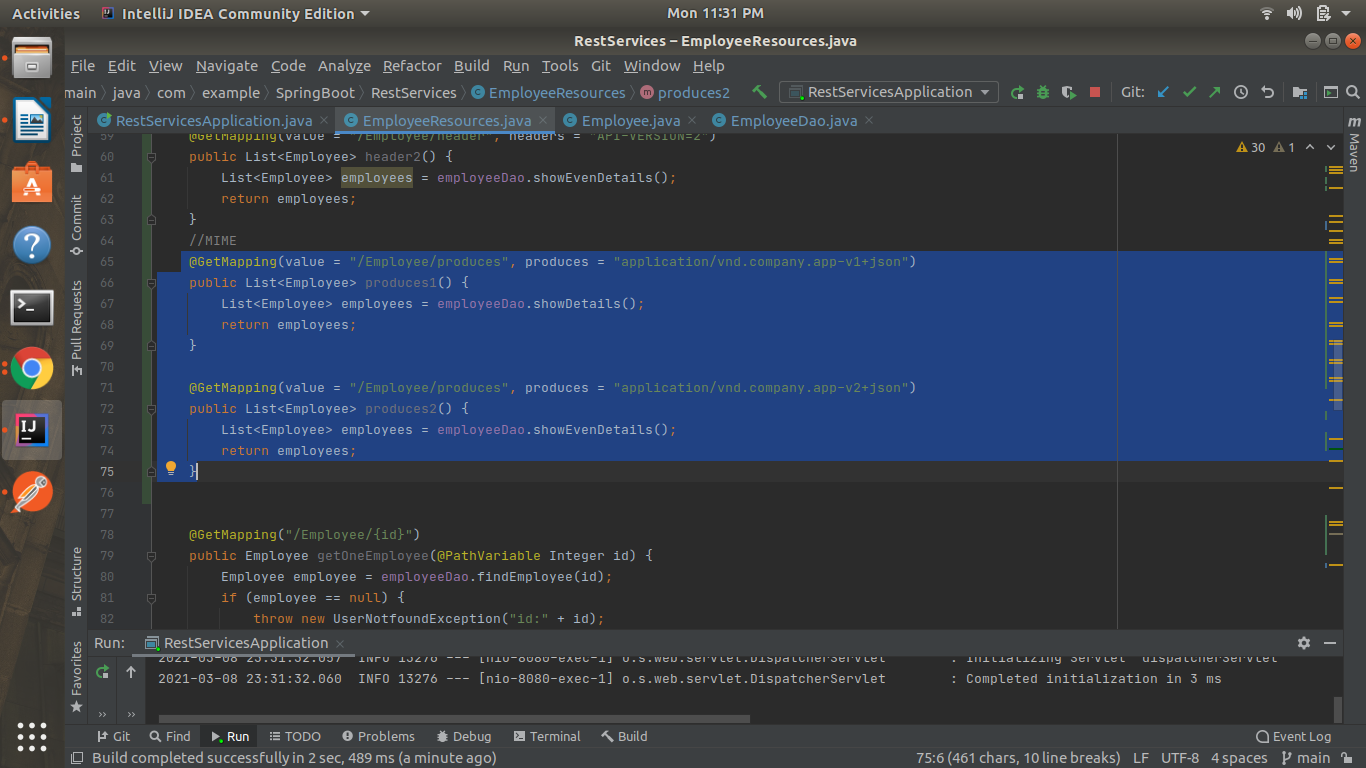
@GetMapping(“value=/employee/param”)



C->MimeType Versioning

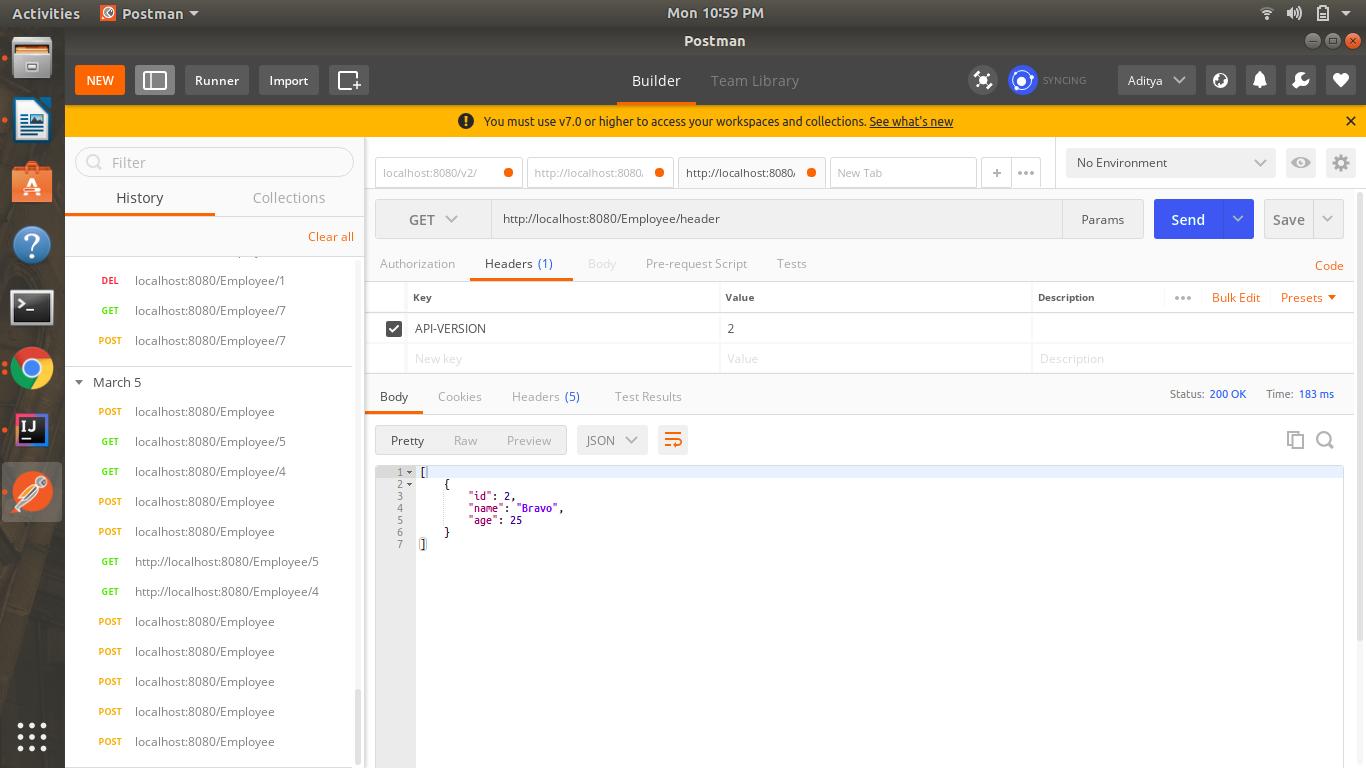
<http://localhost:8080/Employee/produces>

**/Employee->/employee**

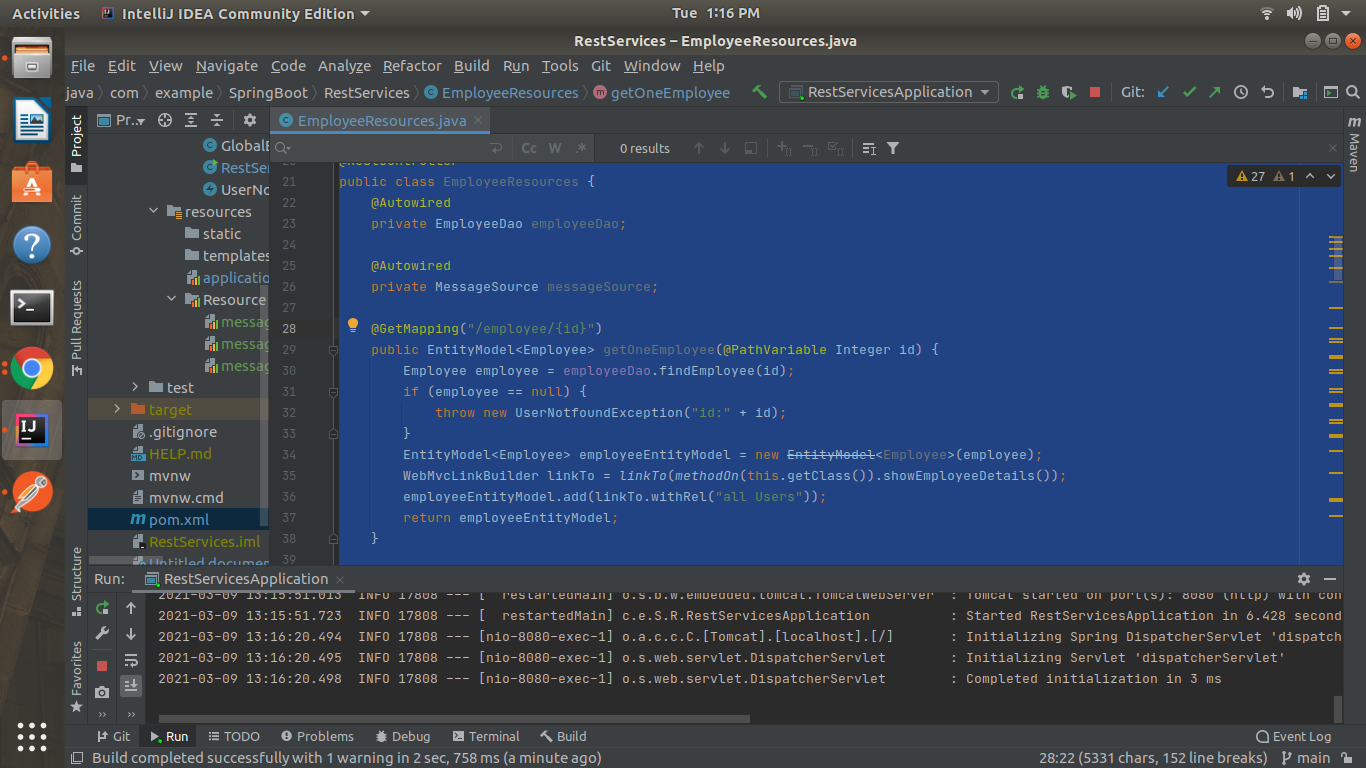


D->Custom Header Versioning

GET-[http://localhost:8080/employee/header](http://localhost:8080/Employee/header)



Q11)Configure hateoas with your spring boot application. Create an api which returns User Details along with url to show all topics.



<http://localhost:8080/employee/1>

