WEEK 4 MANDATORY HANDS-ON QUESTIONS

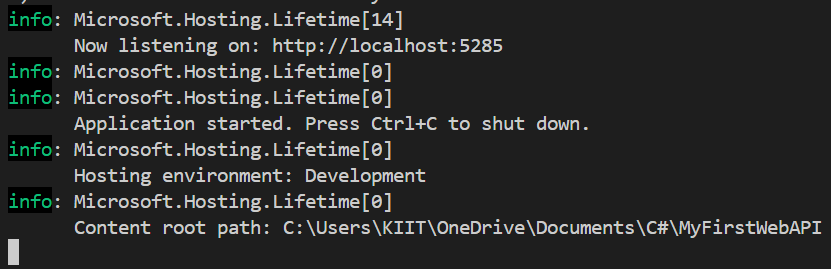
1. **RESTful Web Service**: An API that follows REST principles to allow communication between systems using HTTP.

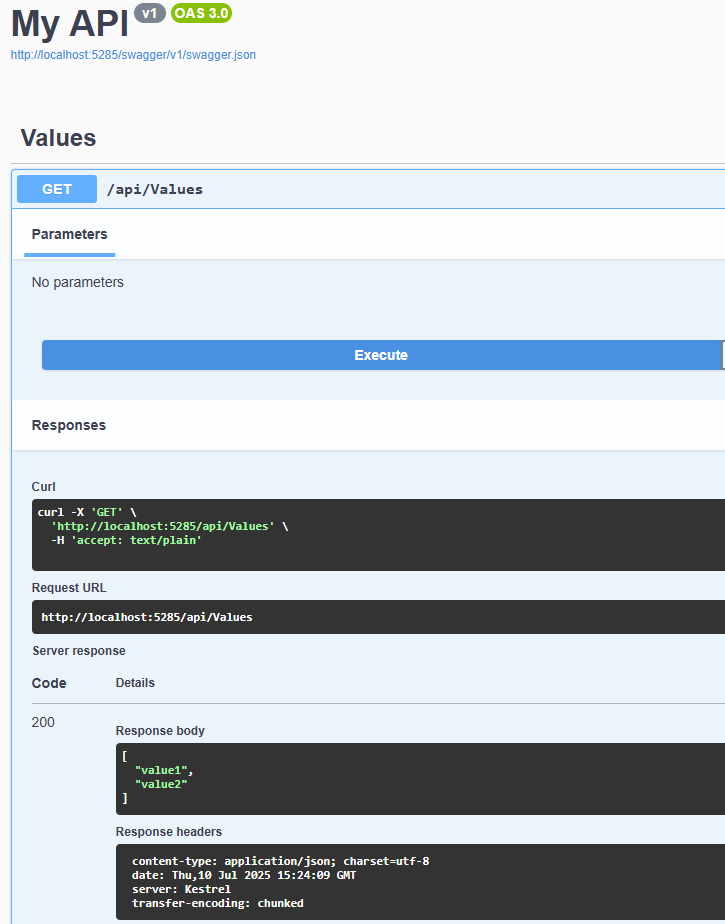
**Web API**: A framework for building HTTP services, especially RESTful services in .NET.

**Microservice**: A small, independently deployable service that does one thing well, part of a larger distributed system.

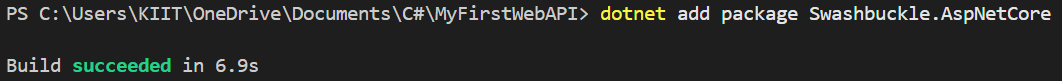
* **Representational State Transfer**: Uses standard HTTP methods to access and manipulate resources.
* **Stateless**: Each request is independent; the server does not store client state.
* **Messages**: Data is exchanged using formats like JSON, XML over HTTP.
* **Not limited to XML**: Can return JSON, plain text, HTML, etc.
* **Microservices Compatible**: REST APIs are ideal for microservice communication due to simplicity and scalability.
* **HttpRequest**: Represents the incoming HTTP request from a client (contains URL, headers, method, body).
* **HttpResponse**: Represents the outgoing response sent back to the client (status code, headers, body).
* **HttpGet**: Used to retrieve data.
* **HttpPost**: Used to create a new resource.
* **HttpPut**: Used to update an existing resource.
* **HttpDelete**: Used to delete a resource.

**First Web Api using .Net core**





1. **NuGet Package**



**Structure in Postman**

* Organized into Workspaces > Collections > Requests
* **Request Type:**

Choose GET, POST, PUT, DELETE from dropdown near URL

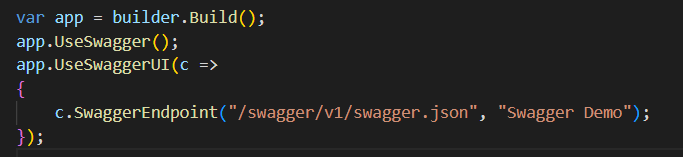
* **Tabs in Center Pane:**

Tabs like Params, Auth, Headers, Body, Test appear after selecting a request

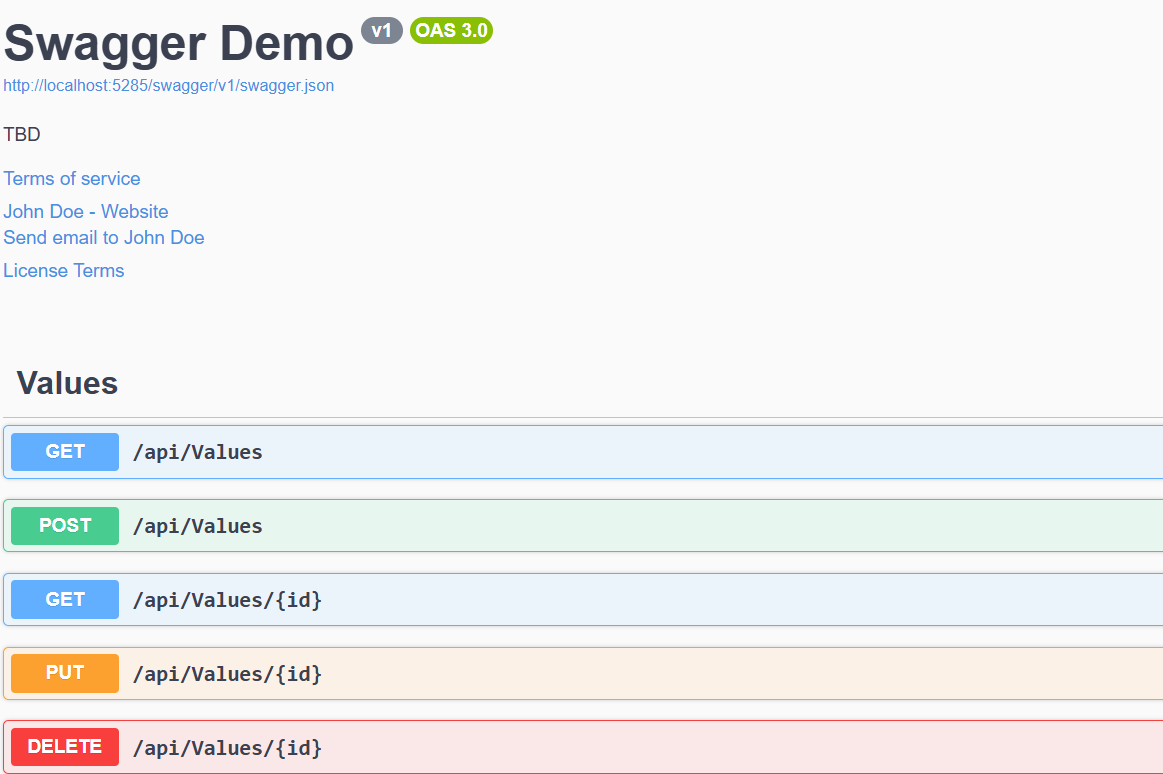
**Web Api using .Net core with Swagger**

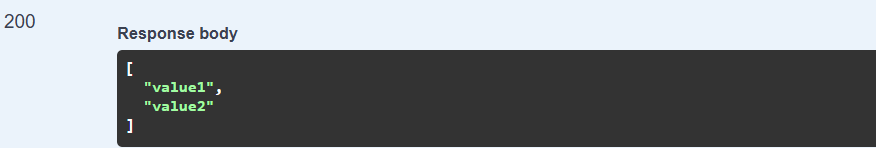


**Enable Swagger in the App Pipeline**

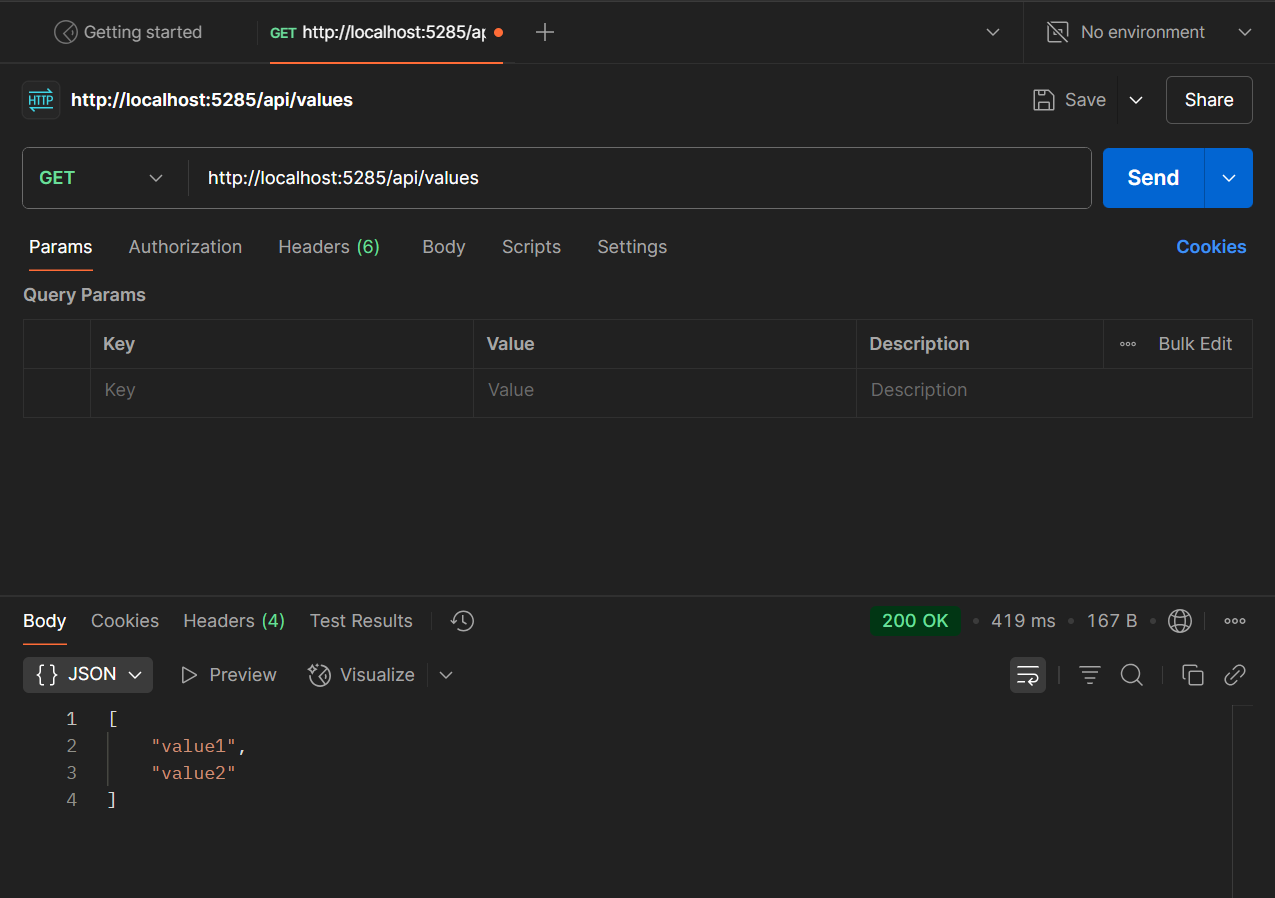


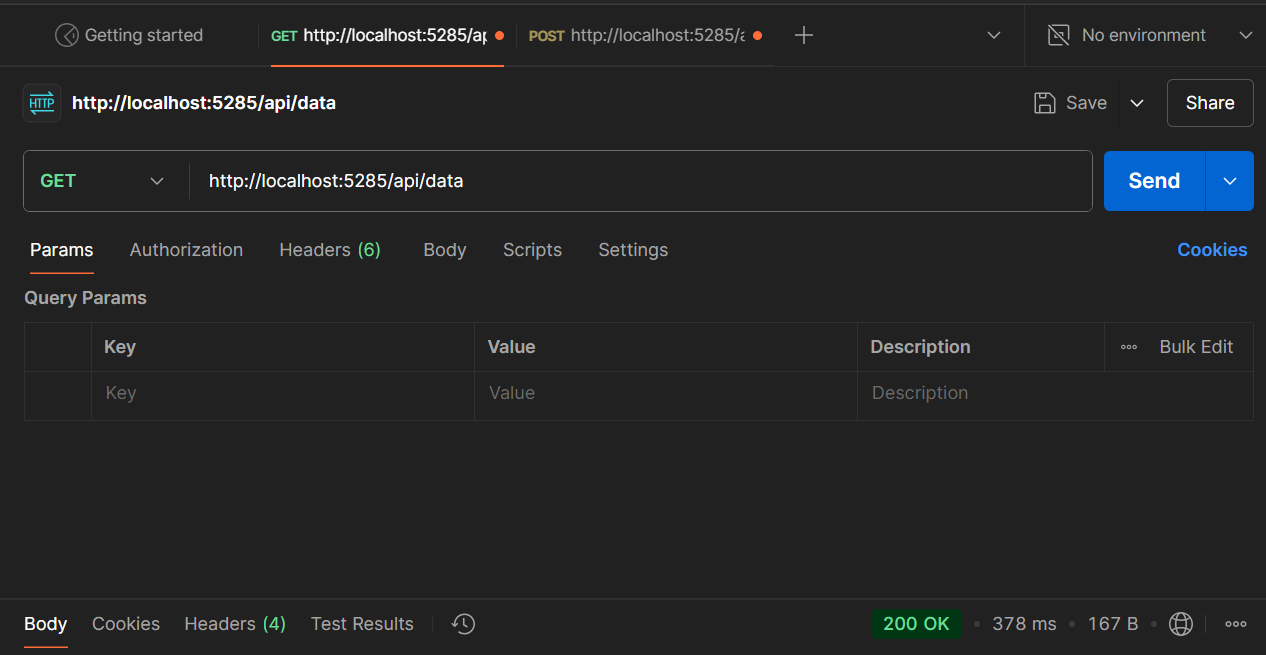
**SWAGGER UI**



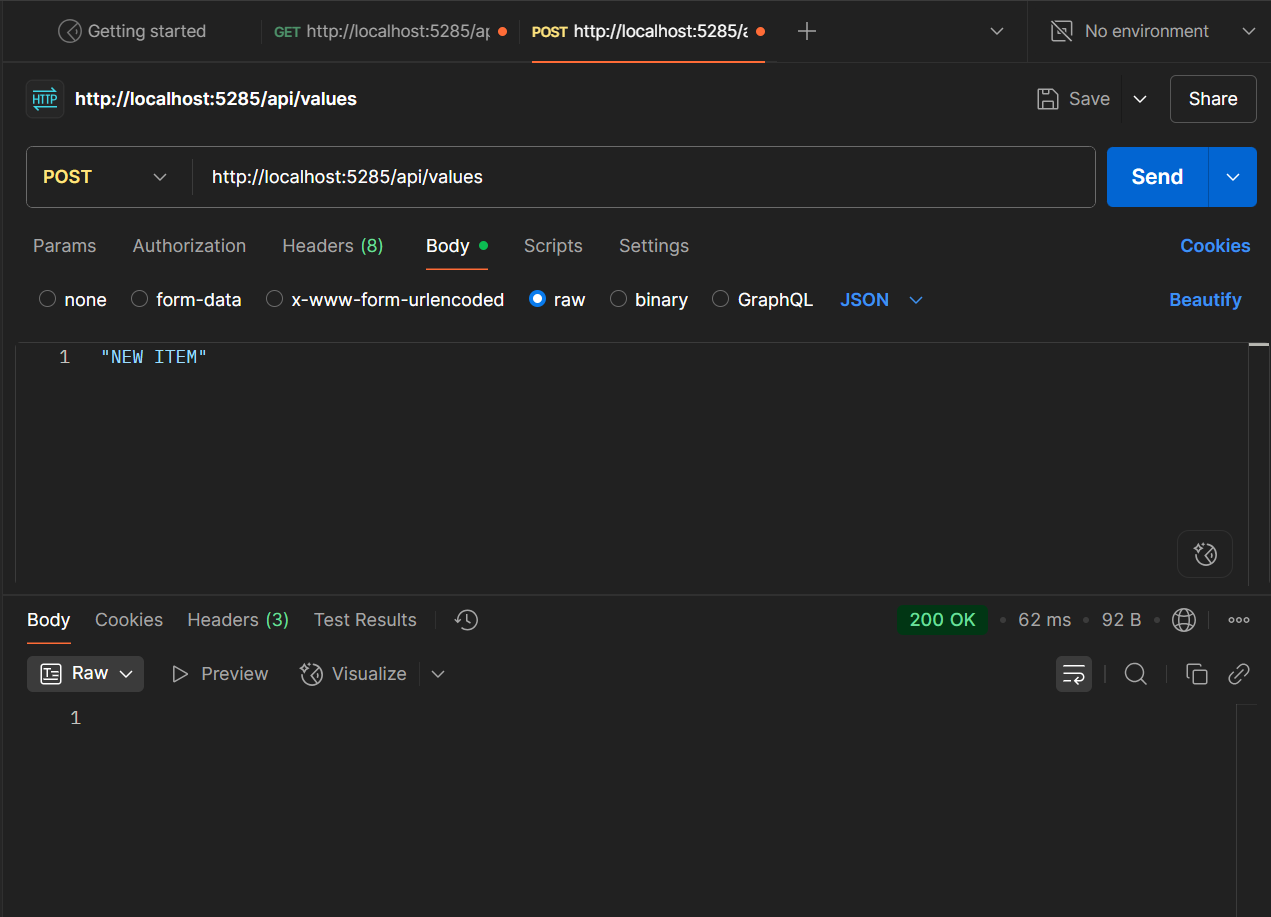


**GET METHOD**



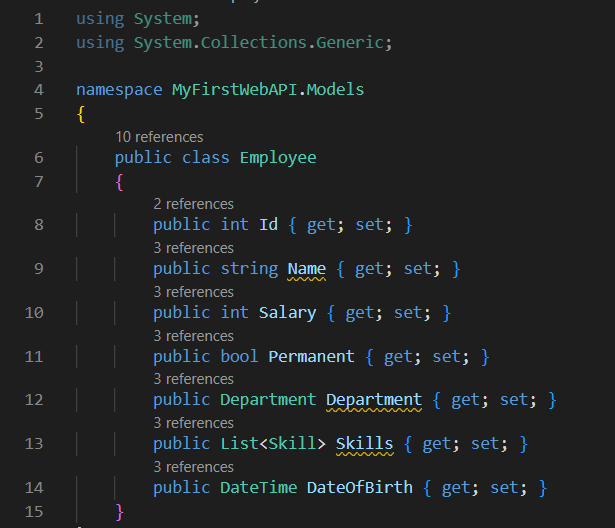


**POST METHOD**

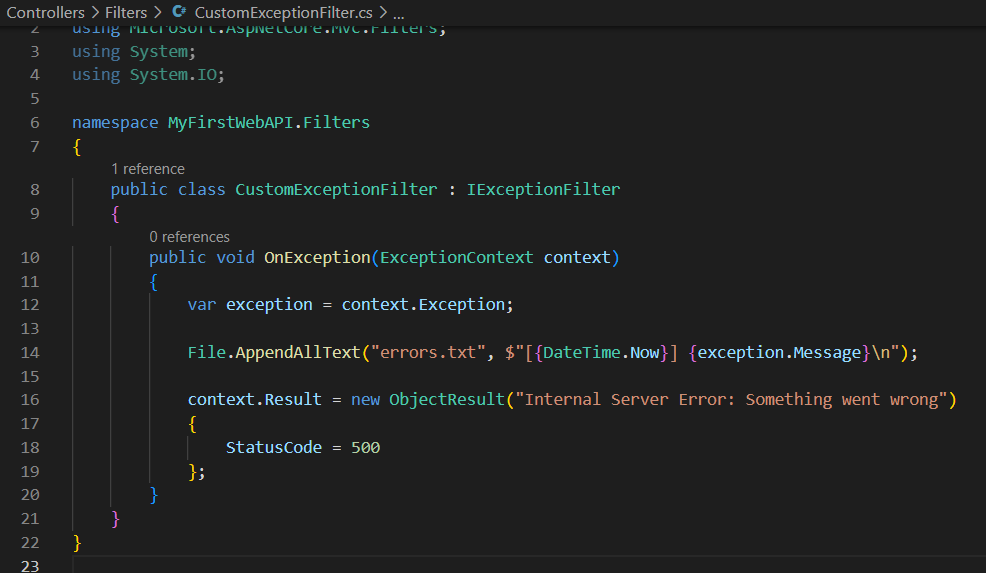


1. **Web Api using custom model class**

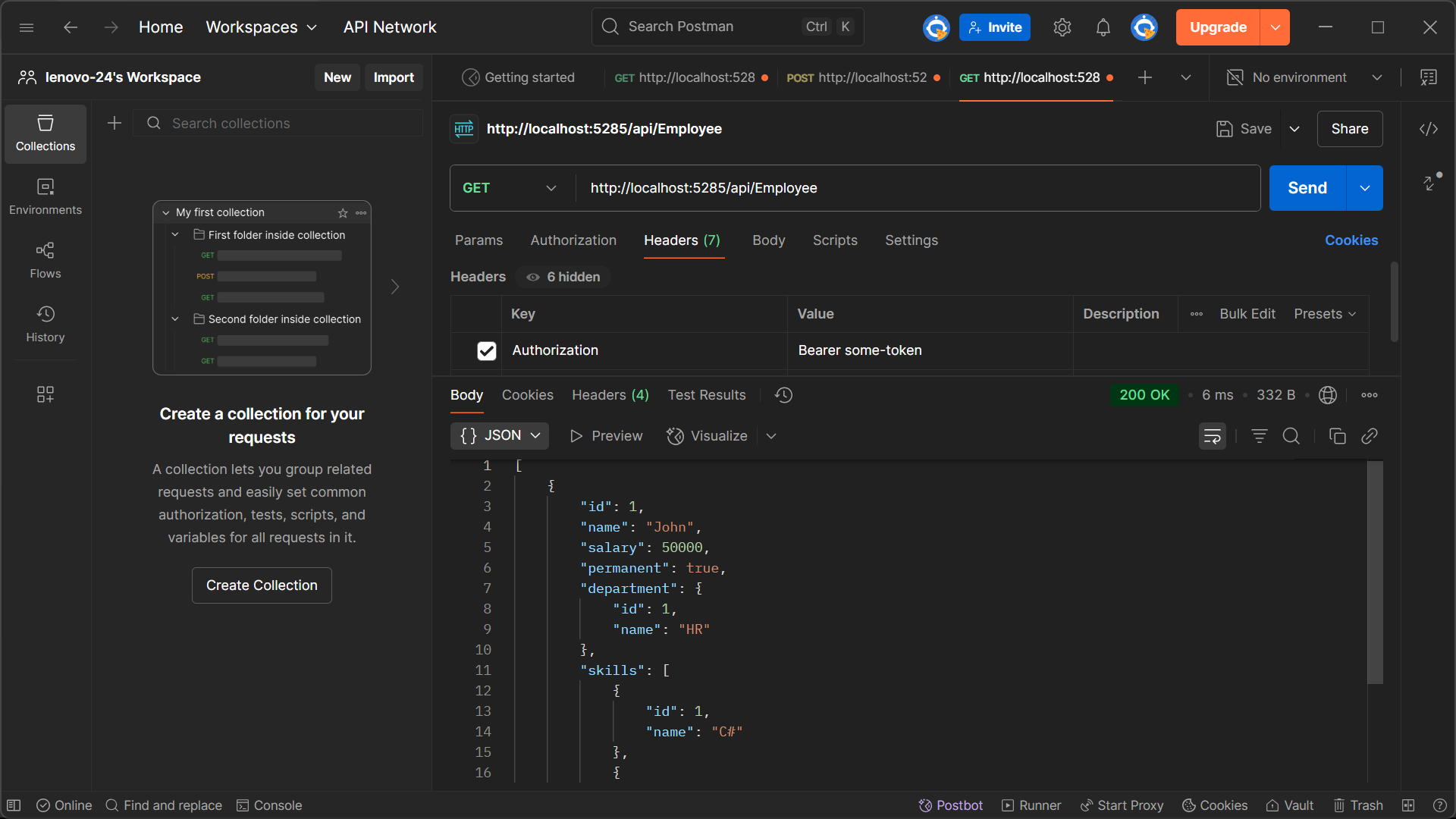
**Creating a Custom class ‘Employee’ of the below defined structure.**



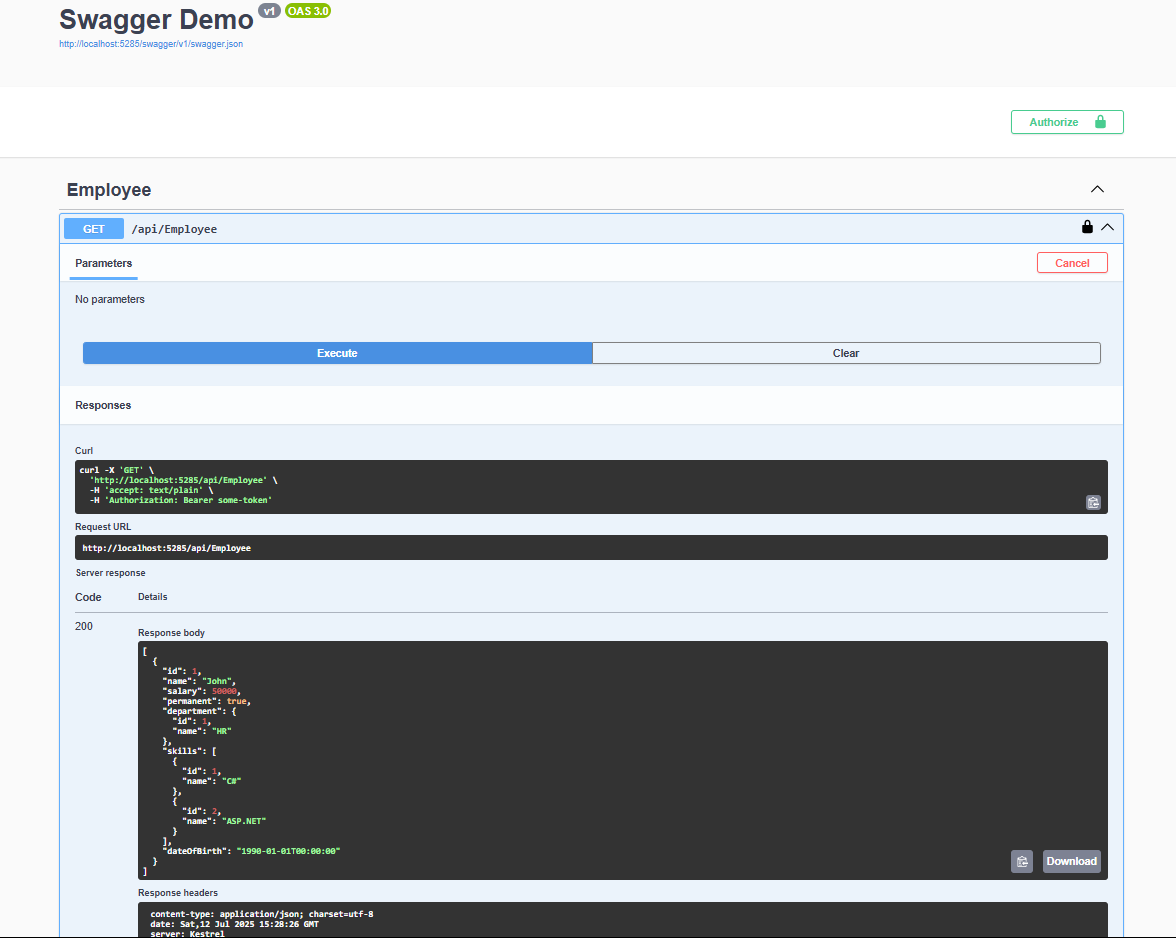
**Create a Custom action filter for Authorization.**



**Use POSTMAN to test the exception and message being thrown**

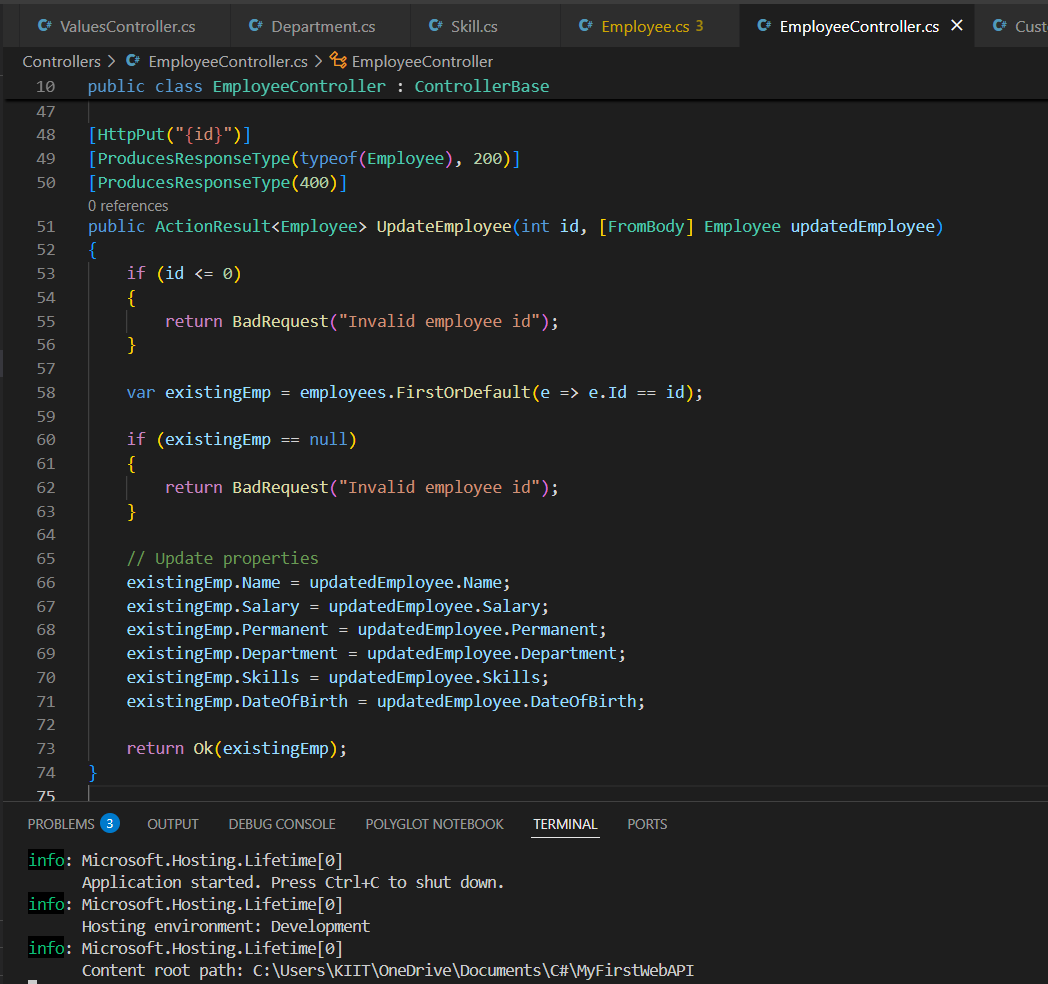


**Use Swagger to test the exception and message being thrown**

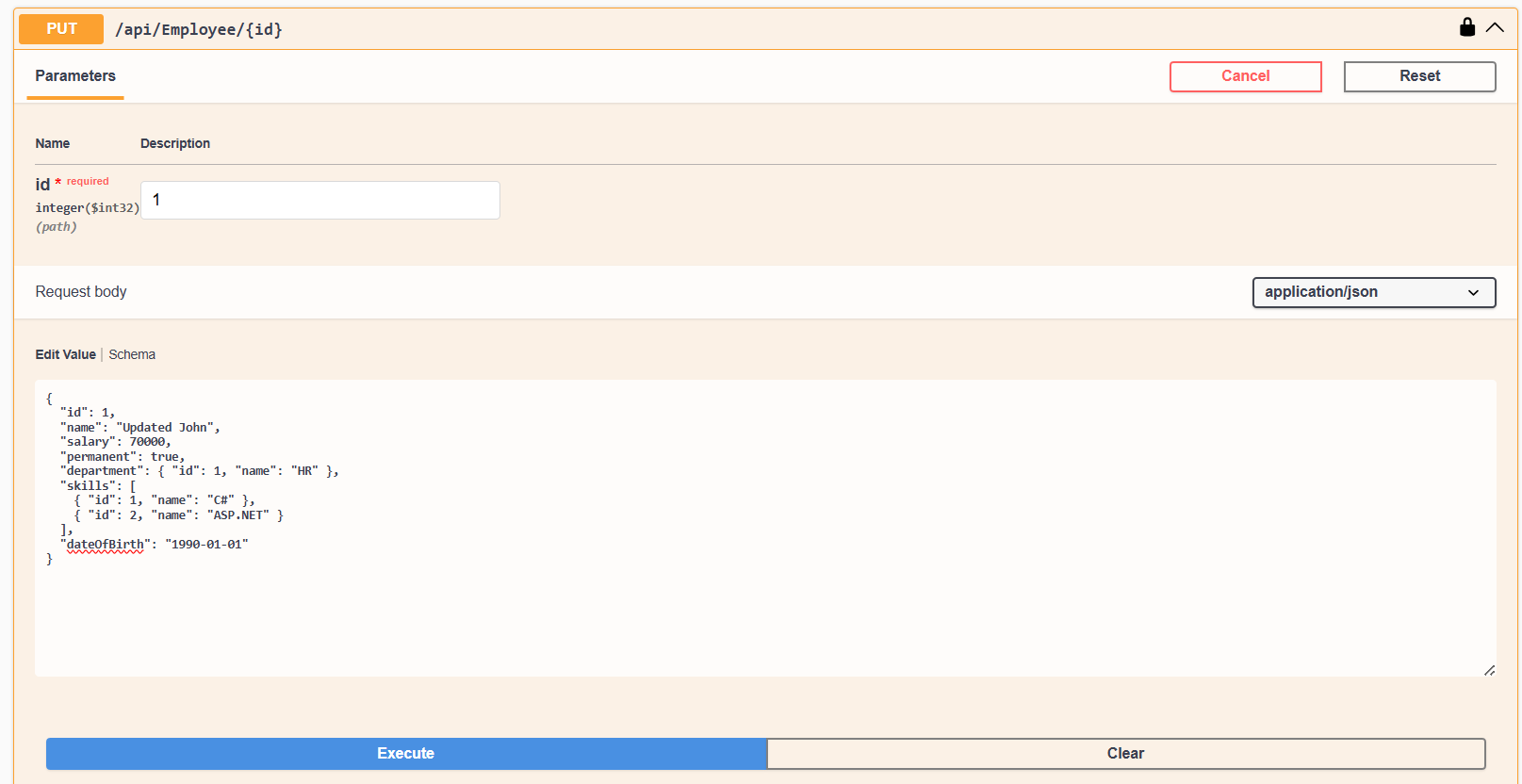


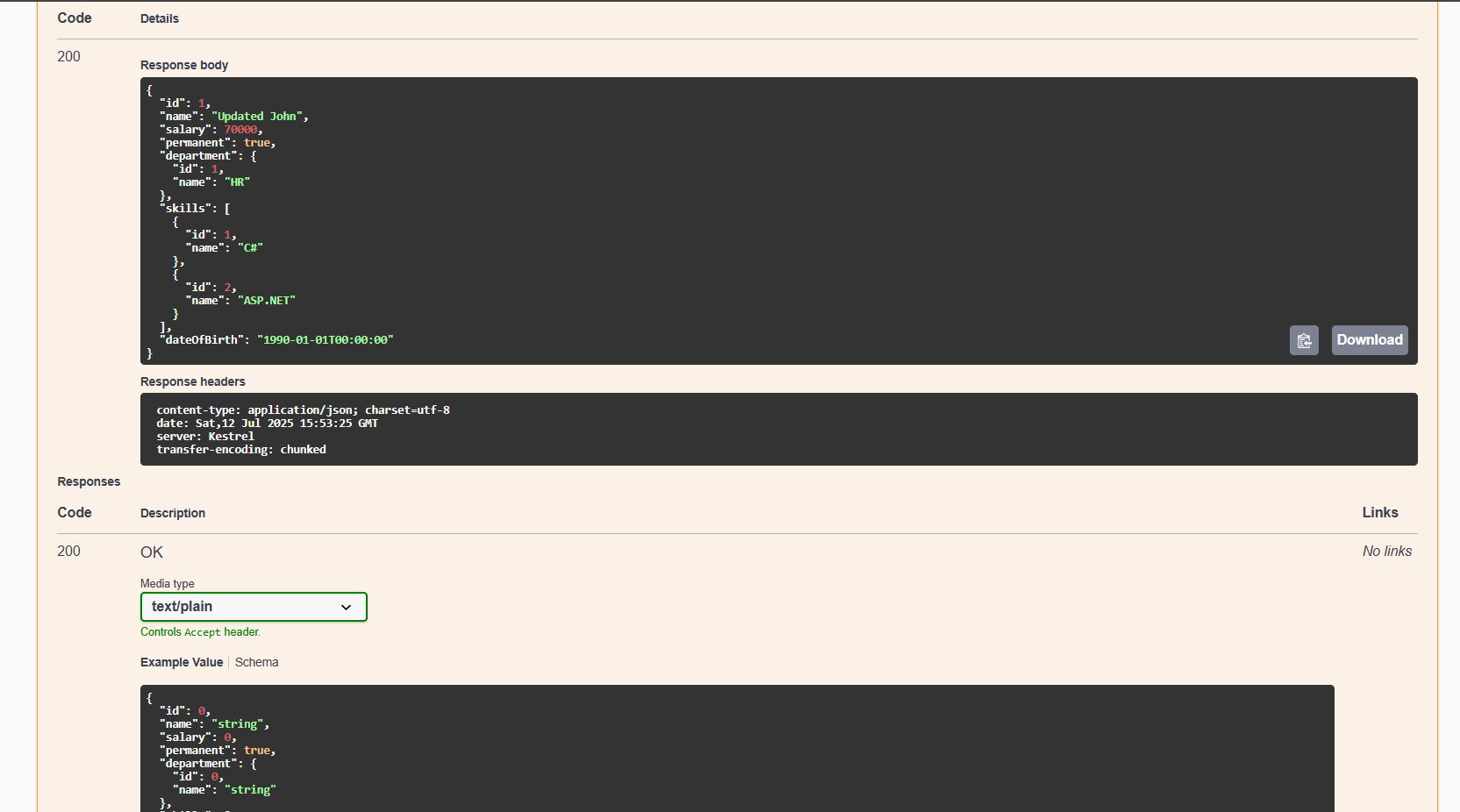
**4. Web Api CRUD operation**

**Updating Employee data as per the input thru Web API PUT action method call**

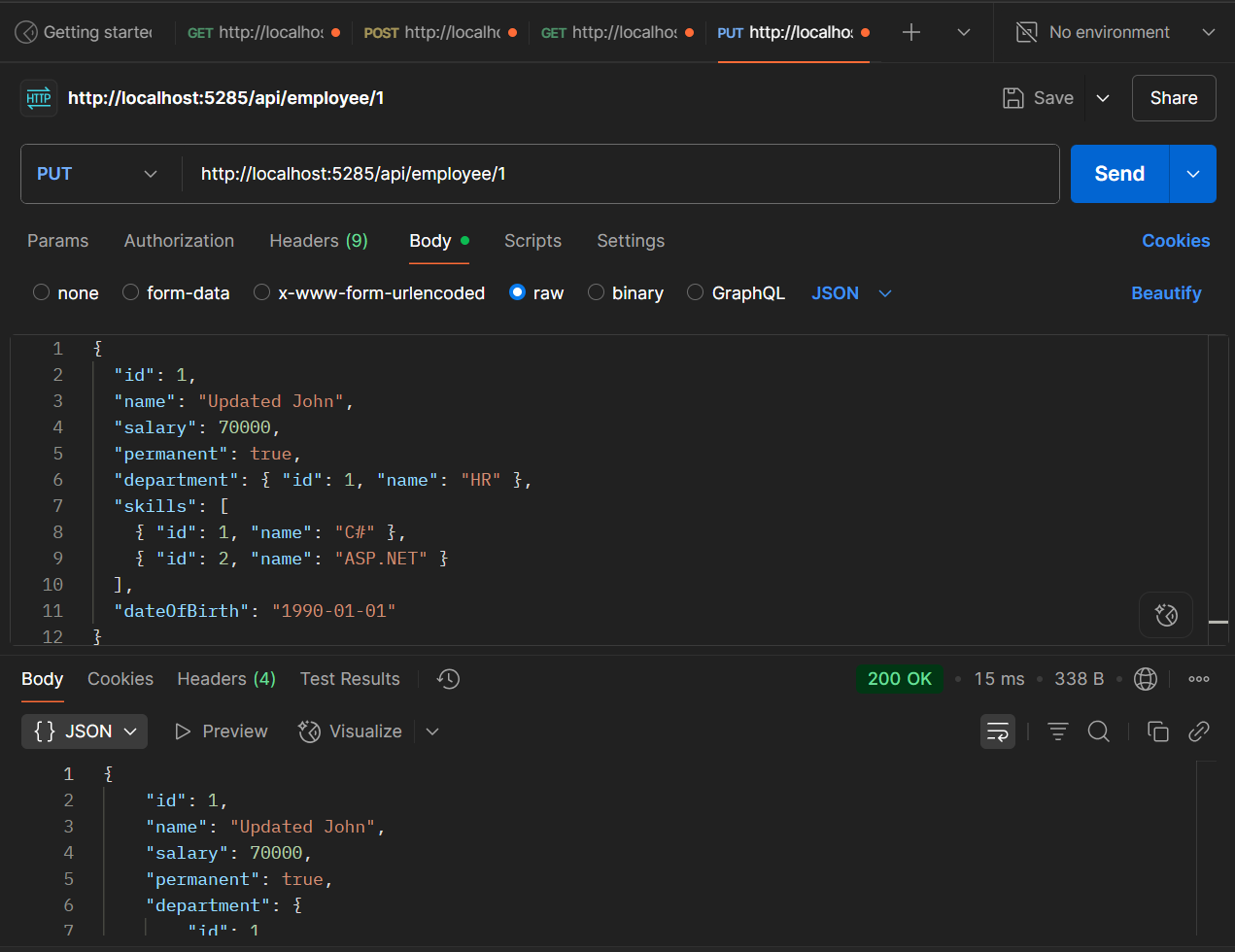


**Using Swagger tool to invoke the action method mapped with Http PUT action verb to update an employee data.**





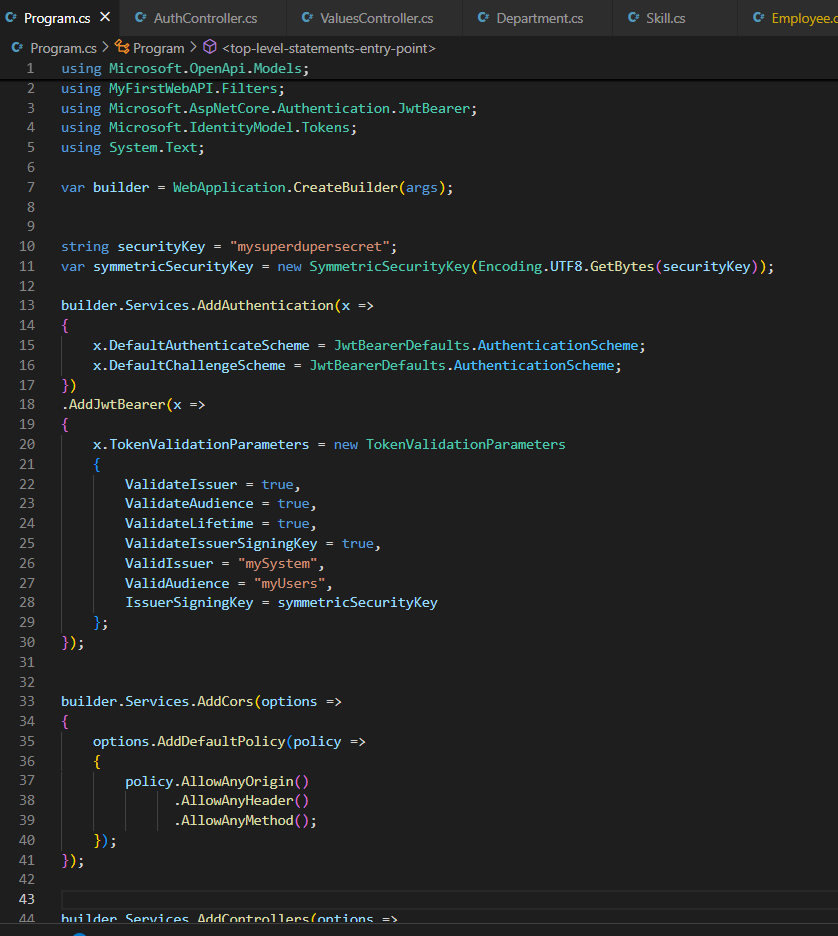
**Filtering the employee list data for the input id and return that as the output.**

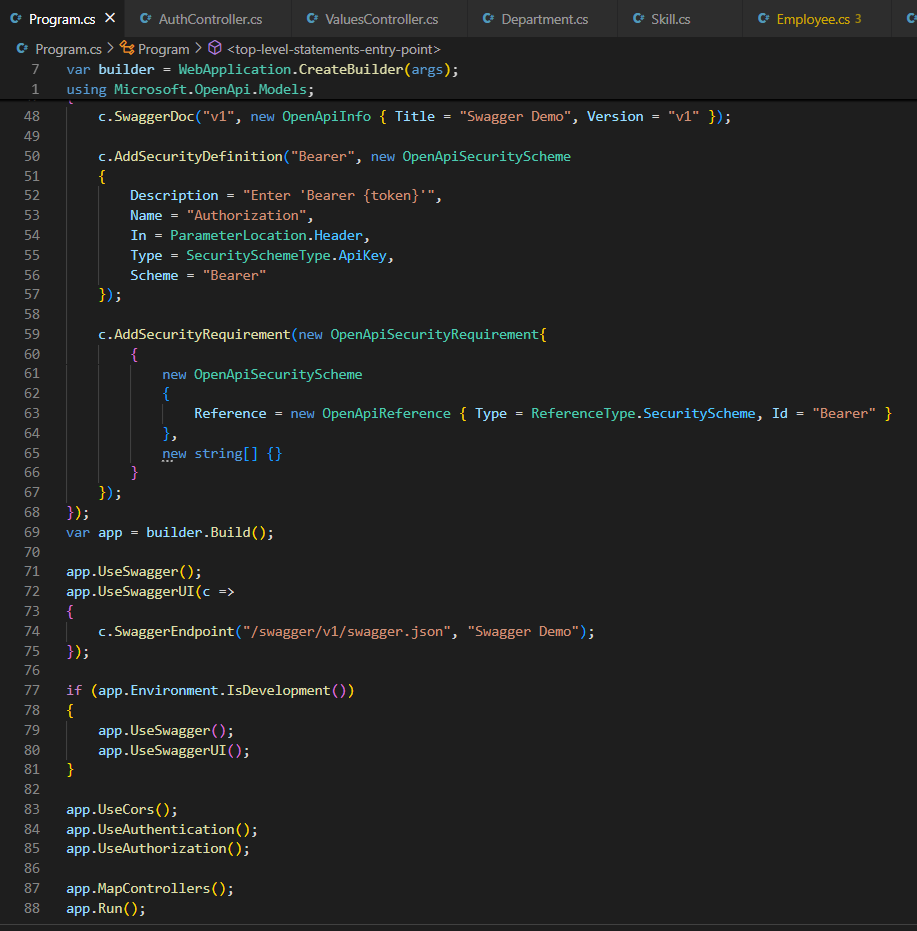


5. Explain CORS enablement for Web API access for local application

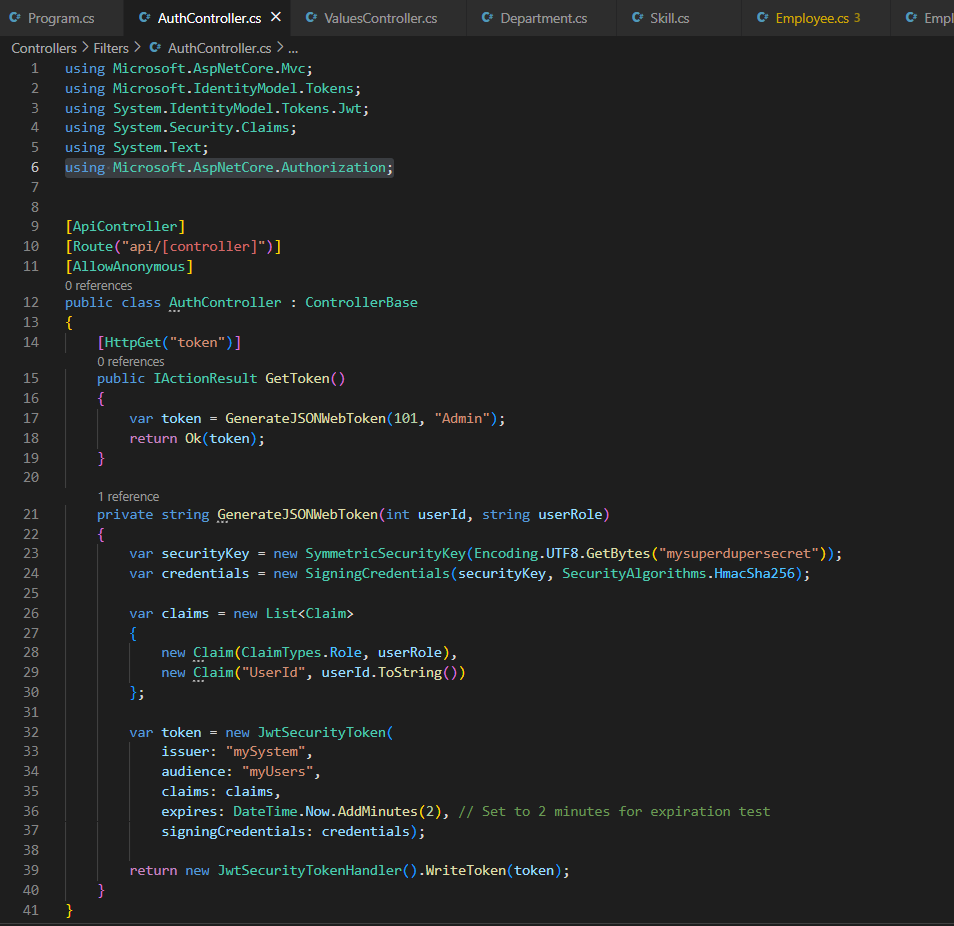
CORS (Cross-Origin Resource Sharing) is a security feature implemented by browsers that restricts web applications from making requests to a domain different from the one that served the web page. This means a frontend app running on http://localhost:3000 cannot access a Web API hosted on http://localhost:5285 unless CORS is explicitly enabled.

**JsonWebToken**

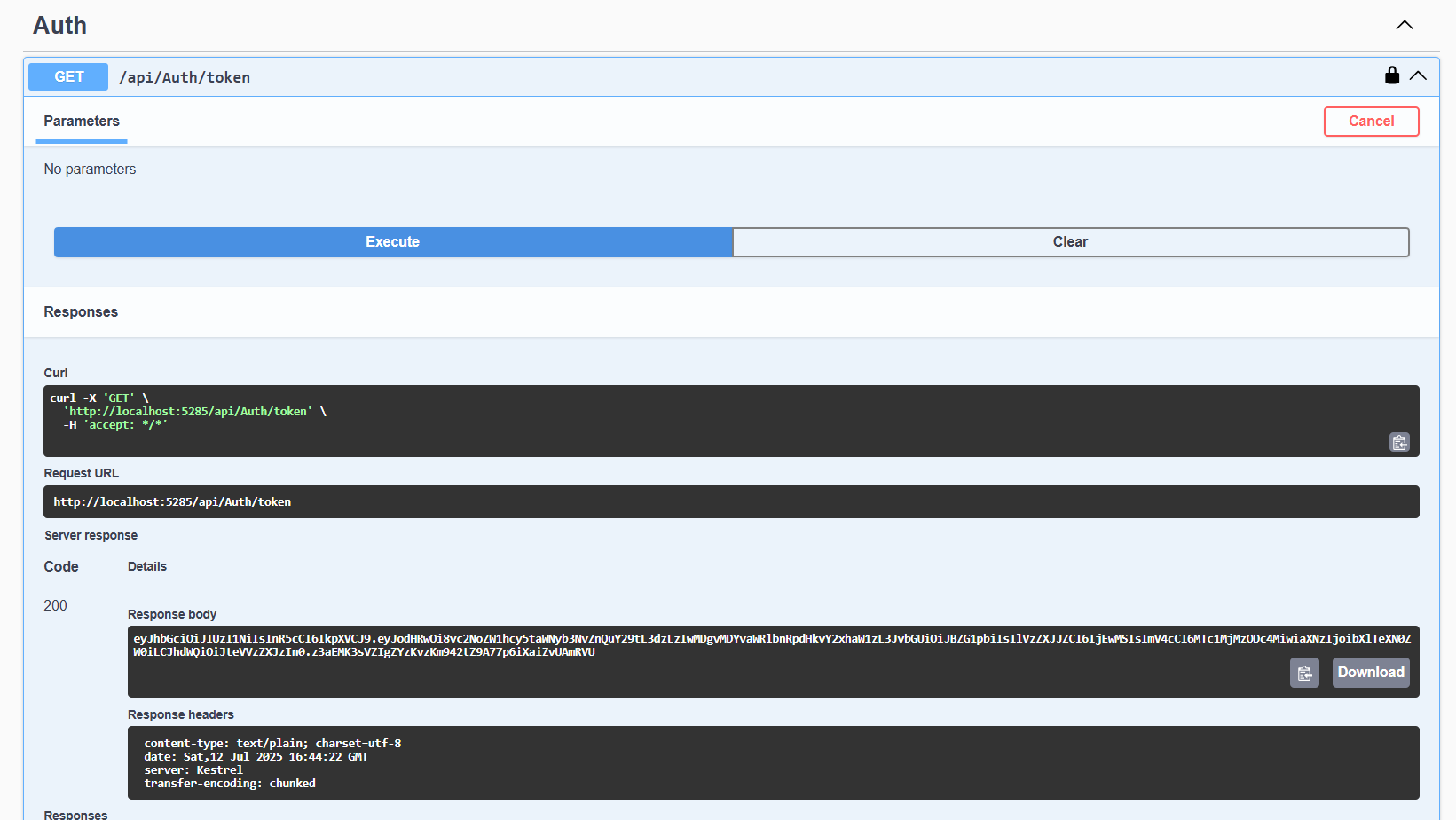


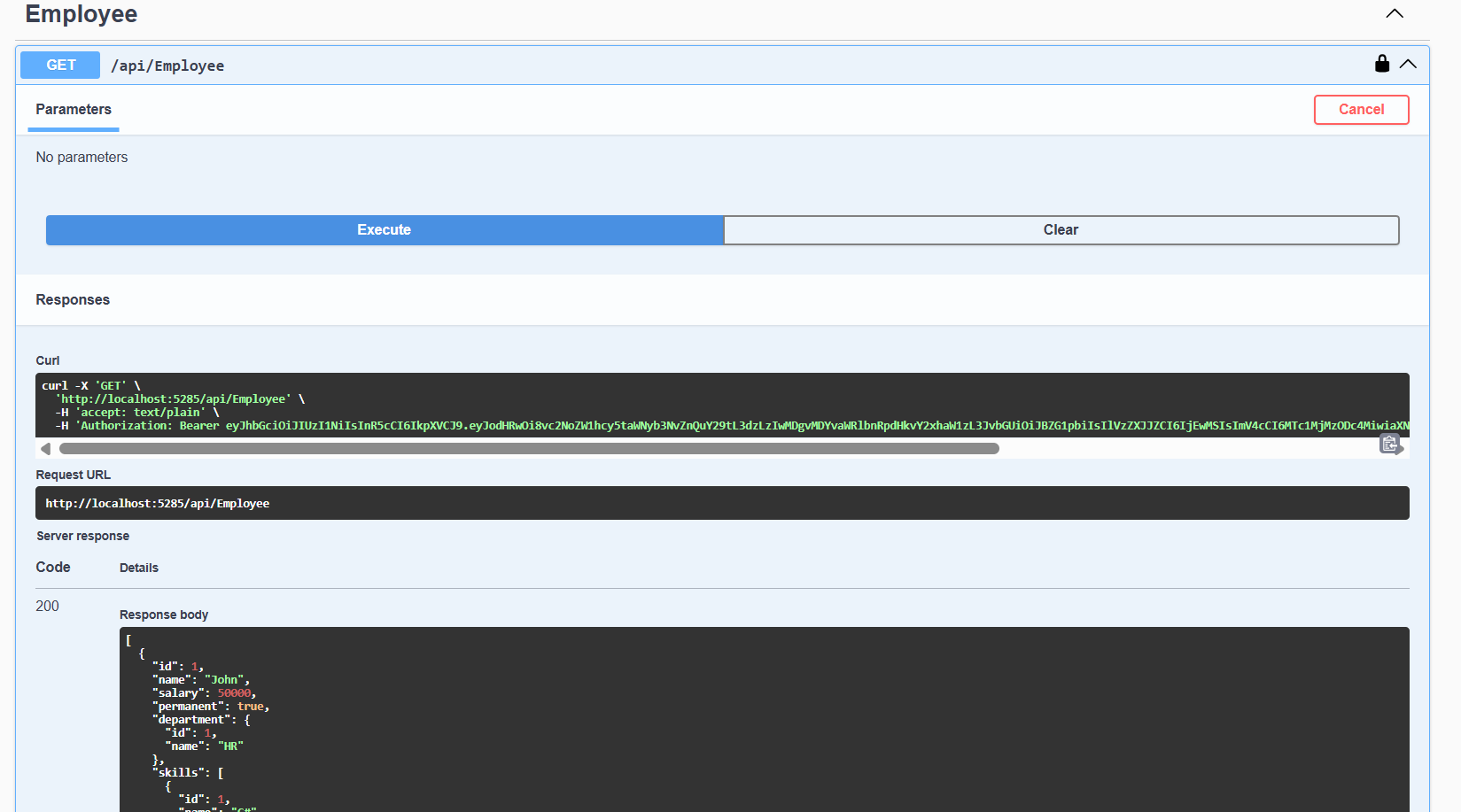


**Creating a new controller AuthController in the Web API application**.

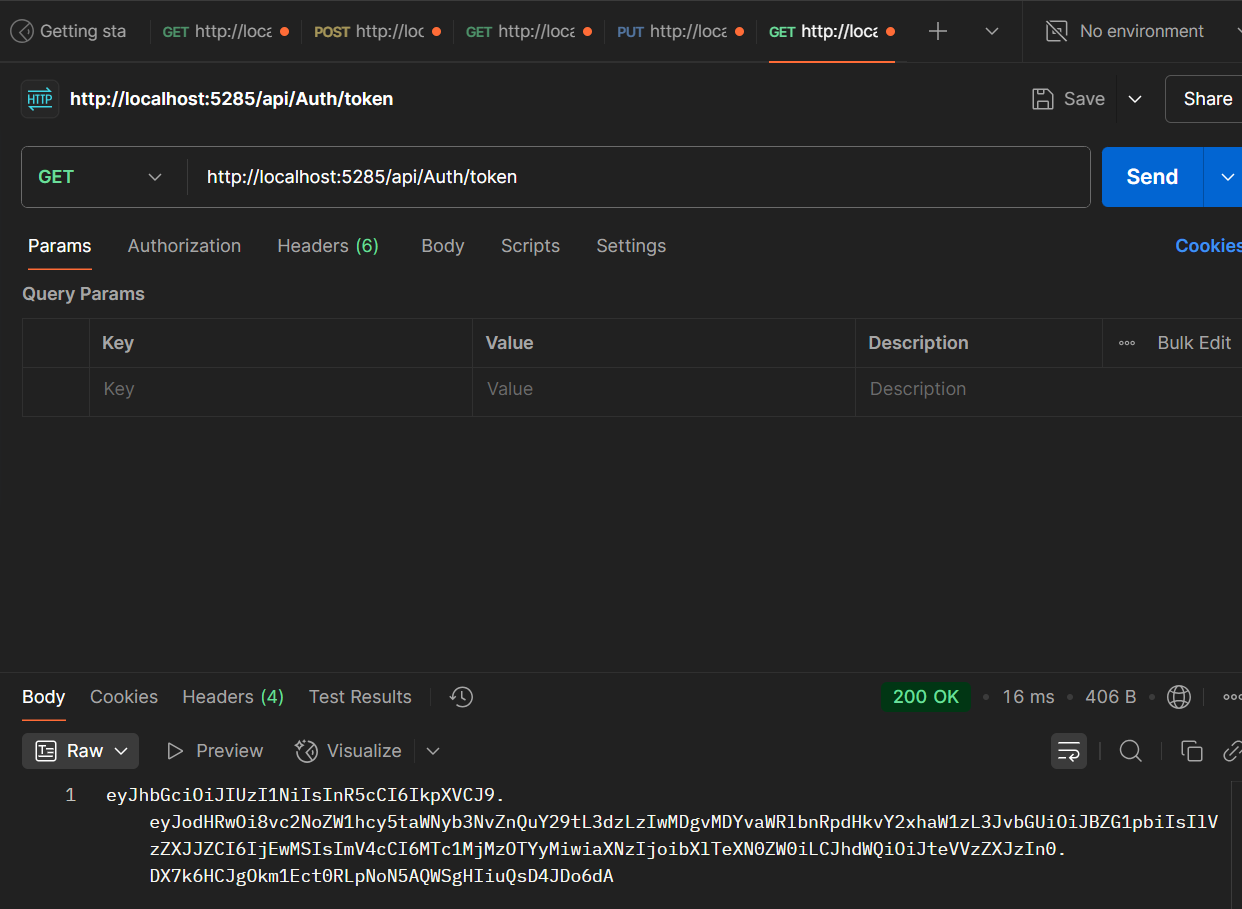


**CHECKING THE STATUS ON SWAGGER**

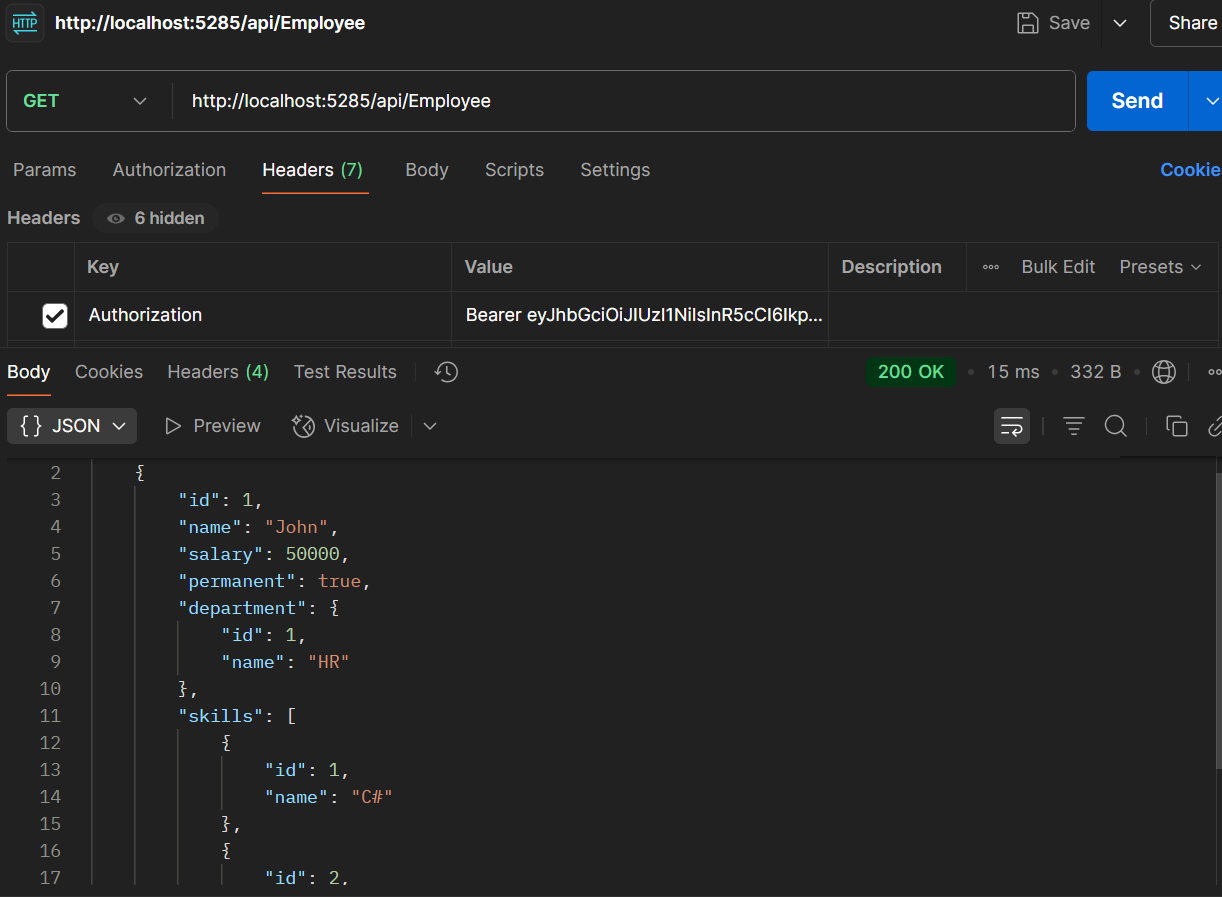




**TESTING ON POSTMAN**



**USING THE TOKEN IN AUTHORIZATION**



**TOKEN EXPIRY TEST**

In my AuthController.cs, token is set to expire after 2 minutes. Therefore, checking the status of api/Employee after 2 minutes.

This confirms the expiry logic is working correctly.

