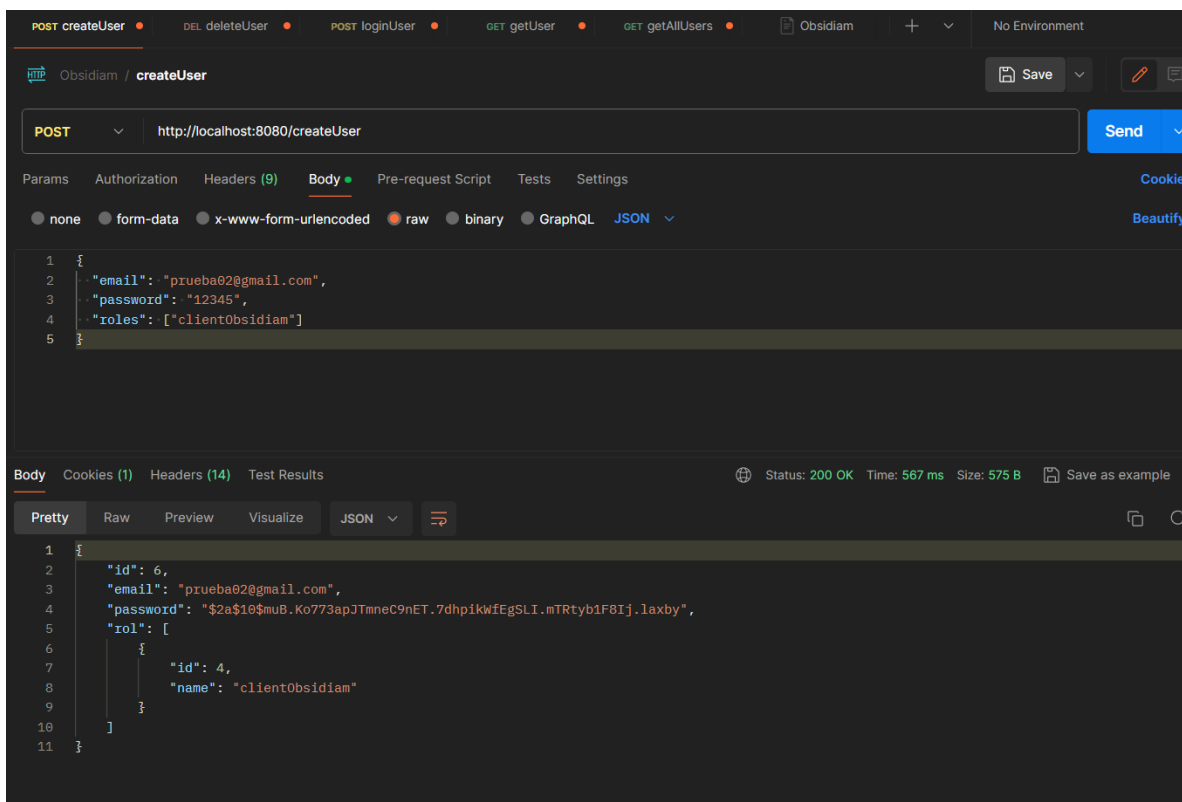


Tarea: Desarrollo de API RESTful en Java

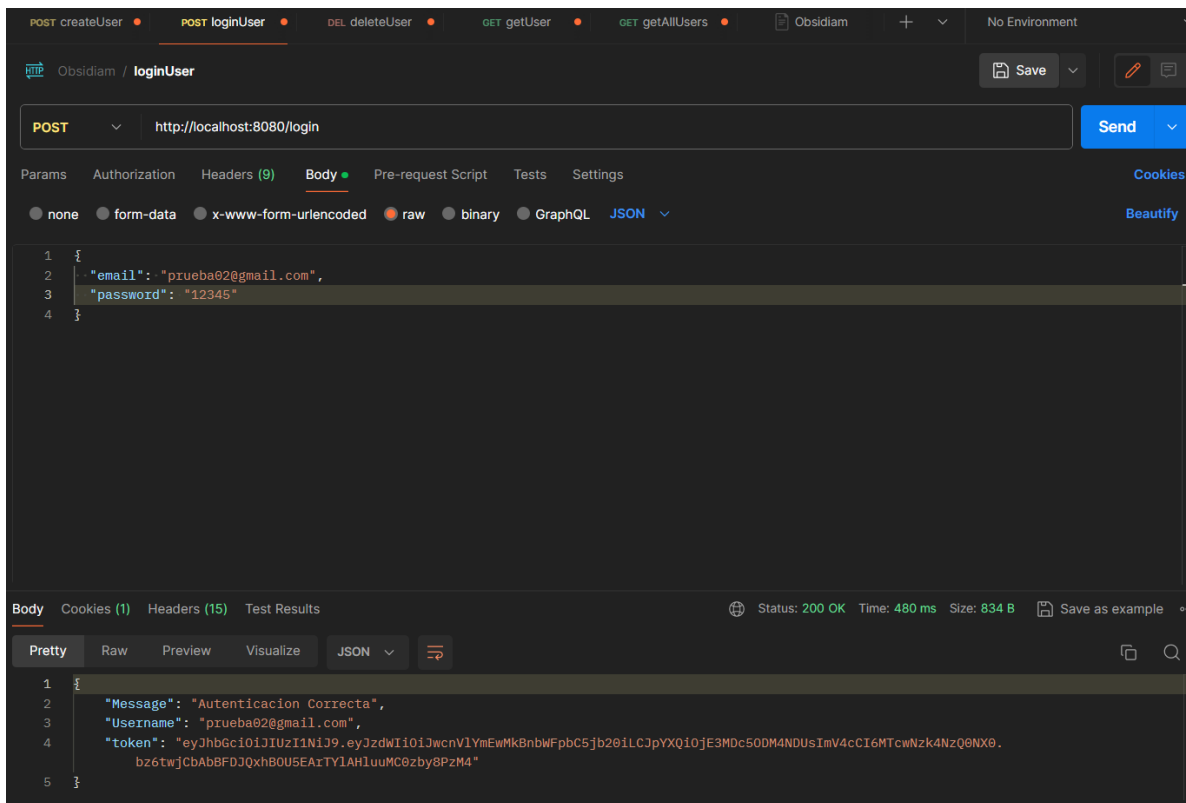
Para el desarrollo de la API se usó una arquitectura de microservicios con diversos componentes, entre ellos controladores para manejar las solicitudes HTTP, repositorios para interactuar con la base de datos, y servicios para manejar la lógica de negocio. Se implementó seguridad por medio de JWT para la autenticación y autorización de los usuarios. Se lograron manejar las excepciones cuando un Token JWT es inválido o cuando un usuario que se solicite no se encuentre en la base de datos. Por último, dentro de los desafíos que se afrontaron está la implementación de la seguridad ya que, en lo personal, no la había utilizado con el lenguaje de programación de Java, aun así, pude aprender bastante durante la implementación de esta seguridad. Se intentó desplegar el método de updateUser pero este presentó muchos conflictos, por tema de tiempo decidí comentarlo y continuar limpiando el resto del código.

Testeo del API:

- 1) Creación del usuario, la contraseña se devuelve para confirmar que está encriptada pero no es lo recomendado.



2) Se inicia con las credenciales de email y password para generar un JWT.



3) Se elimina el usuario con id 5, se debe ingresar el JWT generado para acceder al servicio.

Obsidian / deleteUser

DELETE http://localhost:8080/deleteUser?id=5

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description
id	5	

Body Cookies (1) Headers (14) Test Results

Status: 200 OK Time: 118 ms Size: 445 B Save as example

Pretty Raw Preview Visualize Text

```
1 Se borró el usuario 5
```

DELETE http://localhost:8080/deleteUser?id=5

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Type Bearer Token Token

eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJwcnVlYm...

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

- 4) Se lee el usuario con id 6, se debe ingresar el JWT generado para acceder al servicio.

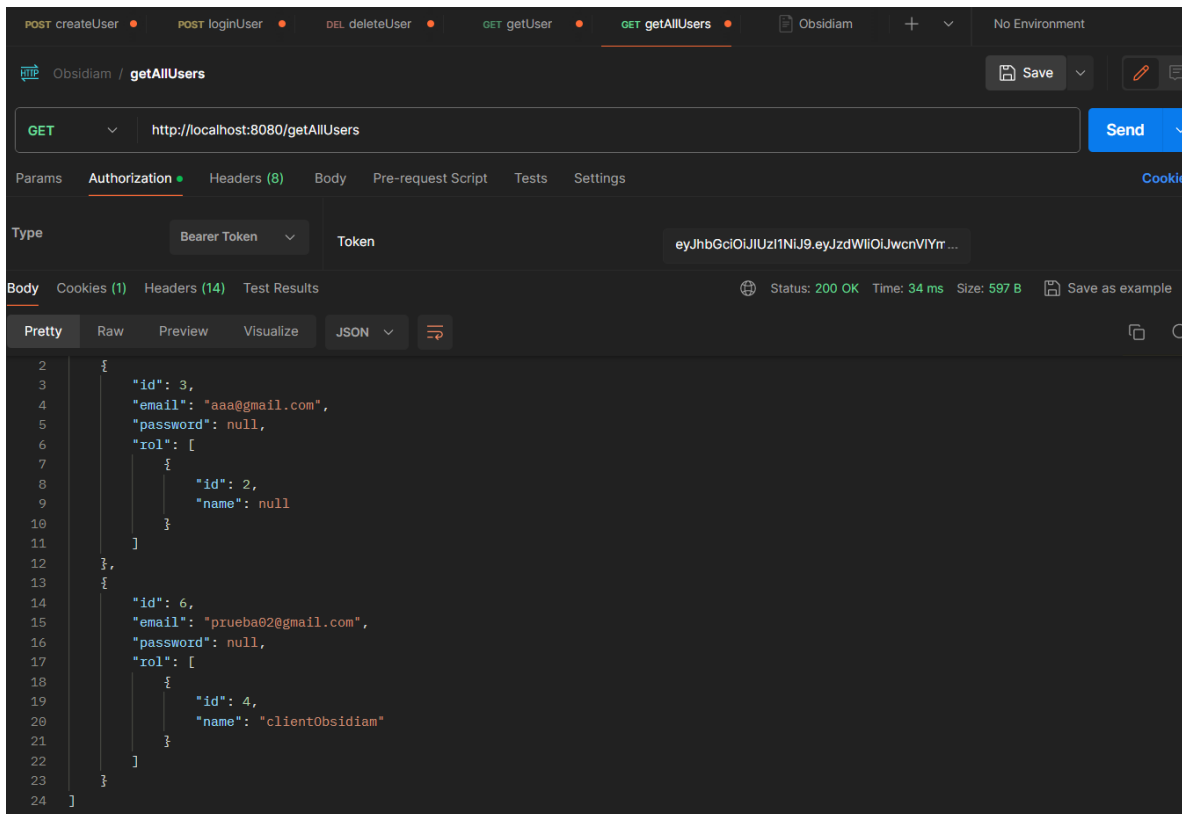
The screenshot shows the Obsidian API client interface. At the top, there's a header with several tabs: POST createUser, POST loginUser, DEL deleteUser, GET getUser (selected), GET getAllUsers, and a dropdown menu. Below the header, the URL bar shows 'http://localhost:8080/getUser?id=6' with a 'Send' button. The 'Params' tab is active, showing a table of query parameters.

Key	Value	Description
id	6	

Below the table, the 'Body' tab is active, showing the response in JSON format. The status is 200 OK, Time is 34 ms, and Size is 517 B. The JSON response is:

```
{
  "id": 6,
  "email": "prueba02@gmail.com",
  "password": null,
  "rol": [
    {
      "id": 4,
      "name": "clientObsidiam"
    }
  ]
}
```

- 5) Se leen todos los usuarios, se debe ingresar el JWT generado para acceder al servicio.



Adjunto la documentación de la API generado en Postman: [Obsidiam \(getpostman.com\)](https://getpostman.com/obsidiam)