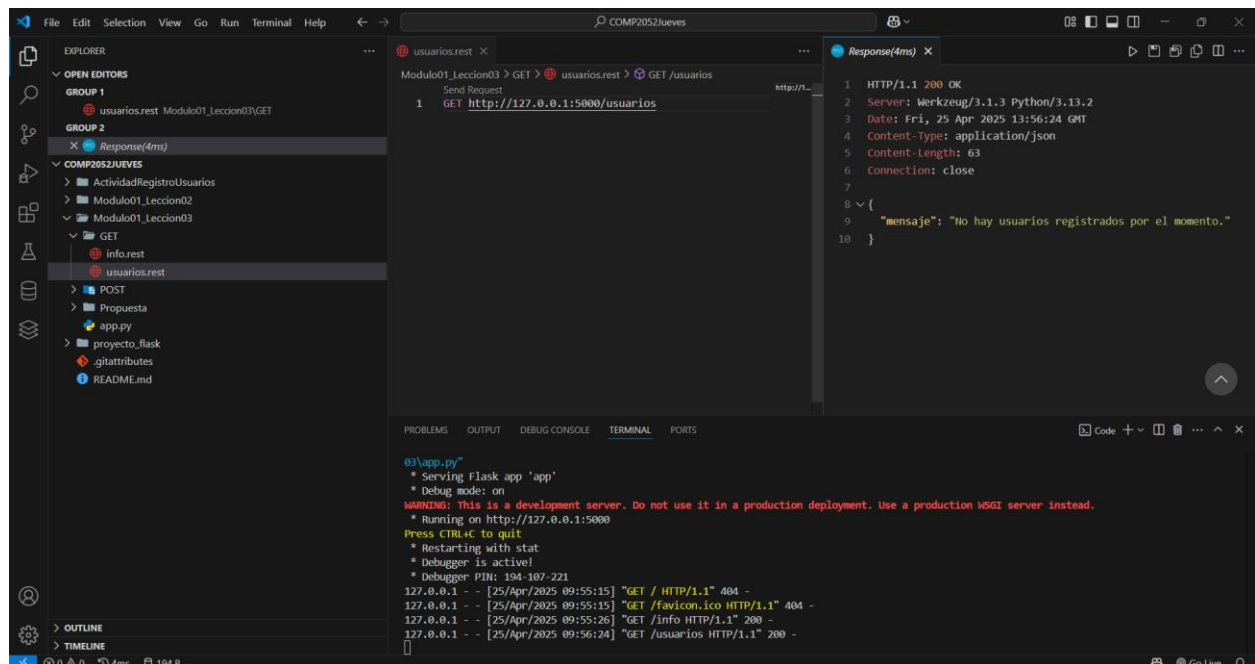


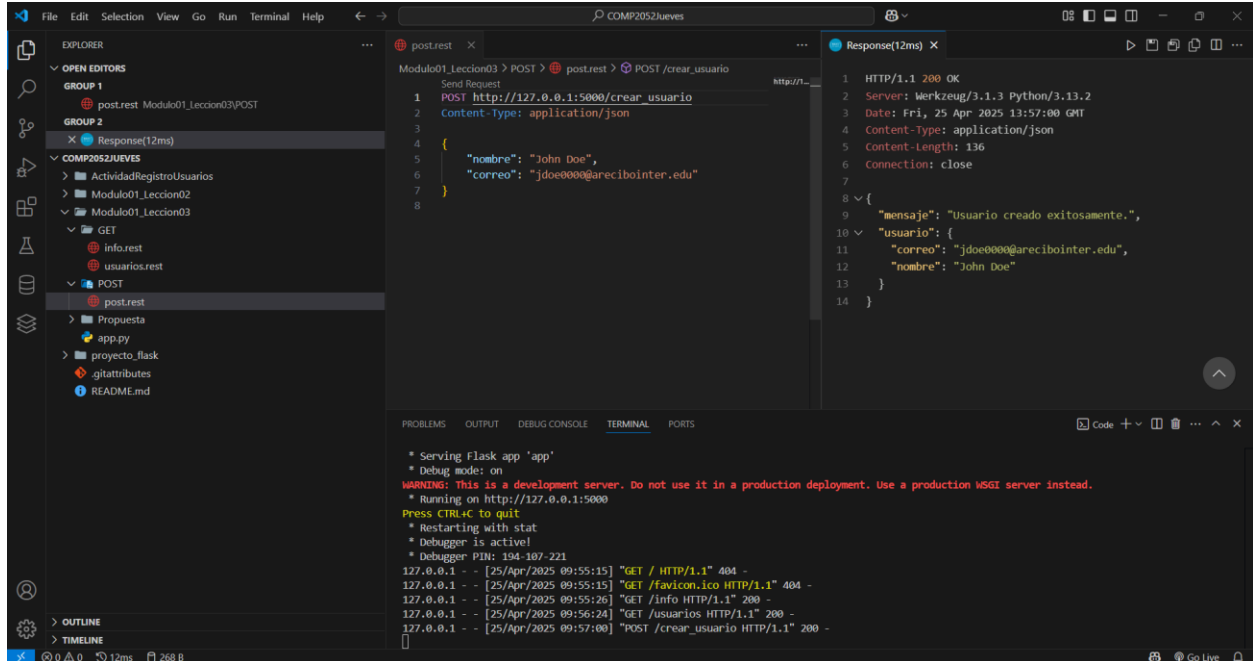
The screenshot shows the Visual Studio Code interface with the REST Client extension. In the Explorer, the file `Modulo01_Lecion03(GET)` is selected. The REST Client editor shows a GET request to `http://127.0.0.1:5000/info`. The response, labeled `Response(6ms)`, is displayed on the right. The response is an HTTP 200 OK with headers: `Server: Werkzeug/3.1.3 Python/3.13.2`, `Date: Fri, 25 Apr 2025 13:55:26 GMT`, `Content-Type: application/json`, and `Content-Length: 121`. The response body is a JSON object: `{ "descripcion": "Este sistema gestiona usuarios y productos.", "nombre": "Sistema de Gestion", "version": "1.0" }`. The terminal at the bottom shows the command `python -u "C:\Users\ayend\OneDrive\Documents\Github\COMP2052Jueves\Modulo01_Lecion03\app.py"` and the output of the Flask application, including a warning about using a development server and a list of log messages for the GET requests.

Se presenta un GET de parte del cliente para pedirle información a la base de datos a través del servidor flask y en la derecha se muestra la respuesta del GET.



The screenshot shows the Visual Studio Code interface with the REST Client extension. In the Explorer, the file `usuarios.rest` is selected. The REST Client editor shows a GET request to `http://127.0.0.1:5000/usuarios`. The response, labeled `Response(4ms)`, is displayed on the right. The response is an HTTP 200 OK with headers: `Server: Werkzeug/3.1.3 Python/3.13.2`, `Date: Fri, 25 Apr 2025 13:56:24 GMT`, `Content-Type: application/json`, and `Content-Length: 63`. The response body is a JSON object: `{ "mensaje": "No hay usuarios registrados por el momento." }`. The terminal at the bottom shows the command `python -u "C:\Users\ayend\OneDrive\Documents\Github\COMP2052Jueves\Modulo01_Lecion03\app.py"` and the output of the Flask application, including a warning about using a development server and a list of log messages for the GET requests.

Se presenta un GET el cual pide el listado de usuarios registrados y como no había ninguno le presenta un mensaje.



The screenshot shows the Visual Studio Code interface with a REST client extension. The Explorer panel on the left shows the project structure with folders for 'ActividadRegistroUsuarios', 'Modulo01_Lecion02', and 'Modulo01_Lecion03'. The 'Modulo01_Lecion03' folder is expanded, showing a 'POST' method. The main editor displays a REST client request for 'POST http://127.0.0.1:5000/crear_usuario' with a JSON body:

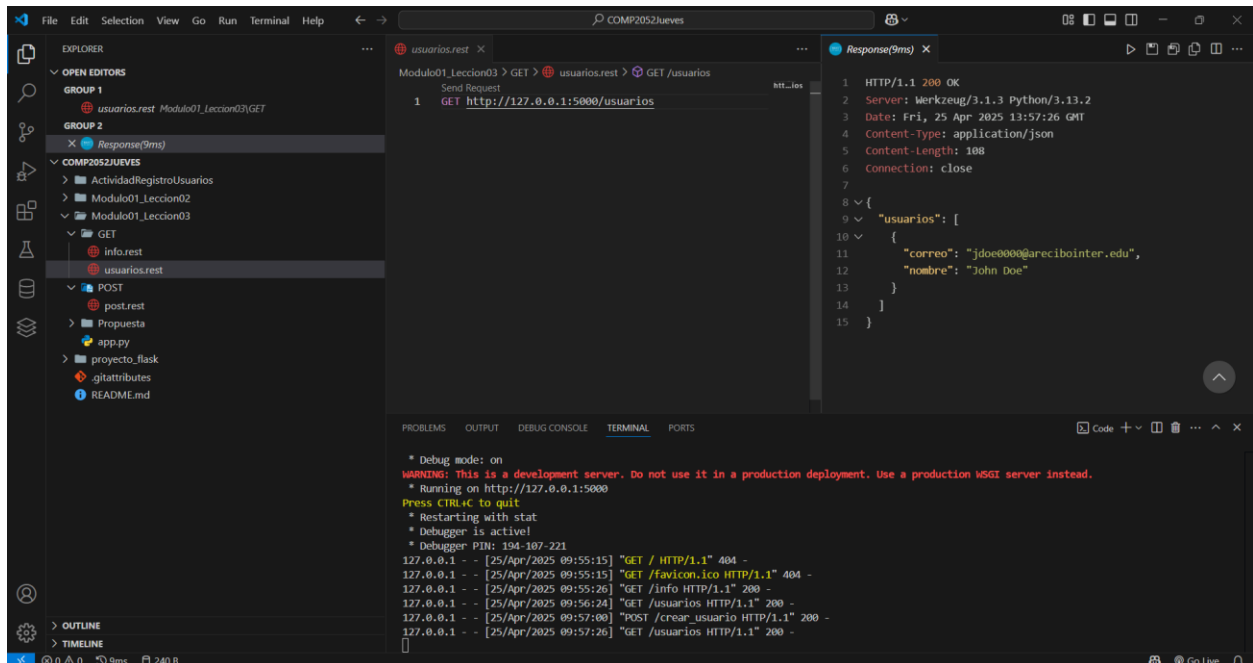
```
{  "nombre": "John Doe",  "correo": "jdoe0000@arecibointer.edu"}
```

. The response panel on the right shows a 200 OK status with a JSON body:

```
{  "mensaje": "Usuario creado exitosamente.",  "usuario": {    "correo": "jdoe0000@arecibointer.edu",    "nombre": "John Doe"  }}
```

. The terminal at the bottom shows the Flask application running on http://127.0.0.1:5000.

Aquí se utiliza un POST para insertar la información de un usuario a la base de datos por parte del cliente y al ingresarlo correctamente le aparece un mensaje.



The screenshot shows the Visual Studio Code interface with a REST client extension. The Explorer panel on the left shows the project structure with folders for 'ActividadRegistroUsuarios', 'Modulo01_Lecion02', and 'Modulo01_Lecion03'. The 'Modulo01_Lecion03' folder is expanded, showing a 'GET' method. The main editor displays a REST client request for 'GET http://127.0.0.1:5000/usuarios'. The response panel on the right shows a 200 OK status with a JSON body:

```
{  "usuarios": [    {      "correo": "jdoe0000@arecibointer.edu",      "nombre": "John Doe"    }  ]}
```

. The terminal at the bottom shows the Flask application running on http://127.0.0.1:5000.

Finalmente se regresa al GET del listado de usuarios y como ya hay uno le presenta el nombre y el email.

Jeremy A. Ayende Santiago
R00658578

<https://github.com/ayende10/actcomp2052.git>