Tabla de contenido

Entrega desafío técnico previred (Tecnova)

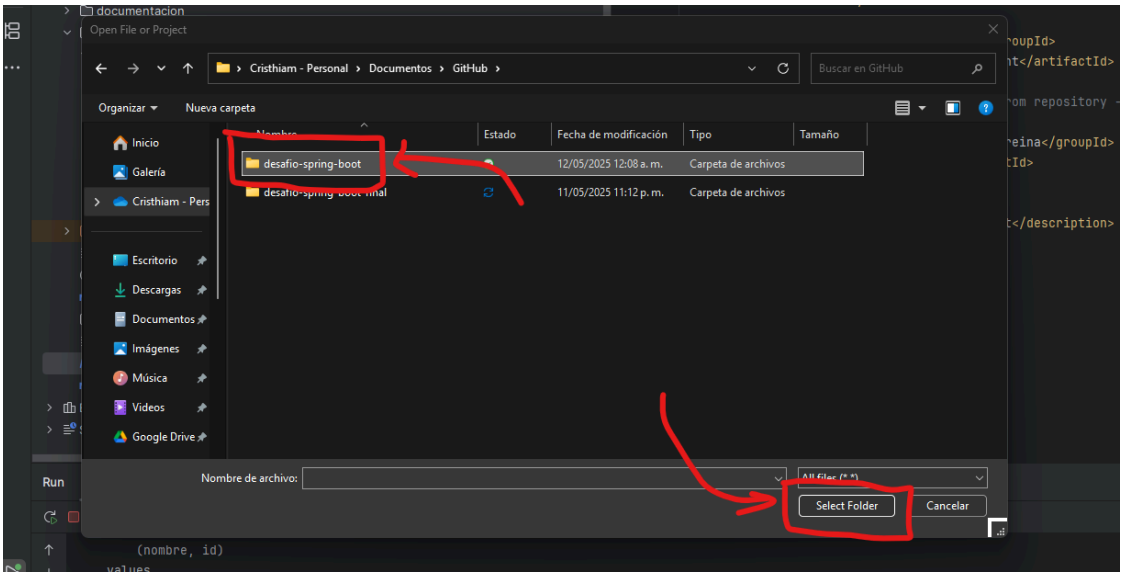
Estructura de carpetas del proyecto

- **desafio-spring-boot**
  - **documentacion** (Carpeta de documentacion)
    - **JavaDoc**
    - **Postman**
    - **Swagger**
  - **src.main.java** (Carpeta de archivos aplicacion)
    - **com.tecnova.previred.cristhiam.reina.desafio.spring.boot** (paquete root)
      - **config** (Paquete que contiene las clases de configuración de jwt)
      - **data** (Paquete con las clases que insertan datos en db)
      - **exception** (Paquete con excepciones personalizadas)
    - **mapping** (contiene las interfaces de las funciones de mapping entre entity a model y viceversa)
      - **impl** (Contiene las clases con las funciones de mapping)
    - **repository** (contiene interfaces relacionadas con el manejo de datos en base de datos con spring data jpa)
      - **Entity** (Contiene clases entity de base de datos)
    - **rest.impl** (contiene las clases que implemente las interfaces generadas a partir de archivo openapi.yml con la librería openapi-generator para la gestión de los servicios)
  - **src.main.resources** (carpeta que contiene el archivo application.properties que contiene la configuracion de la aplicacion y el archivo openapi.yml con que la libreria openapi-generator, genera las interfaces para gestión de los servicios)

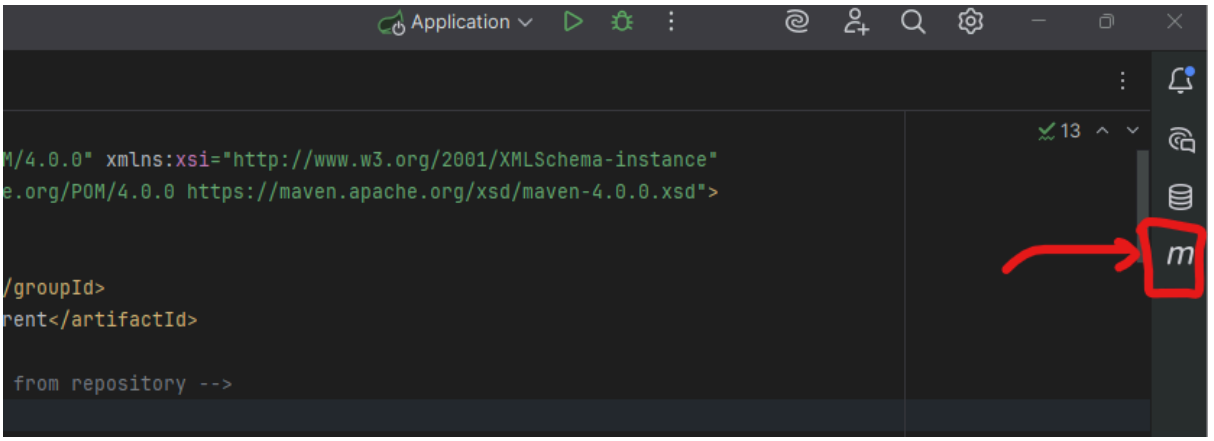
Abrir y desplegar el proyecto.

Para ejecutar el proyecto debemos utilizar el ide IntelliJ y a continuación detallamos su despliegue.

1. Abrimos IntelliJ, luego buscamos el botón con 6 rayas, para desplegarlo, buscamos la opción open y se despliega el explorador de archivos y buscamos la carpeta del proyecto desafio-spring-boot y le damos click y luego damos click al botón Select Folder.

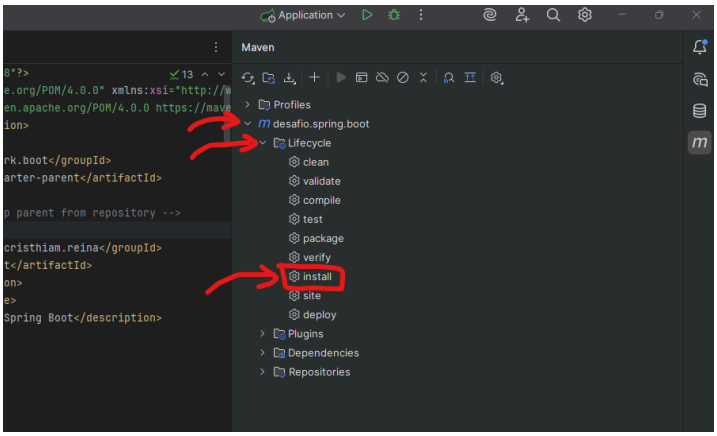


2. con el proyecto abierto en el ide buscamos el boton maven y le damos clic.

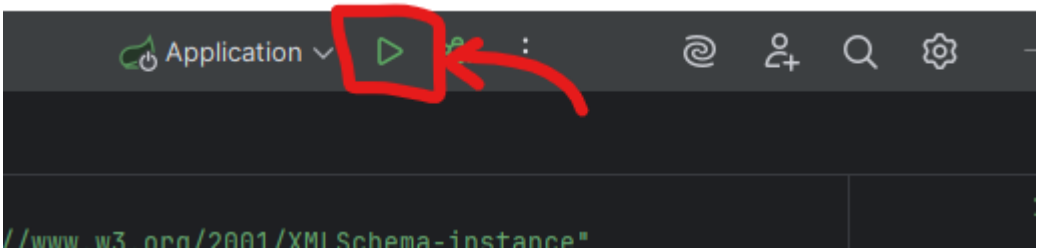


3. Se despliega la sección de tareas de maven, buscamos desafio.spring.boot y al lado derecho del nombre hay una flecha y le damos clic, se despliega en la primera opción Lifecycle buscamos la flecha que se encuentra al lado izquierdo para desplegar las opciones y buscamos la tarea install, esperamos a que termine la

tarea ejecutada.



4. Buscamos el botón de play que se encuentra arriba al lado izquierdo y le damos clic para desplegar el proyecto.



Documentación

En la carpeta **documentacion** se encuentran, tres carpetas, **JavaDoc** que contiene el documento con con la descripción de las clases e interfaces que compone el proyecto, **Postman** que contiene el json con la configuración para ejecutar los servicios del proyecto y la carpeta **Swagger,** con el archivo **openapi.yml** que contiene la definición de los servicios, archivo que también se encuentra en la carpeta **src.main.resources.**

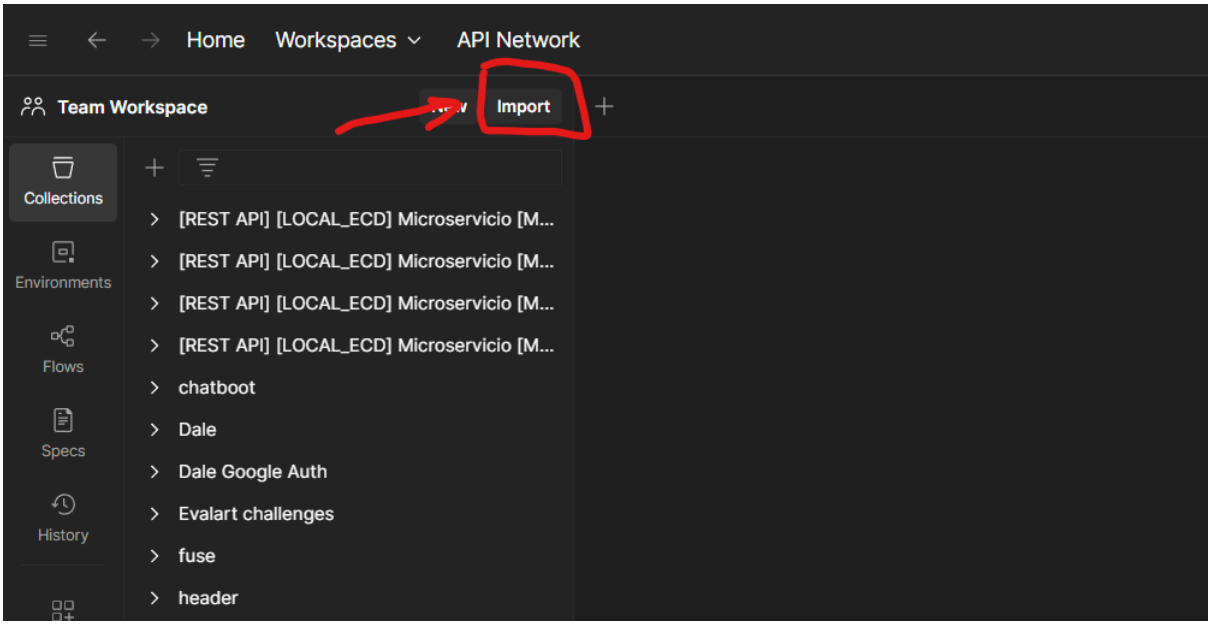
Con el proyecto desplegado en intelij, abrimos un navegador y entramos a la siguiente url para abrir el **swagger** de la aplicación.

1. http://localhost:8080/swagger-ui/index.html

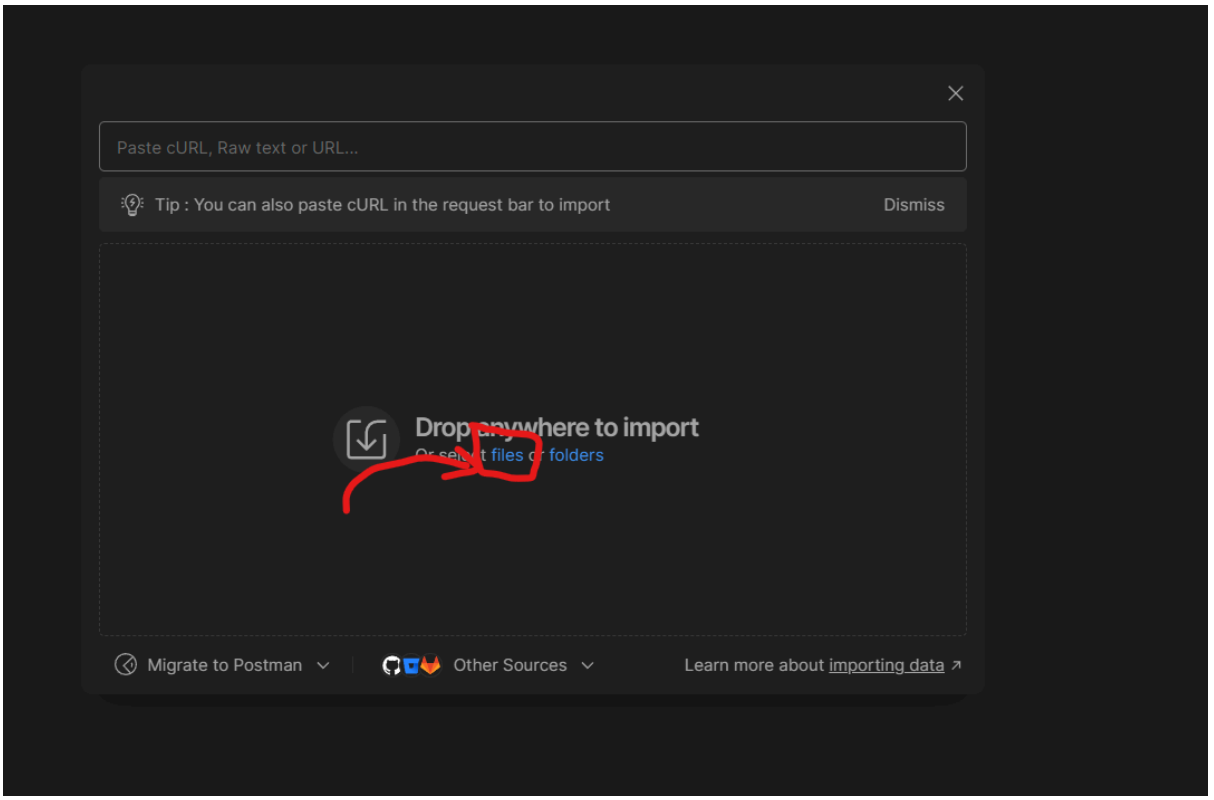
Configuración Postman

Para probar los servicios se envía la colección en **postman**, a continuación se describe el procedimiento para configurar el json en postman.

1. Abrir postman.
2. Ubicar el boton Import que se encuentra en la parte de arriba a la izquierda de postman.



3. Aparece una ventana flotante para arrastrar el archivo de json que contiene la definición de los servicios, buscamos la palabra files y damos click ahi.

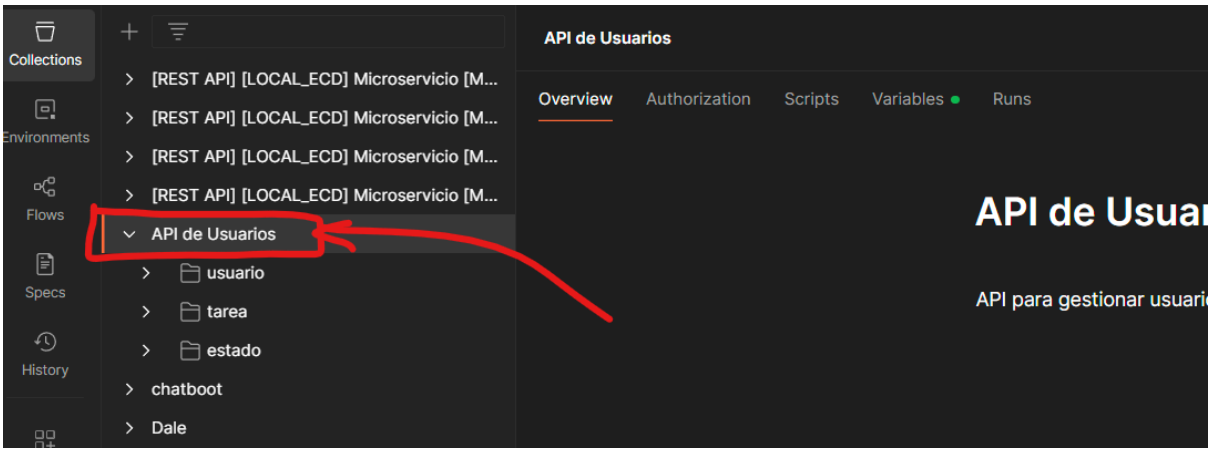


4. Se despliega la ventana de explorador de archivos, buscamos la carpeta del proyecto, entramos a la carpeta **documentacion,** entramos a la carpeta **Postman** y damos click en el archivo API de Usuarios.postman_collection.json.
5. El programa carga la colección y al terminar al lado derecho debe aparecer una colección llamada **Api de Usuarios,** y terminamos de configurar la colección en postman.

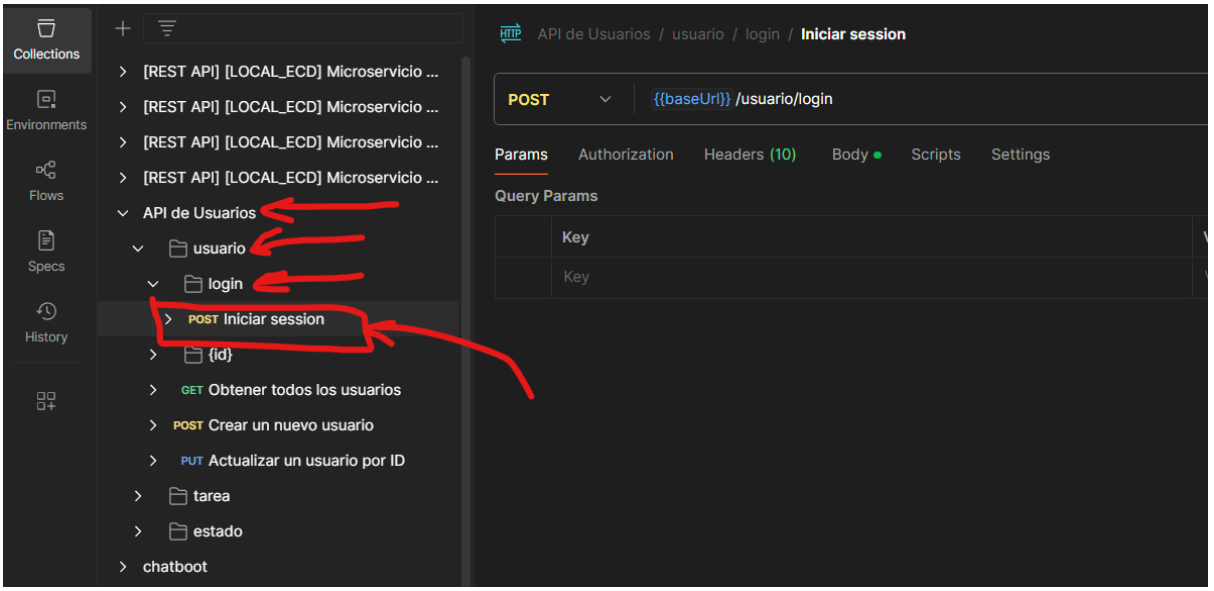
Iniciar sesión en los servicios

Para iniciar sesión debemos entrar a la colección API de Usuarios de servicios de **postman** que se configuró en la sección anterior.

- Buscamos API de Usuarios y lo desplegamos.

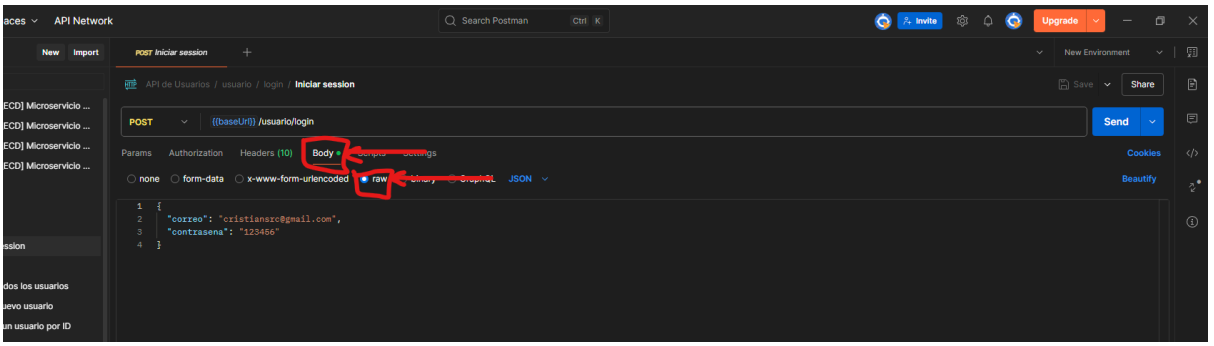


- Desplegamos la carpeta **usuario,** en las opciones que se despliegan, desplegamos la opción **login** y damos clic en **iniciar session.**

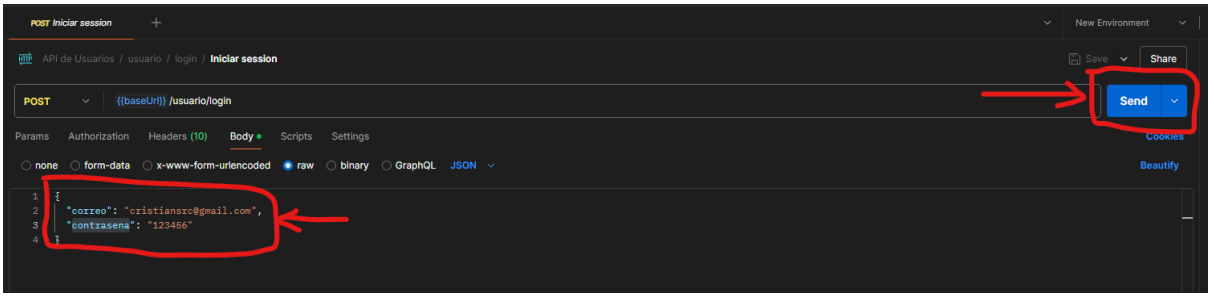


- Se despliega la pantalla del servicio de iniciar sessionen la parte del centro de la pantalla, arriba, observamos el nombre del servicio y abajo hay unas opciones y seleccionamos la opción **Body**, luego se despliega otras opciones y seleccionamos

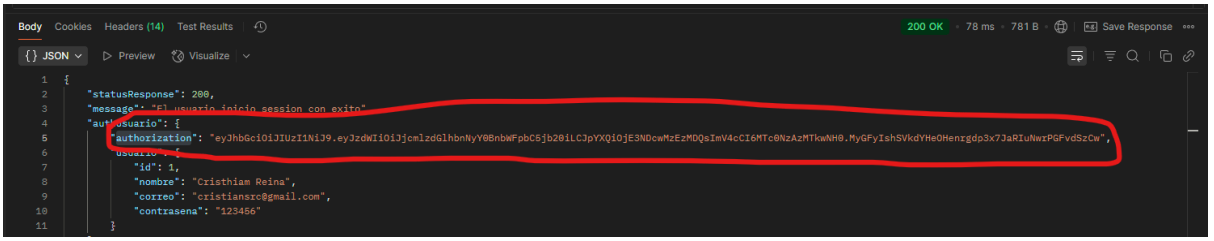
la opción **raw**.



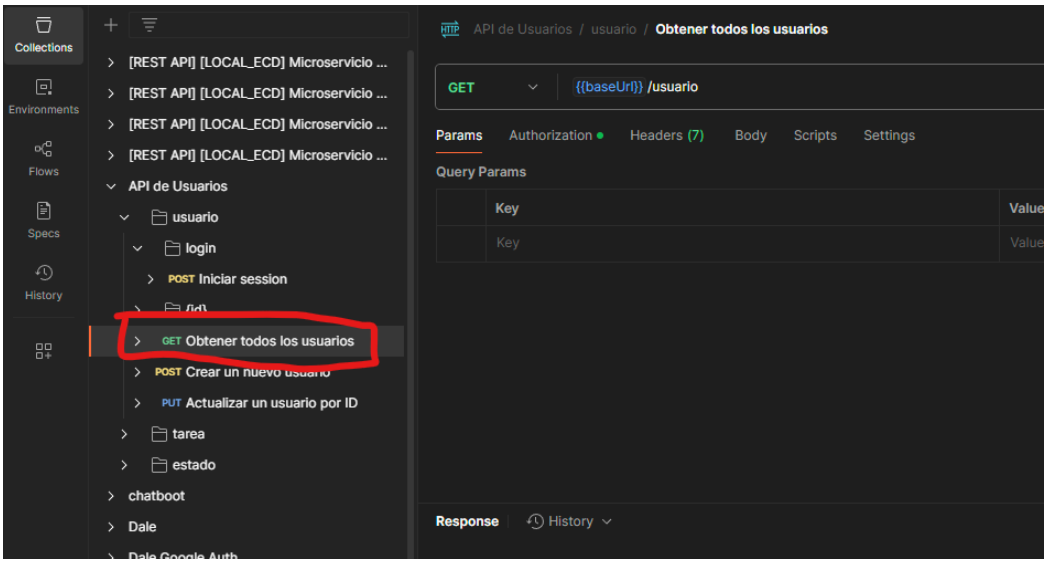
- Luego un poco más abajo, encontramos el json de entrada del servicio con dos datos, “correo” y “contrasena”, los datos que encontramos ahí nos sirve para inciar sesión, de necesitar probar con otros usuario, dejo a continuación otros usuario para las pruebas.
    - correo -> agarcia@gmail.com , contraseña -> 123456
    - correo -> cgutierra@gmail.com , contraseña -> 123456
    - correo -> ggoimez@gmail.com , contraseña -> 123456
    - correo -> cristiansrc@gmail.com , contraseña -> 123456
- A continuación colocamos los datos con los queremos probar en el json y le damos en Send.



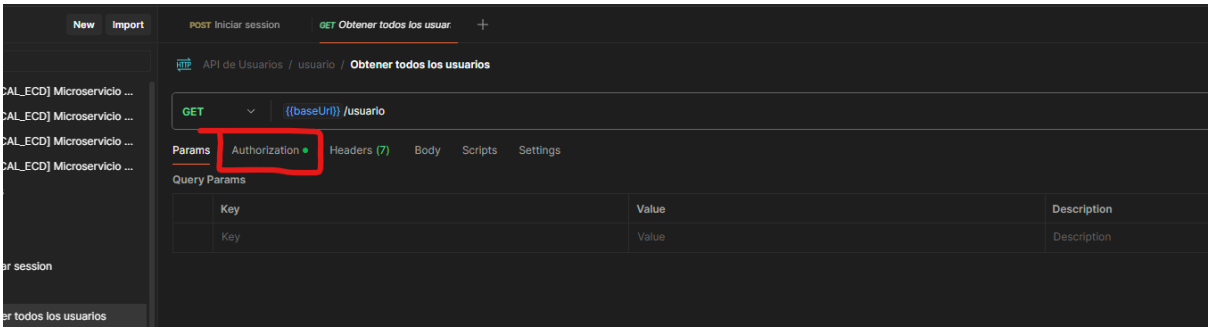
- Luego de realizar la petición del servicio postman nos pinta la respuesta, del json de respuesta buscamos el campo **authorization** y copiamos su valor.



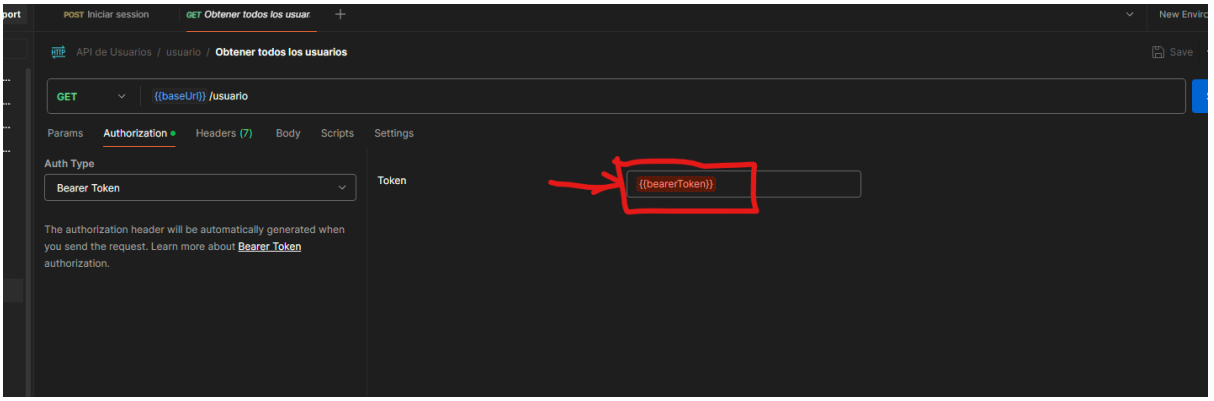
- A continuación vamos al servicio, obtener todos los usuario y le damos click.



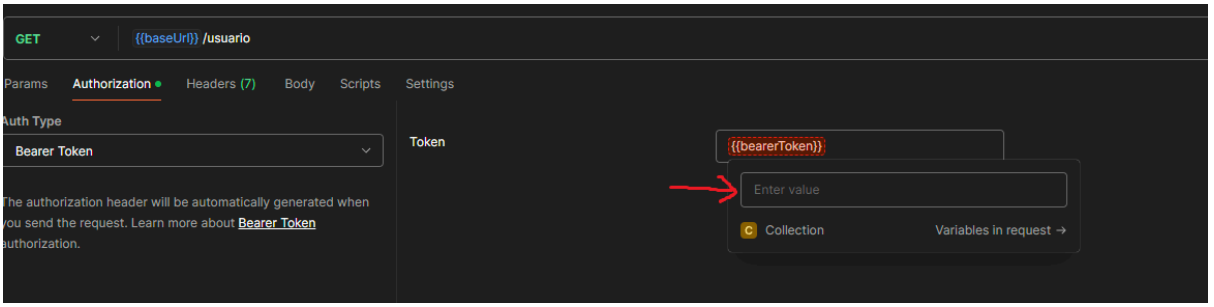
- al lado derecho arriba, buscamos la opción **Authorization** y damos clic.



- Se despliega la configuración de **Authorization** buscamos la variable {{bearerToken}} y le damos clic.



- Se despliega una pequeña ventana, ponemos el cursor en la parte donde dice **Enter value** y ahi pegamos el token generado en la petición anterior**.**



- Con eso todos los servicio de la API de Usuario se podrán ejecutar con ese token, caso de vencer el token, pueden realizar el mismo procedimiento de nuevo, luego vamos al botón **Send** para realizar la petición al servicio.