UNIVERSIDAD NACIONAL DE LA PLATA

MAESTRÍA EN INGENIERÍA DE SOFTWARE

Trabajo Practico Final

**Asignatura**: Tópicos de Ingeniería de Software II

Año: 2023

**Título**: Trabajo Integrador: Servicio Web (API) – Predicción de Riesgo Cardíaco

**Autor**: Karina Policano

**Docentes**:

Prof. Andrés Diaz Pace

Prof. Claudia Pons

Prof. Gabriela Pérez

Prof. Matías Urbieta

Contenido

[Requisitos Obligatorios para Ejecutar el Proyecto 3](#_Toc164002244)

[Pasos del Proyecto 3](#_Toc164002245)

[Servicios ofrecidos: 5](#_Toc164002246)

[Conexión entre servicios: 5](#_Toc164002247)

[Decisiones de Diseño 5](#_Toc164002248)

[Uso de Redis: 5](#_Toc164002249)

[Funciones Implementadas 6](#_Toc164002250)

[Servicio *predictor* 6](#_Toc164002251)

[validartiempo: 6](#_Toc164002252)

[predecir\_conTiempo: 6](#_Toc164002253)

[validarParametros: 6](#_Toc164002254)

[Programas Python del Proyecto 6](#_Toc164002255)

[Para la ejecución: 6](#_Toc164002256)

[Caso de prueba 10](#_Toc164002257)

[Servicio altaUsuario 11](#_Toc164002258)

[Incorporación de usuarios 11](#_Toc164002259)

[Errores probables 13](#_Toc164002260)

[Servicio predictor 17](#_Toc164002261)

[Servicio guardarBitacora 17](#_Toc164002262)

[Manejo de consultas con Redis 19](#_Toc164002263)

[Errores posibles 20](#_Toc164002264)

[Servicio autorizacion 21](#_Toc164002265)

[Errores posibles 22](#_Toc164002266)

# Requisitos Obligatorios para Ejecutar el Proyecto

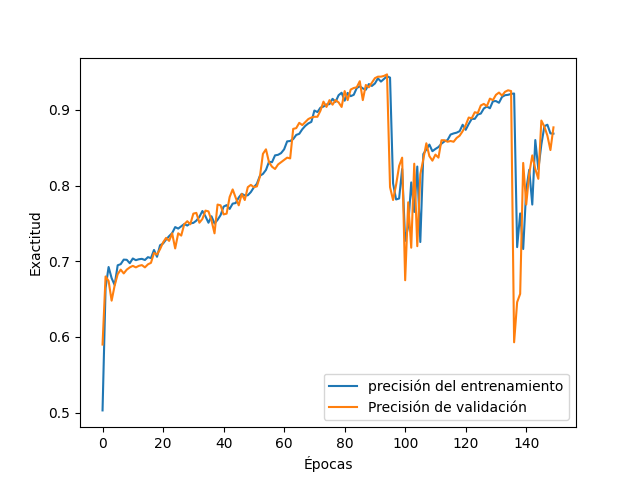
1. Acceso al Repositorio: <https://github.com/polka2005/EntregaParcialML>
2. Python:
   * Tener Python instalado en tu sistema.
   * Trabajo realizado con versión 3.10
3. Visual Studio Code:
   * Necesita tener instalado Visual Studio Code, un entorno de desarrollo de código

abierto.

1. Redis:
   * La aplicación depende de Redis como sistema de almacenamiento en caché.
   * Se adjunta archivos en carpeta Redis
2. Postman:
   * Se recomienda tener Postman para probar y documentar las API.

# Pasos del Proyecto

1. Mediante el archivo *C:\RiesgoCardiaco\RiesoCardiaco\_ML.py,* se realizó el entrenamiento utilizando 3 capas, de 50, 25 y 35, 1 neurona y 150 épocas. Con lo que se obtuvo el modelo extensión keras y su respectivo gráfico de precisión.





Una vez que se obtuvo el modelo para poder realizar las predicciones se procede a:

1. Ejecutar los servicios alojados en C:\RiesgoCardiaco\1\_flask\app1\flaskr

## Servicios ofrecidos:

Los servicios utilizan methods de tipo *POST:*

Desarrollados en app.py:

*altaUsuario*: servicio para la incorporación de usuarios para el uso de la predicción

*predictor*: servicio que predice si se posee o no riesgo cardiaco

Desarrollado en guardarBitacora.py:

*guardarBitacora*: servicio de carga en la bitácora

Desarrollado en autorizacion.py:

*autorizacion*: servicio de autentificación

## Conexión entre servicios:

El servicio *predictor* requiere de los servicios <http://127.0.0.1:5001/autorizacion> a fin de validar usuarios que al devolver el valor *200,* indica que ha validado correctamente al usuario.

Luego de una *correcta predicción*, evaluándose cantidad de consultas según tipo de usuario, cantidad y calidad de los parámetros para dicha medición, este resultado se almacena utilizando el servicio http://127.0.0.1:5005/guardarBitacora

# Decisiones de Diseño

Con respecto al Conjunto de API Keys Válidas:

Actualmente, se utiliza un conjunto predefinido de API keys válidas, por ejemplo, "karina", "karina2"

Es dable mencionar que la llamada en la Base de Datos demora más que lo previsto para acceder a la nube y predecir, que utilizando una base de datos de manera local.

## Uso de Redis:

Redis se utiliza para administrar la cantidad de consultas en el tiempo según el tipo de usuario que accede a la predicción.

# Funciones Implementadas

## Servicio *predictor*

## validartiempo:

Objetivo: La función validarTiempo tiene como objetivo evaluar diferentes condiciones relacionadas con el tiempo y el usuario para determinar si deben permitirse más consultas, enviando respectivos mensajes:

Cuando un usuario FREEMIUM ha superado 5 consultas.

Cuando un usuario PREMIUM ha superado 50 consultas.

### predecir\_conTiempo:

Objetivo: La función predecir\_conTiempo valida las condiciones del usuario según su tipo considerando lo registrado en Redis.

validarParametros:

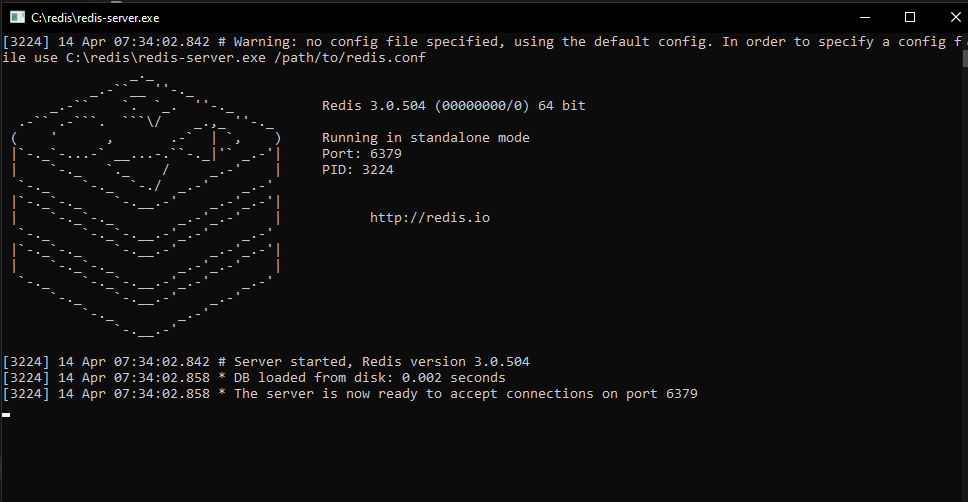
Objetivo: La función que valida los parámetros, considerando la cantidad de parámetros enviados para la predicción como así también que estos se encuentren dentro de un rango correcto pre establecido, dependiendo del parámetro.

# Programas Python del Proyecto

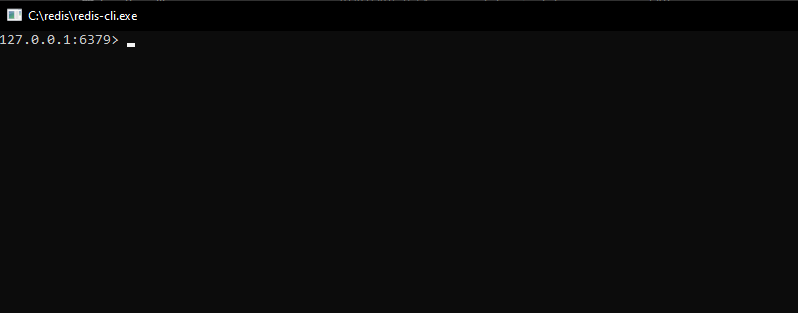
• db.py: Define el acceso a la base de datos.

## Para la ejecución:

1) Se debe ejecutar redis-server

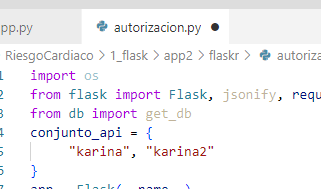


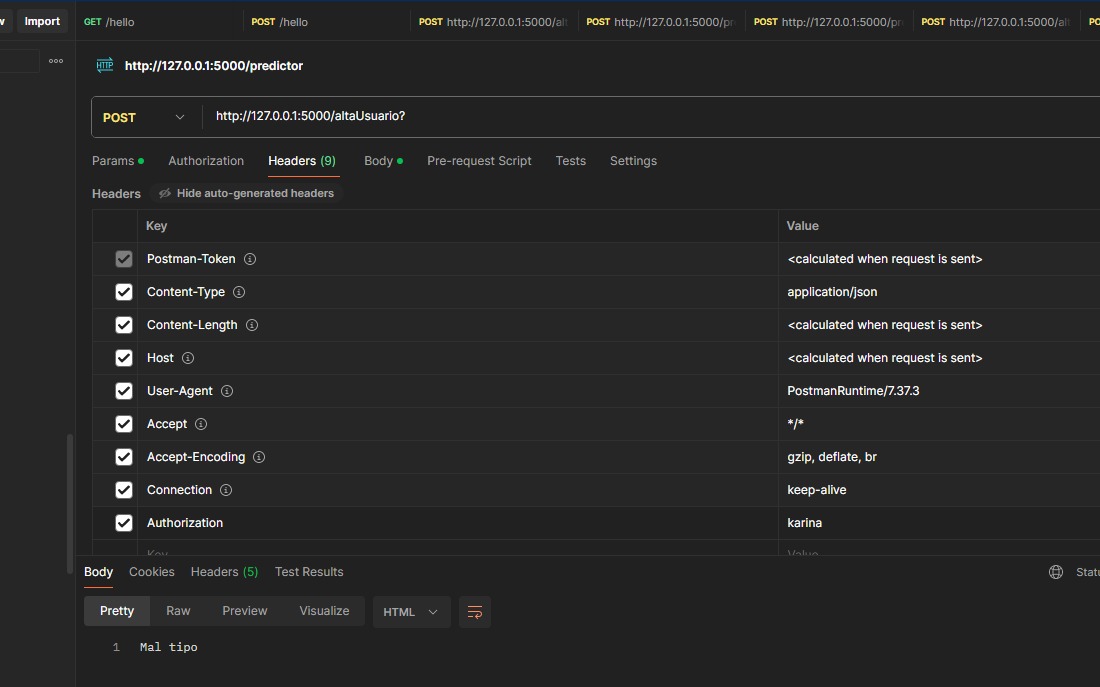
2) Se debe ejecutar redis-cli

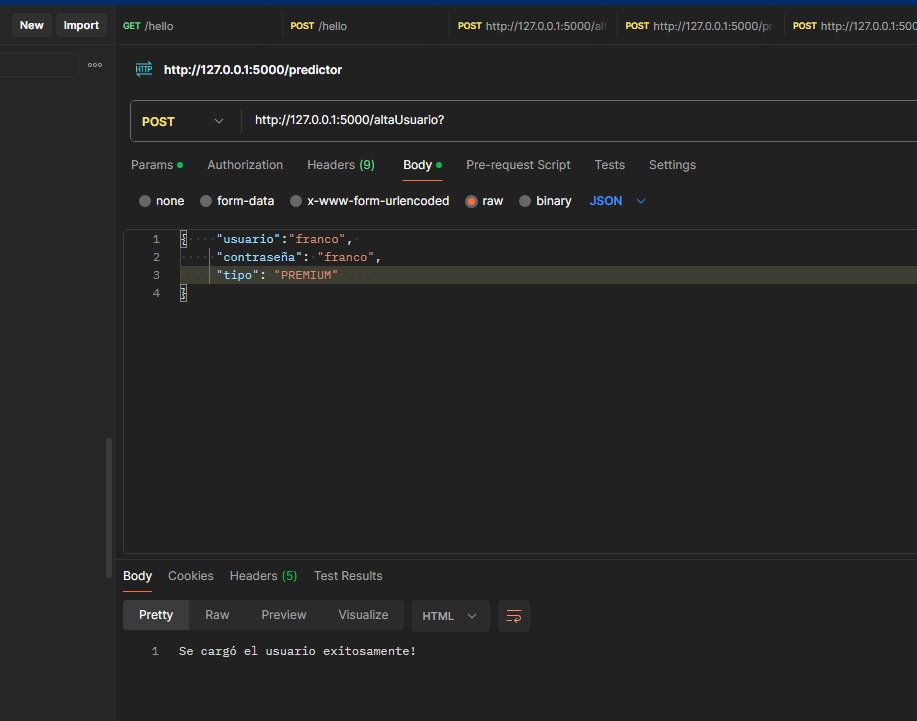


3) En la consola, se pone cada uno de los servicios: “app.py”; “autorizacion.py”; “guardarBitacora.py”.

4) Ejecutar **Postman**, la herramienta que utilizará para realizar solicitudes a la aplicación. En la sección de encabezados (Headers), ingresa en el campo Authorization una de las siguientes Api\_keys válidas: "karina", "karina2".







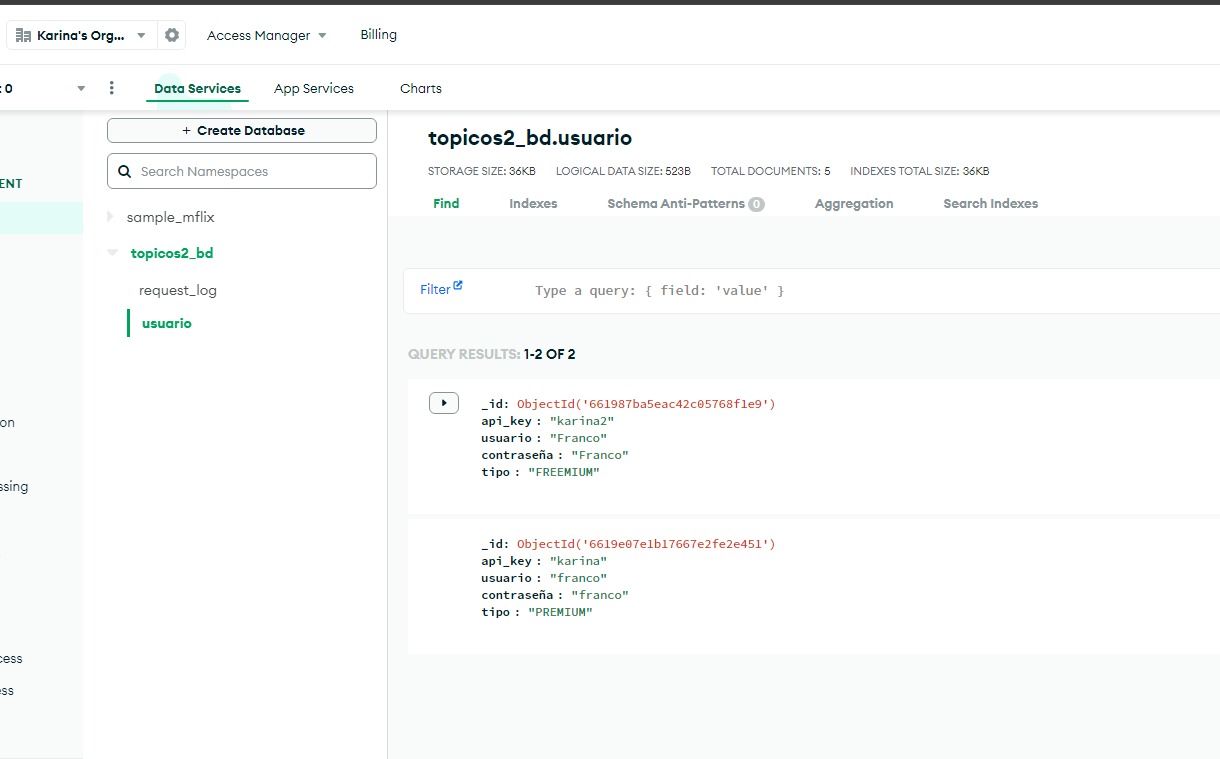
5) Teniendo la siguiente Base de Datos topicos2\_bd:

Ingresando a la BBDD alojada en **Atlas**

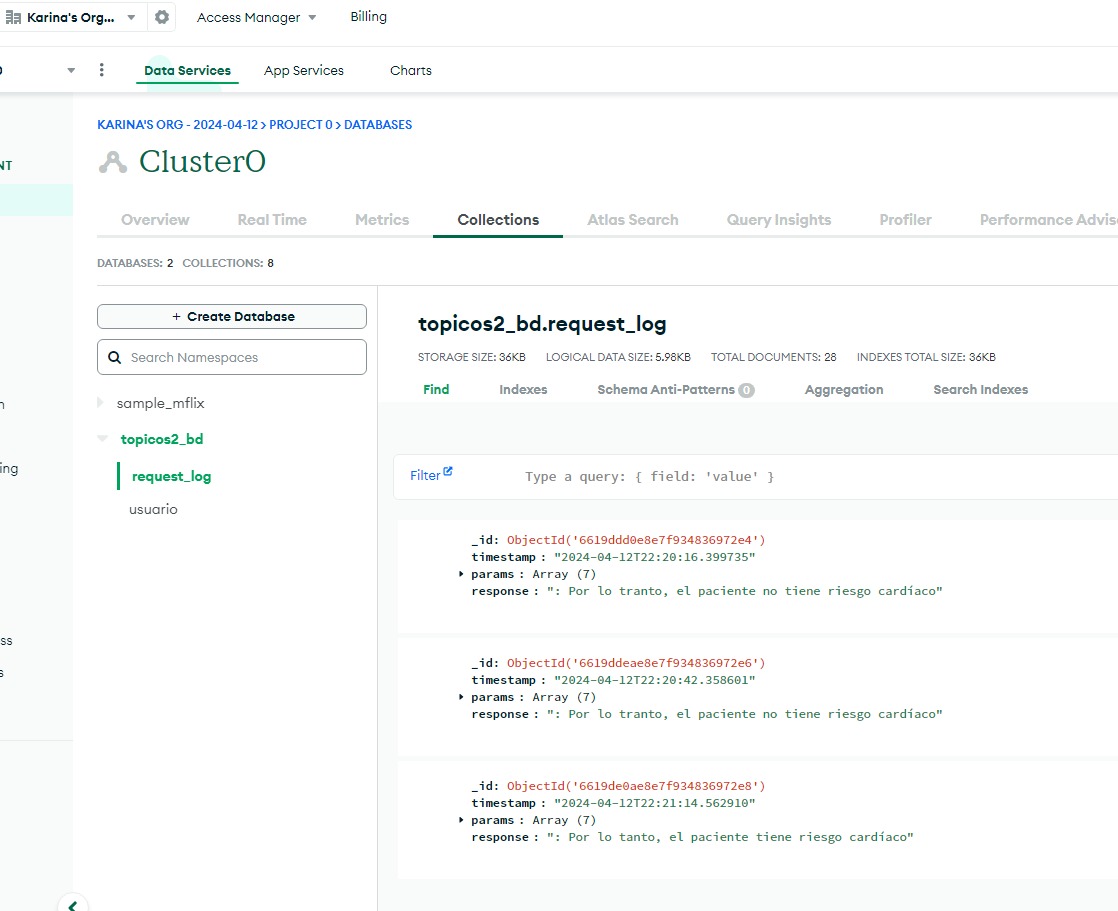
*Usuario*: [kpolicano@hotmail.com](mailto:kpolicano@hotmail.com)

*Contraseña*: PracticoT2

Donde en la estructura *usuario* se guardan los datos de los usuarios, considerando para ser cargados: appy\_key, usuario, contraseña y tipo de usuario (FREEMIUM-PREMIUM)



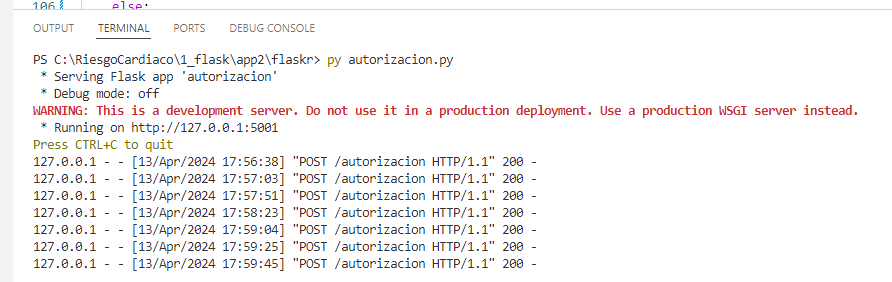
Asimismo, cuenta con una bitácora, *request\_log*, donde se registra cada consulta exitosa, dejando un historial de las mismas.



# Caso de prueba

Activación de los servicios

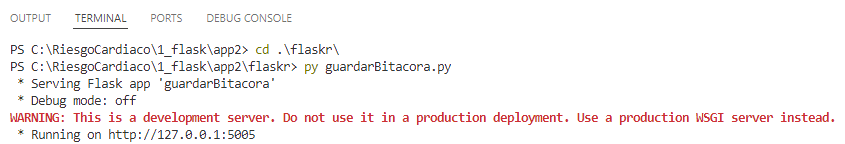
*Autorizacion.py* utilizando el puerto: 5001



*app.py* utilizando el puerto: 5000



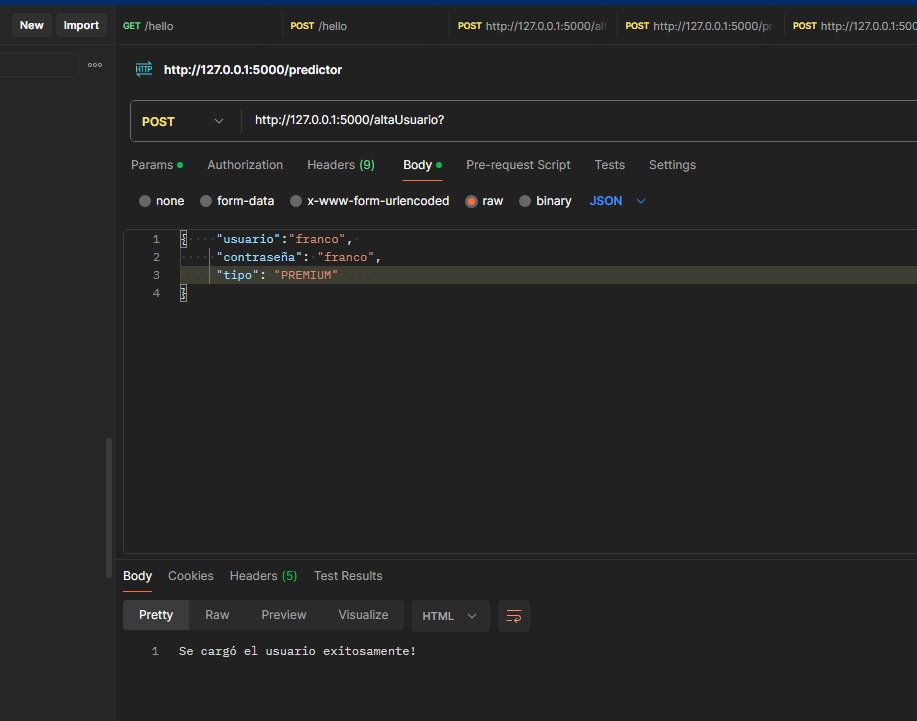
*guardarBitacora.py* utilizando el puerto: 5005



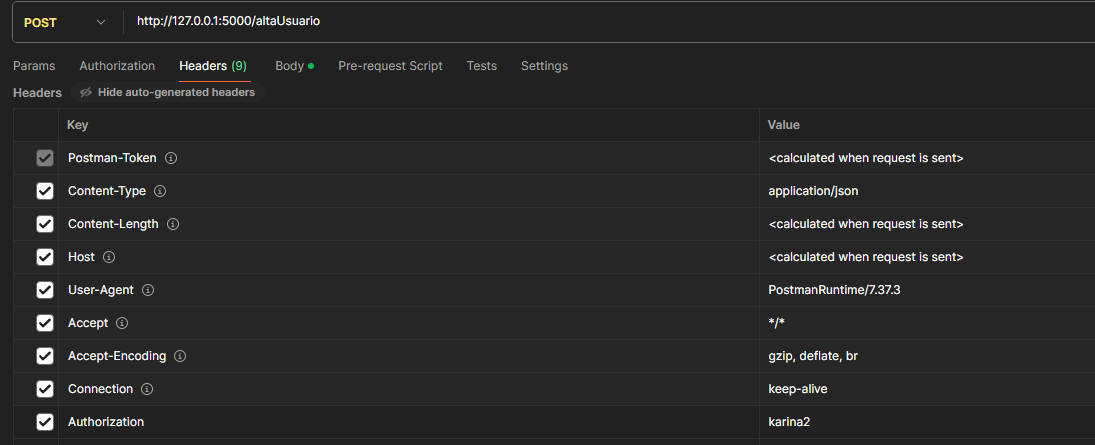
## Servicio altaUsuario

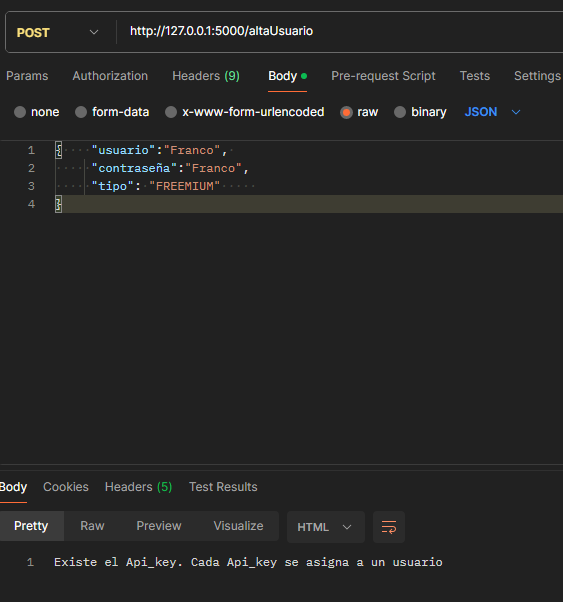
### Incorporación de usuarios

Se procede a insertar los usuarios de ambos tipos para la predicción, utilizándose para el usuario: franco de tipo PREMIUM la appy\_key karina



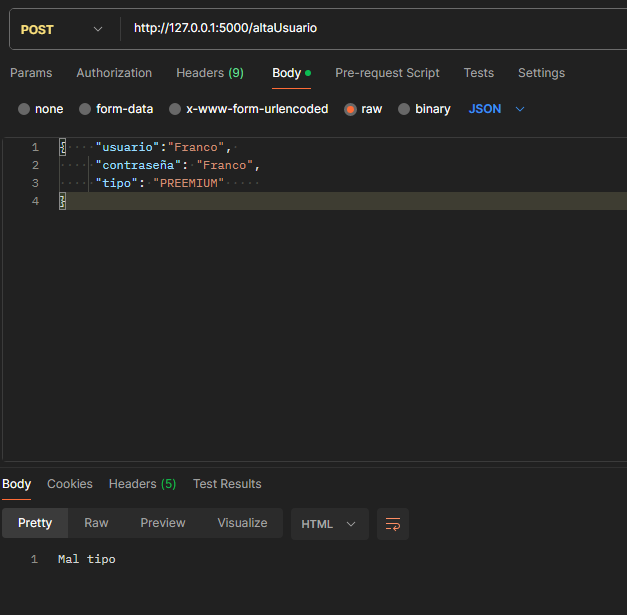
Se procede a intentar insertar un usuario ya registrado, por lo que se obtiene el siguiente mensaje





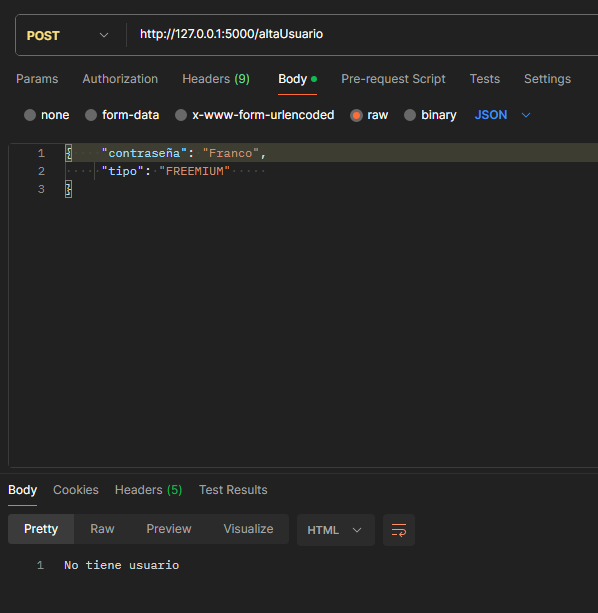
### Errores probables

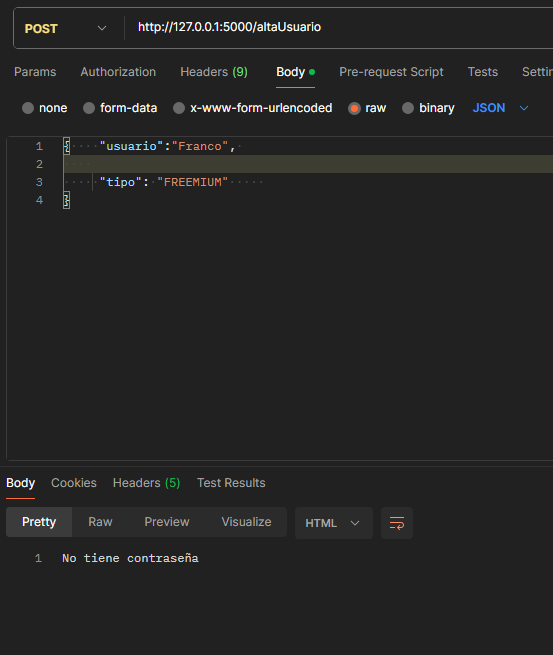
En caso que haya un error en el tipo, en este caso se utilizó el tipo *PREEMIUM* de usuario a ingresar, muestra el mensaje respectivo:



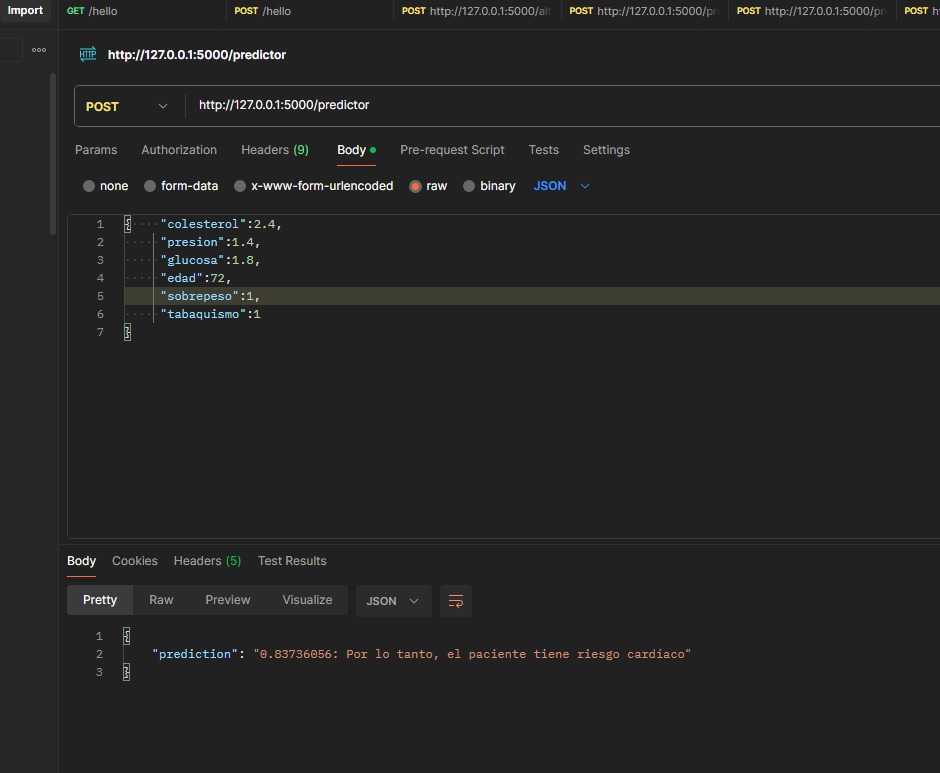


Caso de intentar insertar un usuario sin uno de los parámetros como ser usuario, muestra el mensaje respectivo:





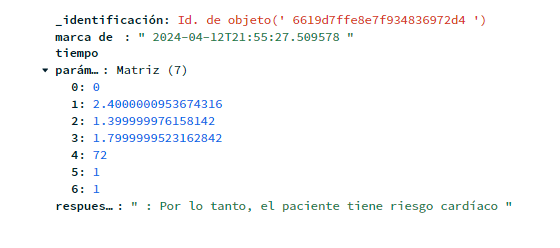
## Servicio predictor

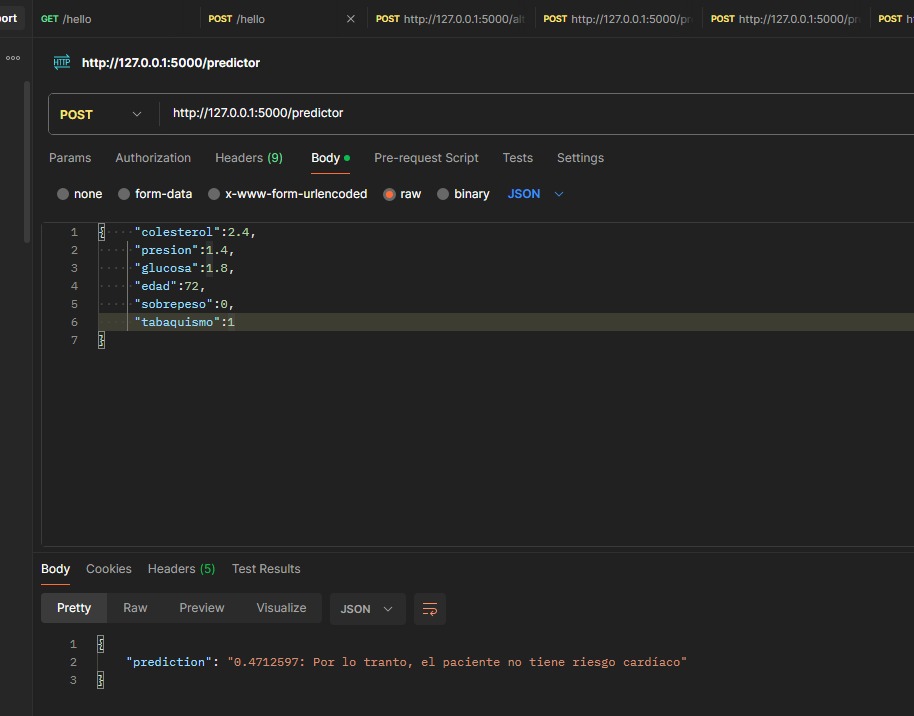


## Servicio guardarBitacora

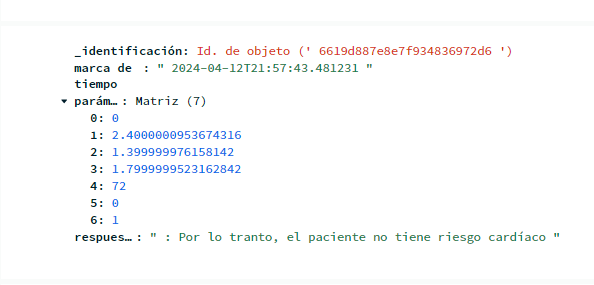
http://127.0.0.1:5005/guardarBitacora

Cuyo guardado en la colección request\_log es:



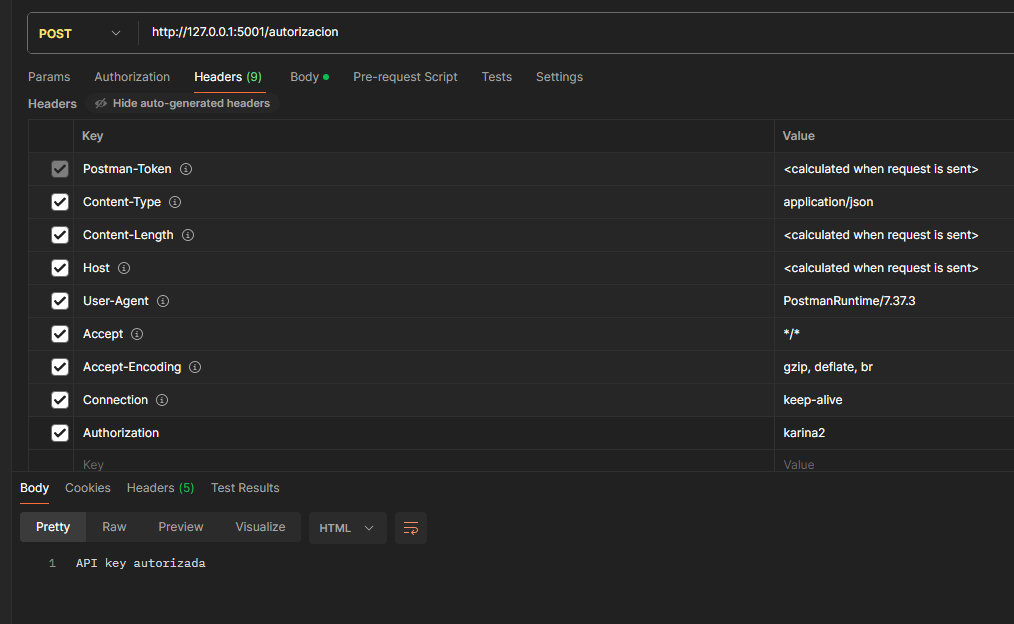


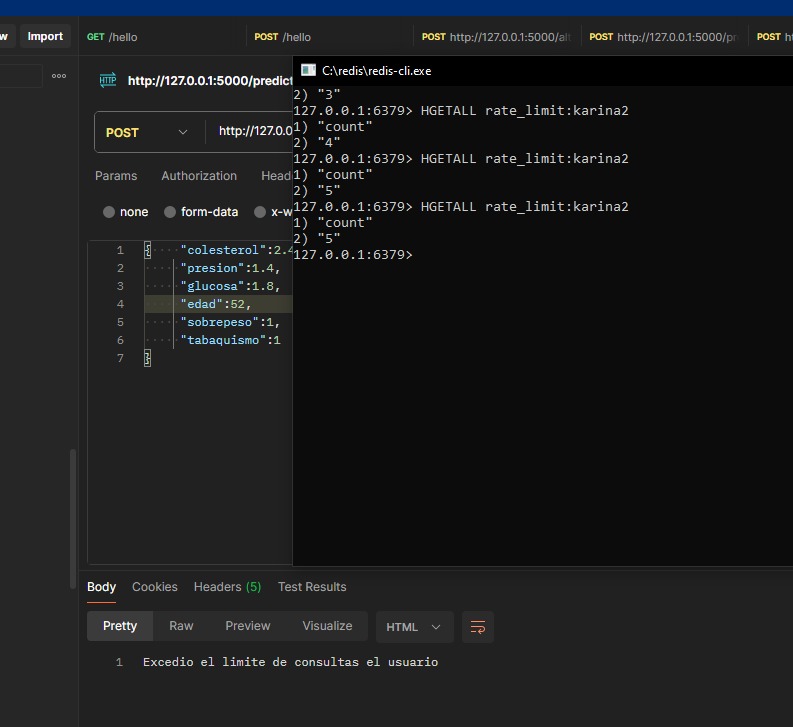
Cuyo guardado en bitácora es:



### Manejo de consultas con Redis

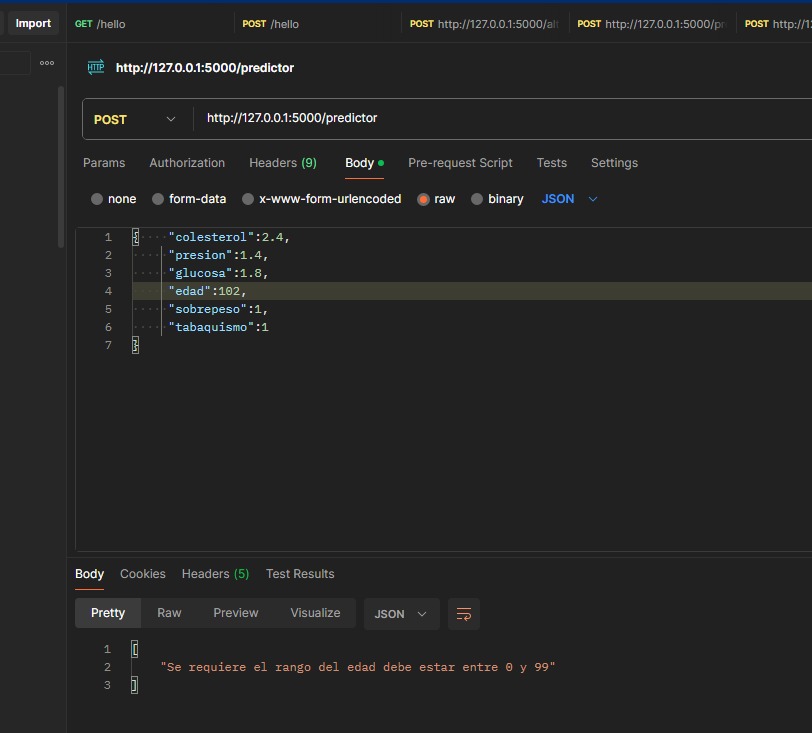
En este caso de usuario tipo **FREEMIUM**, el servicio dejara de ser brindado a las 5 consultas, dato proporcionado por Redis.





### Errores posibles

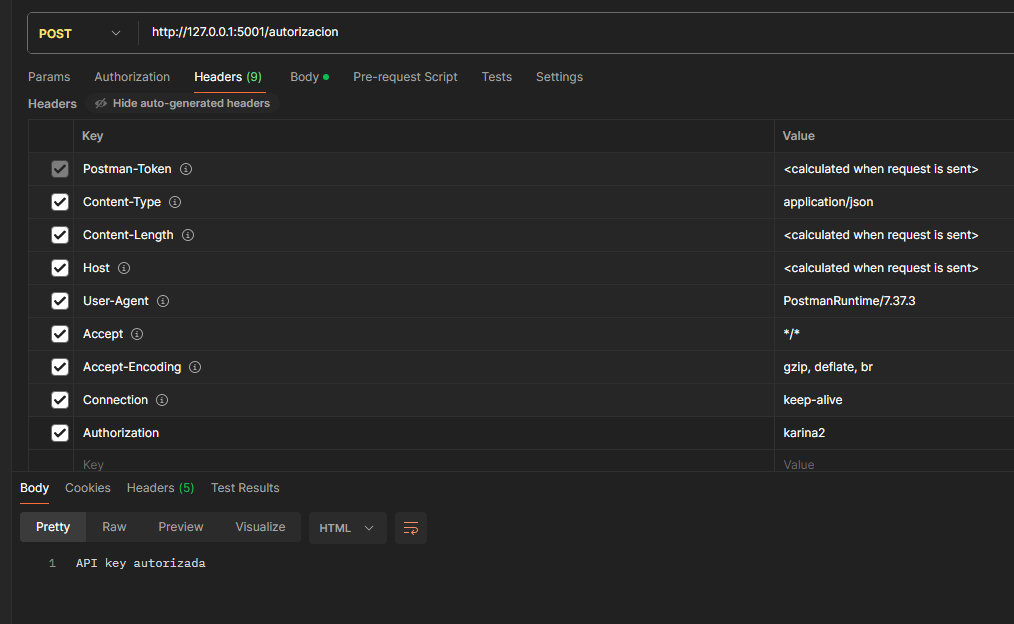
En este caso emite el mensaje de parámetro fuera de rango



## Servicio autorizacion

<http://127.0.0.1:5001/autorizacion>

En este caso se ingresa la apy\_key karina2 registrada



### Errores posibles

En el caso de ingresar una apy\_key no registrada tendremos el siguiente mensaje de error 401

