

**UNIVERSIDAD NACIONAL DE LA PLATA**  
**MAESTRÍA EN INGENIERÍA DE SOFTWARE**



**Trabajo Practico Final**  
**Asignatura: Topicos de Ingenieria de Software**  
**Año: 2023**

**Título: Trabajo Integrador: Servicio Web (API)**

**Autores:**

Gomes de Oliveira Neto, Luiz

Mascelloni, Mariela

**Docentes:**

Prof. Andres Diaz Pace

Prof. Claudia Pons

Prof. Gabriela Perez

Prof. Matias Urbietta

# Requisitos Obligatorios para Ejecutar el Proyecto

1. **Acceso al Repositorio:** <https://github.com/engluzgomes/PracticoTOP2>
2. **Python (Versión 3.11):**
  - Asegúrate de tener Python instalado en tu sistema.
  - La versión requerida es la 3.11.6 Puedes descargarla desde [python.org](https://python.org).
3. **Visual Studio Code:**
  - Necesitarás tener instalado Visual Studio Code, un entorno de desarrollo de código abierto.
  - Puedes descargarlo desde Visual Studio Code.
4. **Redis:**
  - La aplicación depende de Redis como sistema de almacenamiento en caché.
  - Asegúrate de tener Redis instalado. Puedes encontrar información de instalación en [redis.io](https://redis.io).
5. **Postman:**
  - Se recomienda tener Postman para probar y documentar las API.
  - Descarga e instala Postman desde [getpostman.com](https://getpostman.com).

## Pasos para ejecutar exitosamente el Proyecto

1. **Ir a Visual Studio Code**
2. **Ir a la Carpeta del Proyecto de Machine Learning:**

Abre la consola de Visual Studio Code y navega a la carpeta del proyecto de machine learning.  
Comando: `cd practico\0_ml`.

3. **Instalar Dependencias para Machine Learning:**

Ejecuta los siguientes comandos para instalar las dependencias necesarias:

```
pip install tensorflow scikit-learn matplotlib
pip install pandas
pip install scikit-learn
```

4. **Actualizar Pip:**

Para actualizar pip, ejecuta el siguiente comando:

```
python.exe -m pip install --upgrade pip
```

5. **Generar Predicción y Guardar el Modelo:**

En la consola de Visual Studio Code, ejecuta el siguiente comando para generar la predicción y guardar el modelo:

```
cd practico\0_ML
python .\IA-riesgo.py
```

El modelo se almacenará en `'model.keras'`.

### Para ejecutar IA-riesgo.ipynb (Cuaderno de Jupyter):

Si decides ejecutar el cuaderno de Jupyter `'IA-riesgo.ipynb'`, asegúrate de tener instalado `'seaborn'`. Puedes instalarlo con el siguiente comando:

```
pip install seaborn
```

Luego, puedes abrir y ejecutar el cuaderno utilizando tu entorno de Jupyter.

**Nota:** En este escenario, ten en cuenta que el modelo no se guarda, ya que la funcionalidad de guardar el modelo está asociada específicamente al programa `'IA-riesgo.py'`.

## 6. Ir a la Carpeta de la Aplicación Flask:

Navega a la carpeta de la aplicación Flask.

Comando: `cd .\1_flask\app`.

## 7. Crear y Activar el Entorno Virtual:

Crea un entorno virtual con el siguiente comando:

```
python -m venv .venv
```

Activa el entorno virtual:

```
.\venv\Scripts\activate
```

## 8. Instalación de Dependencias para Flask:

Ejecuta los siguientes comandos para instalar las dependencias en el entorno virtual:

```
python.exe -m pip install --upgrade pip
pip install Flask
pip install --upgrade Flask
pip install redis
pip install numpy
pip install tensorflow
pip install pymongo
pip install requests_cache
```

## 9. Ejecutar los servicios:

Para ejecutar simplemente se ingresa en `C:\practico\1_flask\app\flaskr` se pone el servicio, como por ejemplo:

```
.\auth_servicio.py
```

## 10. Servicios ofrecidos:

**app.py: servicio que predice si se posee o no riesgo cardiaco**

**usuarios\_servicio.py: servicio de usuarios**

**logger\_servicio.py: servicio de carga en la bitácora**

**auth\_servicio.py: servicio de autenticación**

# Decisiones de Diseño

## Con respecto al Conjunto de API Keys Válidas:

Actualmente, se utiliza un conjunto predefinido de API keys válidas, por ejemplo, "luiz", "mariela", "mariela2", etc.

## Algunas sugerencias y recomendaciones a tener en cuenta para el manejo de API Keys, en un futuro:

- **Para Mayor Seguridad:**

Se podría generar y almacenar dinámicamente API keys utilizando algún método para evitar vulnerabilidades y aumentar la seguridad.

- **Generación Dinámica de API Keys:**

Se podría implementar un método para generar API keys de manera dinámica y segura.

Estas nuevas keys podrían generarse utilizando algún algoritmo criptográfico o un método seguro para garantizar su aleatoriedad y evitar la adivinación.

- **Almacenamiento en la Base de Datos de las mismas:**

La sugerencia es guardar las API keys generadas dinámicamente en una base de datos segura.

Esto proporcionaría una capa adicional de seguridad y facilitaría la gestión de claves en el tiempo.

Se adjuntó dentro del servicio “**auth\_servicio**” se ofrece “**ingresarApi**”, pero no se usó en la resolución del trabajo final, debido a la llamada en la Base de Datos la cual demoraba más que lo previsto para acceder a la nube y predecir.

## Con respecto a la definición de la tabla Usuarios:

Por una cuestión de simplificación solo se valida contra la api\_key, es decir, en Postman se podría haber seteado el usuario y contraseña, además de la api\_key, y validar los tres elementos (api\_key+usuario+contraseña).

# Funciones Implementadas

- **validarTiempo():**

- **Objetivos:**

La función ‘**validarTiempo**’ tiene como objetivo evaluar diferentes condiciones relacionadas con el tiempo y el usuario para determinar si deben permitirse más consultas.

- **Retornos:**

La función retorna los siguientes códigos:

- 0: Cuando un usuario FREEMIUM ha superado 5 consultas.
- 1: Cuando un usuario PREMIUM ha superado 50 consultas.
- 2: Cuando la diferencia entre la primera y la última consulta es mayor a 1 minuto.
- 8: Cuando no se encuentra el usuario.

- 9: Actualiza la cantidad de consultas, en un retorno exitoso.

- **Uso de Redis:**

Redis se utiliza para evitar el acceso continuo a la base de datos y mejorar el rendimiento.

- **Tabla hash**

Se utilizó una tabla hash para guardar los datos de los usuarios. La estructura de la tabla hash es la siguiente:

```
HGETALL user_rate_limit:mariela
1) "start_time"
2) "2024-03-05 16:03:07.835166"
3) "count"
4) "1"
5) "tipo"
6) "FREEMIUM"
```

En el cual la clave de acceso es `user_key = f"user_rate_limit:{api_key}"`

#### Funciones Redis Utilizadas para el manejo de tabla hash:

- `cx.hmset(user_key, {"start_time": str(start_time), "count": 1, "tipo": tipo})`: se utiliza para insertar en una tabla hash.
- `cx.expire(user_key, 60)`: Establece un TTL (tiempo de vida) de 60 segundos.
- `user_data = cx.hgetall(user_key)`: Recupera los datos de la tabla hash por la clave del usuario, que en este caso es el `api_key`.
- `cx.hset(user_key, "count", count)`: Actualiza el contador de consultas.
- `validarParametros()`:

Función que valida los parámetros, tanto en cantidad, como que estén dentro de un rango correcto, dependiendo del parámetro.

## Funciones Apps Implementadas en los 4 servicios

- `'auth()'`: Controla la autorización de la API basándose en un conjunto de `'api_key'` posibles.
  - Está en el servicio `.\auth_servicio.py`
- `'predict()'`: Realiza la autenticación, obtiene la API Key, valida el tiempo, predice, guarda en la bitácora y formatea el resultado y `'predictor()'`: Predice el modelo.
  - Está en el servicio `.\app.py`
- `'ingresar()'`: Ingresa usuarios en la base de datos, no se implementó la modificación de los usuarios, ni la baja de ellos.
  - Está en el servicio `.\usuarios_servicio.py`
- `'logger()'`: Graba en la bitácora (un log).
  - Está en el servicio `.\logger_servicio.py`

## **Detalle de la función app principal, predict():**

### **1. Autenticación de API Key:**

Autentica la API key proporcionada en la solicitud.

### **2. Obtención de API Key de Postman:**

Obtiene la API key de la solicitud de Postman, que puede estar utilizando para realizar solicitudes.

### **3. Validación de Tiempo y Tipo de Usuario:**

Valida el tiempo de la solicitud en relación con el tipo de usuario.

Puede realizar comprobaciones basadas en el tipo de usuario, como limitar el número de consultas.

### **4. Predicción (predictor):**

Utiliza un predictor para realizar una predicción basada en los parámetros proporcionados en la solicitud.

Se asegura de que todos los parámetros necesarios estén presentes y dentro del rango correcto.

### **5. Registro en Bitácora:**

Guarda la información relevante en la bitácora.

Puede incluir detalles como la API key, parámetros de la solicitud, resultados de la predicción, etc.

### **6. Formateo del Resultado:**

Formatea el resultado de la predicción para devolverlo de manera clara y legible en la respuesta.

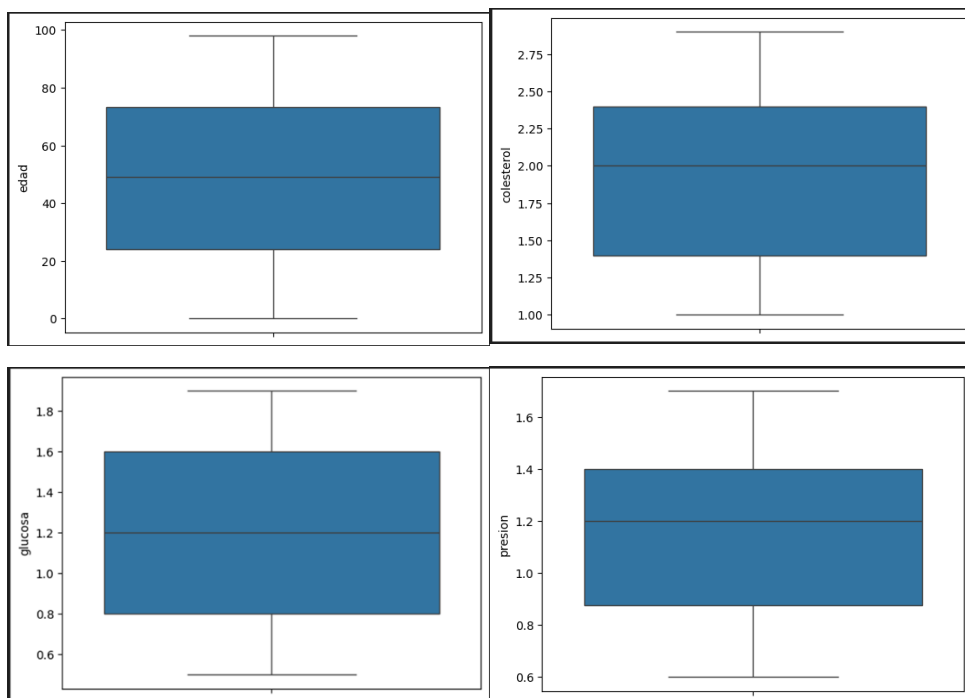
**Aclaración:** Se asegura de que todos los parámetros necesarios estén presentes y dentro del rango correcto al realizar la predicción.

## Ejemplos de cómo se convoca el servicio auth\_servicio

```
C:\Windows\py.exe
* Serving Flask app 'auth_servicio'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5002
Press CTRL+C to quit
```

## Análisis de Datos con IA

Tenemos que los parámetros oscilan según las siguientes graficas:



### División del Conjunto de Datos:

Después de cargar el conjunto de datos, se dividió en conjuntos de entrenamiento y prueba. Esta práctica es fundamental para evaluar el rendimiento del modelo en datos no vistos. Hemos dividido el conjunto de datos en conjuntos de entrenamiento y prueba utilizando la función `train_test_split` de `scikit-learn`. Esta división es esencial para evaluar el rendimiento del modelo en datos no observados.

- Separar los datos de entrada X y los datos de salida Y

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

### Escalado de Variables Numéricas:

Para garantizar una convergencia eficiente y la estabilidad del modelo, escalamos las variables numéricas utilizando la clase **StandardScaler** de **scikit-learn**. Este paso es crucial para algoritmos sensibles a la escala, como la regresión logística. Escalamos los conjuntos de entrenamiento y prueba por separado para evitar fugas de información entre ellos.

- Escalar variables numéricas para el conjunto de entrenamiento

```
scaled_X_train = scaler.fit_transform(X_train)
scaled_X_train = pd.DataFrame(scaled_X_train, columns=X_train.columns)
```

- Escalar variables numéricas para el conjunto de prueba

```
scaled_X_test = scaler.fit_transform(X_test)
scaled_X_test = pd.DataFrame(scaled_X_test, columns=X_test.columns)
```

### Conjunto de Entrenamiento Escalado:

Después de escalar las variables numéricas para el conjunto de entrenamiento, convertimos los datos escalados de nuevo a un **DataFrame** de Pandas. Esto nos permite visualizar las primeras filas del conjunto de entrenamiento escalado, proporcionando una comprensión inicial de la transformación aplicada.

- Mostrar las primeras filas de los conjuntos escalados para el conjunto de entrenamiento

```
print("Conjunto de Entrenamiento Escalado:")
print(scaled_X_train.head())
```

### Conjunto de Prueba Escalado:

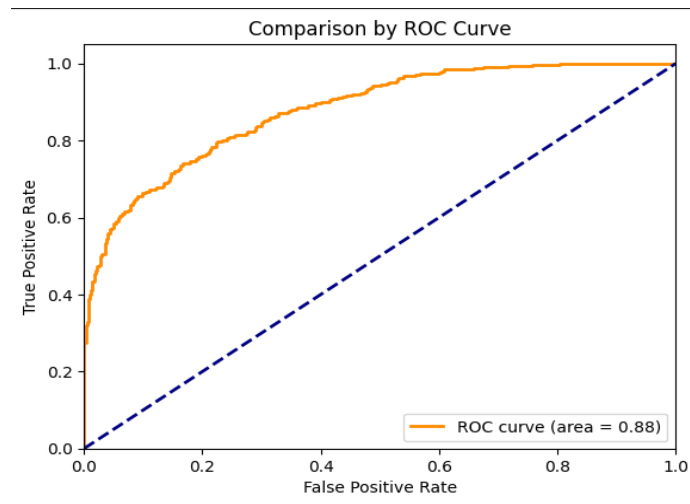
De manera similar al conjunto de entrenamiento, escalamos las variables numéricas en el conjunto de prueba utilizando el mismo escalador ajustado al conjunto de entrenamiento. Los datos escalados también se convierten en un **DataFrame** de Pandas para facilitar la inspección de las primeras filas y garantizar consistencia en los procedimientos aplicados.

- Mostrar las primeras filas de los conjuntos escalados para el conjunto de prueba

```
print("\nConjunto de Prueba Escalado:")
print(scaled_X_test.head())
```

**Se puede observar que la regresión lógica del modelo utilizado es:**





A medida que entrenamos el modelo, se obtienen diversas precisiones durante la evaluación. Por ejemplo, al realizar la evaluación del modelo con la siguiente instrucción:

```
score = model.evaluate(X_test, y_test, verbose=0)
print('Precisión:', score[1])
```

Podemos observar que las precisiones varían, obteniendo valores como **0.7639999985694885**, **0.7789999842643738**, ..., **0.875**. Estos valores representan la precisión del modelo en el conjunto de prueba en diferentes momentos durante el entrenamiento. La variación en estos números puede ser indicativa de cómo el modelo está aprendiendo y generalizando a lo largo del proceso de entrenamiento. Un aumento en la precisión generalmente sugiere una mejora en el rendimiento del modelo.

En nuestro modelo grabado, que tiene una precisión del **0.921999990940094**, se ha determinado que una persona de 82 años con sobrepeso tiene un riesgo cardiaco del **98,56%**. Este resultado sugiere un alto riesgo cardiovascular para una persona con esas características según las predicciones de nuestro modelo. La precisión del **92.2%** indica la confianza del modelo en sus predicciones, aunque siempre es importante considerar otras variables y fuentes de información al interpretar estos resultados.

<http://127.0.0.1:5000/predict?colesterol=1.4&presion=1.5&glucosa=1.8&edad=82&sobrepeso=1&tabaquismo=0...>
Save

POST
http://127.0.0.1:5000/predict?colesterol=1.4&presion=1.5&glucosa=1.8&edad=82&sobrepeso=1&tabaquismo=0...
Send

Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies

Headers
8 hidden

	Key	Value	Description	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Authorization	mariela2				
	Key	Value	Description			

body
Cookies
Headers (5)
Test Results
200 OK
1137 ms
202 B
Save as example

Pretty
Raw
Preview
Visualize
HTML

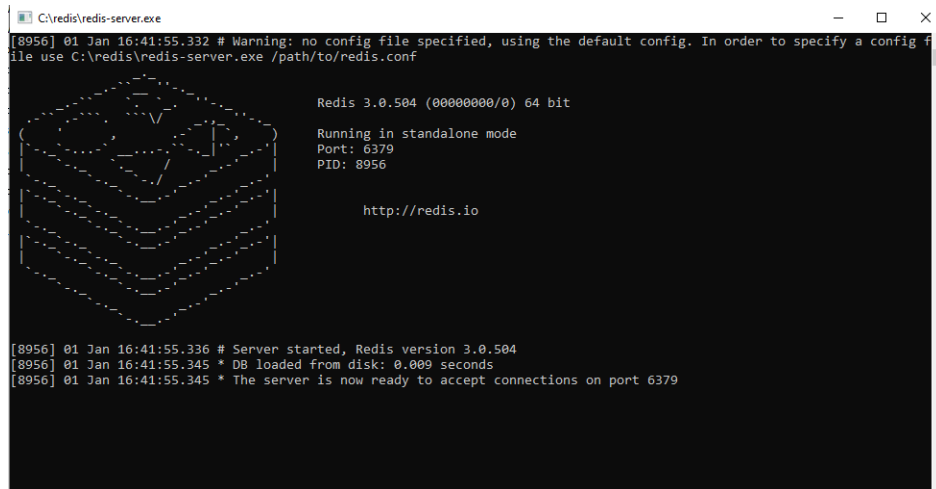
1 Tiene un riesgo de 98.56147 %

## Programas Python del Proyecto

- **'IA-riesgo.py'**: Realiza la inteligencia para calcular el riesgo cardíaco y guarda el modelo.
- **'\_init\_.py'**: Define las funciones más importantes.
- **'db.py'**: Define el acceso a la base de datos.

### Para la ejecución:

- 1) Se debe ejecutar **redis-server**



```
C:\redis\redis-server.exe
[8956] 01 Jan 16:41:55.332 # Warning: no config file specified, using the default config. In order to specify a config file use C:\redis\redis-server.exe /path/to/redis.conf

Redis 3.0.504 (00000000/0) 64 bit

Running in standalone mode
Port: 6379
PID: 8956

http://redis.io

[8956] 01 Jan 16:41:55.336 # Server started, Redis version 3.0.504
[8956] 01 Jan 16:41:55.345 * DB loaded from disk: 0.009 seconds
[8956] 01 Jan 16:41:55.345 * The server is now ready to accept connections on port 6379
```

- 2) Se debe ejecutar **redis-cli**



```
C:\redis\redis-cli.exe
127.0.0.1:6379>
```

- 3) En la consola, se pone cada uno de los servicios: **"app.py"**; **"auth\_servicio.py"**; **"usuarios\_servicio.py"**; **"logger\_servicio.py"**.
- 4) Abre Postman, la herramienta que utilizarás para realizar solicitudes a tu aplicación. En la sección de encabezados (Headers), ingresa en el campo Authorization una de las siguientes claves: **"mariela"**, **"mariela2"**, **"mariela3"**, **"luiz"**, **"luizrr"** o **"ana"**.

HTTP <http://127.0.0.1:5000/predict?> Save

POST <http://127.0.0.1:5000/predict?> Send

Params Authorization **Headers (9)** Body Pre-request Script Tests Settings Cookies

Headers 8 hidden

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	Authorization	luiz	
	Key	Value	

5) Teniendo la siguiente Base de Datos Práctico:

▼ **Práctico**

| **request\_log**

usuario

Con usuario y contraseña, como sigue:

**Usuario:** mmascelloni@yahoo.com.ar

**Contraseña:** Mariela2023

Con los siguientes usuarios cargados:

```
_identificación: Id. de objeto(' 65899e3aed7e9722f2ebb501 ')  
Clave : " mariela "  
API  
usuario: " mariela "  
contraseña: " mariela "  
tipo: " DE PRIMERA CALIDAD "
```

```
_identificación: Id. de objeto (' 6589a01fed7e9722f2ebb511 ')  
Clave : " mariela2 "  
API  
usuario: " mariela "  
contraseña: " mariela "  
tipo: " GRATIS "
```

```
_identificación: Id. de objeto(' 658b3fbdbe4f95854eb5f59c ')  
Clave : " mariela24 "  
API  
usuario: " mariela3 "  
contraseña: " mariela "  
tipo: " DE PRIMERA CALIDAD "
```

```
_identificación: Id. de objeto(' 658b413bbe4f95854eb5f5a2 ')  
Clave : " luiz "  
API  
usuario: " luiz "  
contraseña: " luiz "  
tipo: " GRATIS "
```

- Y una bitácora (log), donde se registra cada consulta exitosa, dejando un historial de las mismas.

## Práctico.request\_log

STORAGE SIZE: 44KB   LOGICAL DATA SIZE: 22.76KB   TOTAL DOCUMENTS: 127   INDEXES TOTAL SIZE: 36KB

Find

Indexes

Schema Anti-Patterns ⓘ

Aggregation

Search Indexes

Filter 

Type a query: { field: 'value' }

### QUERY RESULTS: 1-20 OF MANY

```
_id: ObjectId('65899ed5ed7e9722f2ebb504')
timestamp: "2023-12-25T12:25:09.033864"
▶ params: Array (7)
response: "[[0.64441836]]"
```

```
_id: ObjectId('65899f57ed7e9722f2ebb506')
timestamp: "2023-12-25T12:27:19.089398"
▶ params: Array (7)
response: "[[0.64441836]]"
```

```
_id: ObjectId('65899f60ed7e9722f2ebb508')
timestamp: "2023-12-25T12:27:28.563717"
▶ params: Array (7)
response: "[[0.64441836]]"
```

### 6) Con éstas Api\_keys válidas

```
api_keys = {
    "mariela", "mariela2", "mariela3", "luiz", " luizrr", "ana"
}
app = Flask(__name__)
```

## Así como todos los servicios brindados

```
C:\Windows\py.exe
2024-03-09 09:28:01.694308: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.
WARNING:tensorflow:From C:\Users\Mariela Mascelloni\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

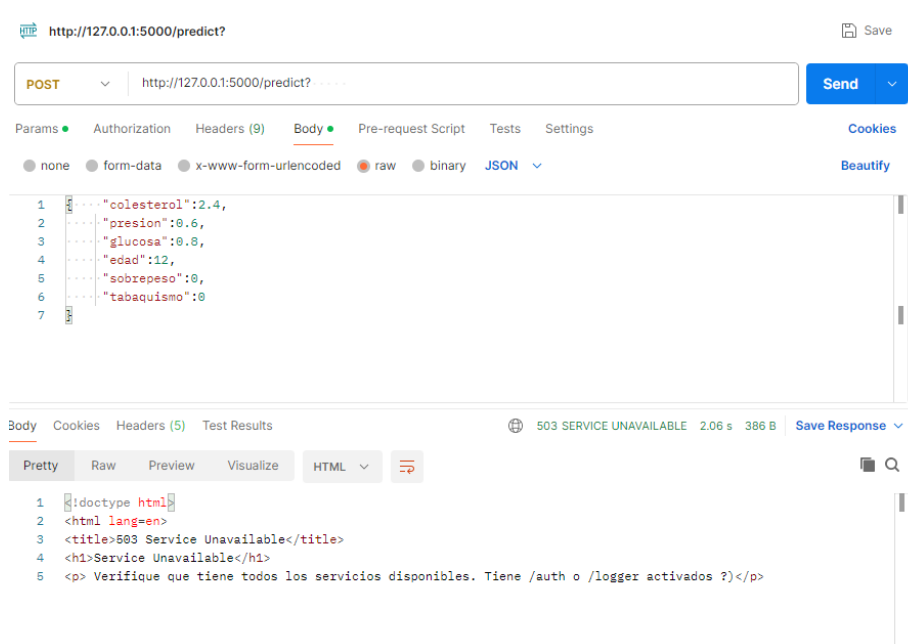
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

```
C:\Windows\py.exe
* Serving Flask app 'usuarios_servicio'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5001
Press CTRL+C to quit
```

```
* Serving Flask app 'auth_servicio'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5002
Press CTRL+C to quit
```

```
* Serving Flask app 'logger_servicio'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5003
Press CTRL+C to quit
```

En la función app del servicio principal **app.py**, si los demás servicios no están activos se controla, como sigue:



# Evidencia de prueba

- Activación del servicio **app.py**

```
C:\Windows\py.exe
2024-03-09 09:28:01.694308: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.
WARNING:tensorflow:From C:\Users\Mariela Mascelloni\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

- Api\_key disponibles

```
api_keys = {
    "mariela", "mariela2", "mariela3", "luiz", " luizrr", "ana"
}
app = Flask(__name__)
```

## Caso de prueba: POST: <http://127.0.0.1:5000/ingresarUsuarios?>

REST client interface showing a POST request to <http://127.0.0.1:5000/ingresarUsuarios?>. The request is configured with the following headers and body:

Key	Value	Bulk Edit
<input checked="" type="checkbox"/> Authorization	mariela	
Key	Value	

The body is set to JSON format with the following content:

```
1 {
2   "usuario": "mariela2",
3   "contraseña": "mariela",
4   "tipo": "FREEMIUM"
}
```

- El servicio que llama no está activado

HTTP <http://127.0.0.1:5000/ingresarUsuarios?> Save

**POST** <http://127.0.0.1:5000/ingresarUsuarios?> Send

Params • Authorization Headers (9) **Body** • Pre-request Script Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary **JSON** Beautify

```

1  {
2    "usuario": "mariela2",
3    "contraseña": "mariela",
4    "tipo": "FREEMIUM"
  }

```

Body Cookies Headers (5) Test Results 503 SERVICE UNAVAILABLE 2.06 s 349 B Save Response

**Pretty** Raw Preview Visualize **HTML** 🔍

```

1  <!doctype html>
2  <html lang=en>
3  <title>503 Service Unavailable</title>
4  <h1>Service Unavailable</h1>
5  <p> Verifique que tiene el servicio usuarios activado.</p>

```

```

2024-03-09 09:28:01.694308: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
WARNING:tensorflow:From C:\Users\Mariela Mascelloni\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
pase por ingresarUsuarios
127.0.0.1 - - [09/Mar/2024 09:35:29] "POST /ingresarUsuarios?%20%20 HTTP/1.1" 503 -

```

- Si se activa el servicio

```

C:\Windows\py.exe
* Serving Flask app 'usuarios_servicio'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5001
Press CTRL+C to quit

```



## Entonces

HTTP <http://127.0.0.1:5000/ingresarUsuarios?> Save

POST <http://127.0.0.1:5000/ingresarUsuarios?> Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary JSON Beautify

```
1 {
2   "usuario": "mariela2",
3   "contraseña": "mariela",
4   "tipo": "FREEMIUM"
}
```

Body Cookies Headers (5) Test Results 200 OK 5.41 s 207 B Save Response

Pretty Raw Preview Visualize HTML

```
1 Se cargó el usuario exitosamente!
```

- En la base de datos de usuarios:

### RESULTADOS DE LA CONSULTA: 1-1 DE 1

```
_identificación: Id. de objeto(' 65ec5800e3393059774248dd ')
Clave : " mariela "
API
usuario: " mariela2 "
contraseña: " mariela "
tipo: " GRATIS "
```

- Se repite el ingreso:

POST <http://127.0.0.1:5000/ingresarUsuarios?> Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary JSON Beautify

```
1 {
2   "usuario": "mariela2",
3   "contraseña": "mariela",
4   "tipo": "FREEMIUM"
}
```

Body Cookies Headers (5) Test Results 200 OK 6.00 s 227 B Save Response

Pretty Raw Preview Visualize HTML

```
1 Existe el Api_key. Cada Api_key se asigna a un usuario
```

## Caso de prueba: POST: http://127.0.0.1:5000/predictor?

HTTP <http://127.0.0.1:5000/predictor?> Save

POST <http://127.0.0.1:5000/predictor?> Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary JSON Beautify

```
1 {
2   "colesterol":2.4,
3   "presion":0.6,
4   "glucosa":0.8,
5   "edad":82,
6   "sobrepeso":1,
7   "tabaquismo":1
8 }
```

Body Cookies Headers (5) Test Results 200 OK 1414 ms 187 B Save Response

Pretty Raw Preview Visualize HTML Search

```
1 [[0.79536766]]
```

- Si un parámetro es incorrecto:

POST <http://127.0.0.1:5000/predictor?> Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary JSON Beautify

```
1 {
2   "colesterol":12.4,
3   "presion":0.6,
4   "glucosa":0.8,
5   "edad":82,
6   "sobrepeso":1,
7   "tabaquismo":1
8 }
```

Body Cookies Headers (5) Test Results 400 BAD REQUEST 7 ms 326 B Save Response

Pretty Raw Preview Visualize HTML Search

```
1 <!doctype html>
2 <html lang=en>
3 <title>400 Bad Request</title>
4 <h1>Bad Request</h1>
5 <p>El rango del colesterol debe estar entre (1.0, 3.0) </p>
```

POST http://127.0.0.1:5000/predictor?..... Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary JSON

```

1  .... "presion":0.6,
2  .... "glucosa":0.8,
3  .... "edad":82,
4  .... "sobrepeso":1,
5  .... "tabaquismo":1
6

```

Body Cookies Headers (5) Test Results 400 BAD REQUEST 6 ms 296 B Save Response

Pretty Raw Preview Visualize HTML

```

1  <!doctype html>
2  <html lang=en>
3  <title>400 Bad Request</title>
4  <h1>Bad Request</h1>
5  <p>Se requiere colesterol</p>

```

### Caso de prueba: POST: http://127.0.0.1:5000/predict?

HTTP http://127.0.0.1:5000/predict? Save

POST http://127.0.0.1:5000/predict?..... Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Headers 8 hidden

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	Authorization	luiz	
	Key	Value	

Body Cookies Headers (5) Test Results 503 SERVICE UNAVAILABLE 2.05 s 386 B Save Response

Pretty Raw Preview Visualize HTML

```

1  <!doctype html>
2  <html lang=en>
3  <title>503 Service Unavailable</title>
4  <h1>Service Unavailable</h1>
5  <p> Verifique que tiene todos los servicios disponibles. Tiene /auth o /logger activados ?</p>

```

- Si activo los dos servicios que llaman que todavía no están activados

```

* Serving Flask app 'auth_servicio'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5002
Press CTRL+C to quit

```

```

* Serving Flask app 'logger_servicio'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5003
Press CTRL+C to quit

```

- Con los parámetros:

POST

http://127.0.0.1:5000/predict? .....

Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Headers 8 hidden

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	Authorization	mariaela	
	Key	Value	

http://127.0.0.1:5000/predict?

Save

POST

http://127.0.0.1:5000/predict? .....

Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary JSON

```
1 ..... "colesterol":2.4,
2 ..... "presion":0.6,
3 ..... "glucosa":0.8,
4 ..... "edad":82,
5 ..... "sobrepeso":1,
6 ..... "tabaquismo":1
7
```

http://127.0.0.1:5000/predict?

Save

POST

http://127.0.0.1:5000/predict? .....

Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary JSON

```
1 ..... "colesterol":2.4,
2 ..... "presion":0.6,
3 ..... "glucosa":0.8,
4 ..... "edad":82,
5 ..... "sobrepeso":1,
6 ..... "tabaquismo":1
7
```

Body Cookies Headers (5) Test Results 200 OK 10.59 s 203 B Save Response

Pretty Raw Preview Visualize HTML

1 Tiene un riesgo de 79.536766 %

- Si solicitamos más de 5 consultas para un usuario **FREEMIUM**:

REST client interface showing a POST request to `http://127.0.0.1:5000/predict?`. The request body is a JSON object:

```
1 {
2   "colesterol": 2.4,
3   "presion": 0.6,
4   "glucosa": 0.8,
5   "edad": 82,
6   "sobrepeso": 1,
7   "tabaquismo": 1
8 }
```

The response is a 409 CONFLICT status with a 5.63 s response time and 309 B body size. The response body is HTML:

```
1 <!doctype html>
2 <html lang=en>
3 <title>409 Conflict</title>
4 <h1>Conflict</h1>
5 <p>Superó las 5 consultas, el usuario FREEMIUM</p>
```

- Si cambiamos los parámetros para observar la variabilidad del riesgo:

REST client interface showing a POST request to `http://127.0.0.1:5000/predict?`. The request body is a JSON object:

```
1 {
2   "colesterol": 2.4,
3   "presion": 0.7,
4   "glucosa": 0.8,
5   "edad": 2,
6   "sobrepeso": 0,
7   "tabaquismo": 0
8 }
```

The response is a 200 OK status with a 5.48 s response time and 202 B body size. The response body is text:

```
1 Tiene un riesgo de 0.011644 %
```

- Si la api\_key no está en el conjunto válido:

HTTP <http://127.0.0.1:5000/predict?> Save

**POST** <http://127.0.0.1:5000/predict?> Send

Params • Authorization Headers (9) Body • Pre-request Script Tests Settings Cookies

Headers 8 hidden

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	Authorization	mariela8999	
	Key	Value	

Body Cookies Headers (5) Test Results 401 UNAUTHORIZED 11 ms 294 B Save Response

Pretty Raw Preview Visualize HTML Search

```
1 <!doctype html>
2 <html lang=en>
3 <title>401 Unauthorized</title>
4 <h1>Unauthorized</h1>
5 <p>Api no autorizada</p>
```

- Si los parámetros no están o no son los adecuados:

HTTP <http://127.0.0.1:5000/predict?> Save

**POST** <http://127.0.0.1:5000/predict?> Send

Params • Authorization Headers (9) Body • Pre-request Script Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary **JSON** Beautify

```
1 {
2   "presion":0.7,
3   "glucosa":0.8,
4   "edad":2,
5   "sobrepeso":0,
6   "tabaquismo":0
7 }
```

Body Cookies Headers (5) Test Results 400 BAD REQUEST 5.00 s 296 B Save Response

Pretty Raw Preview Visualize HTML Search

```
1 <!doctype html>
2 <html lang=en>
3 <title>400 Bad Request</title>
4 <h1>Bad Request</h1>
5 <p>Se requiere colesterol</p>
```

HTTP <http://127.0.0.1:5000/predict?> Save

**POST** <http://127.0.0.1:5000/predict?> Send

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded **raw** binary JSON Beautify

```
1 {
2   "colesterol":2.4,
3   "presion":0.7,
4   "glucosa":8.8,
5   "edad":2,
6   "sobrepeso":0,
7   "tabaquismo":0
8 }
```

Body Cookies Headers (5) Test Results 400 BAD REQUEST 8 ms 323 B Save Response

Pretty Raw Preview Visualize HTML ⋮ 🔍

```
1 <!doctype html>
2 <html lang=en>
3 <title>400 Bad Request</title>
4 <h1>Bad Request</h1>
5 <p>El rango del glucosa debe estar entre (0.6, 2.0) </p>
```

### Caso de prueba: POST: <http://127.0.0.1:5001/ingresar?>

**POST** <http://127.0.0.1:5001/ingresar?> Send

Params Authorization **Headers (9)** Body Pre-request Script Tests Settings Cookies

Headers 8 hidden

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	Authorization	luiz	
	Key	Value	

**POST** <http://127.0.0.1:5001/ingresar?> Send

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded **raw** binary JSON Beautify

```
1 {
2   "usuario":"luiz",
3   "contraseña":"luiz",
4   "tipo":"FREEMIUM"
5 }
```

Body Cookies Headers (5) Test Results 200 OK 5.51 s 207 B Save Response

Pretty Raw Preview Visualize HTML ⋮ 🔍

```
1 Se cargó el usuario exitosamente!
```

- En la base de datos:

```

_identificación: Id. de objeto ( ' 65ec5c31e3393059774248e0 ' )
Clave : " luiz "
API
usuario: " luiz "
contraseña: " luiz "
tipo: " GRATIS "

```

## ERRORES:

- Ingreso de nuevo al usuario:

POST http://127.0.0.1:5001/ingresar? Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary JSON Beautify

```

1 {
2   "usuario": "luiz",
3   "contraseña": "luiz",
4   "tipo": "FREEMIUM"
5 }

```

Body Cookies Headers (5) Test Results 200 OK 5.51 s 227 B Save Response

Pretty Raw Preview Visualize HTML

1 Existe el Api\_key. Cada Api\_key se asigna a un usuario

- No tiene algún parámetro:

POST http://127.0.0.1:5001/ingresar? Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary JSON Beautify

```

1 {
2   "contraseña": "luiz",
3   "tipo": "FREEMIUM"
4 }

```

Body Cookies Headers (5) Test Results 200 OK 5 ms 189 B Save Response

Pretty Raw Preview Visualize HTML

1 No tiene usuario



POST http://127.0.0.1:5001/ingresar?

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary JSON Beautify

```
1 {
2   "usuario": "luiz",
3   "contraseña": "luiz",
4   "tipo": "FREEMI"
}
```

Body Cookies Headers (5) Test Results 200 OK 3 ms 180 B Save Response

Pretty Raw Preview Visualize HTML

```
1 Mal tipo
```

### Caso de prueba: POST: http://127.0.0.1:5002/auth?

http://127.0.0.1:5000/ingresarUsuarios? Save

POST http://127.0.0.1:5002/auth? Send

Params Authorization **Headers (9)** Body Pre-request Script Tests Settings Cookies

Headers 8 hidden

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	Authorization	luiz	
	Key	Value	

Body Cookies Headers (5) Test Results 200 OK 7 ms 191 B Save Response

Pretty Raw Preview Visualize HTML

```
1 API key autorizada
```

POST

http://127.0.0.1:5002/auth?

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

Headers

8 hidden

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	Authorization	alberto	
	Key	Value	

Body

Cookies

Headers (5)

Test Results

401 UNAUTHORIZED5 ms298 B

Save Response

Pretty

Raw

Preview

Visualize

HTML

```
1 <!doctype html>
2 <html lang=en>
3 <title>401 Unauthorized</title>
4 <h1>Unauthorized</h1>
5 <p>API key no autorizada</p>
```

## Caso de prueba: http://127.0.0.1:5003/logger?

POST

http://127.0.0.1:5003/logger?

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

JSON

Beautify

```
1 {
2   "colesterol":2.4,
3   "presion":0.7,
4   "glucosa":0.8,
5   "edad":2,
6   "sobrepeso":0,
7   "tabaquismo":0
8 }
```

Body

Cookies

Headers (5)

Test Results

200 OK5.29 s207 B

Save Response

Pretty

Raw

Preview

Visualize

HTML

```
1 guardé en bitácora exitosamente!
```

```

└─ _identificación: Id. de objeto(' 65ec5f994f27fe40ccfff6b4 ')
   marca de : " 2024-03-09T10:09:45.602341 "
   tiempo
   ▼ pará... : Matriz (7)
      0: 0
      1: 2.4000000953674316
      2: 0.699999988079071
      3: 0.800000011920929
      4: 2
      5: 0
      6: 0
   ▼ respue... : Matriz (1)
      ▼ 0: Matriz (1)
         0: 0.00011644
```

- Si falta algún parámetro o es incorrecto:

POST

http://127.0.0.1:5003/logger?

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

JSON

Beautify

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

body

Cookies

Headers (5)

Test Results

500 INTERNAL SERVER ERROR

12 ms

342 B

Save Response

Pretty

Raw

Preview

Visualize

HTML

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535