

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA  
CENTRO UNIVERSITARIO DE OCCIDENTE  
DIVISIÒN DE CIENCIAS DE LA INGENIERÍA



MANUAL TECNICO PYRAMID AVL  
ESTRUCTURA DE DATOS

PRESENTADO POR:

DANIEL EDUARDO BAUTISTA FUENTES

201930588

DOCENTE  
ING. OLIVER SIERRA  
AUXILIAR  
ODRA69


QUETZALTENANGO - QUETZALTENANGO - GUATEMALA  
09/05/2022

# ÍNDICE

Manual de aplicación.	1
Estructura AVL	14
Insertar nodos	14
Rotaciones en nodos	15
Rotación simple a la izquierda (RI)	15
Rotación simple a la derecha (RD)	15
Rotación derecha izquierda (RI)	16
Rotación simple a la izquierda (ID)	16
Rotación simple a la izquierda (DI)	16
Búsqueda de nodo	17
Eliminar nodos	17

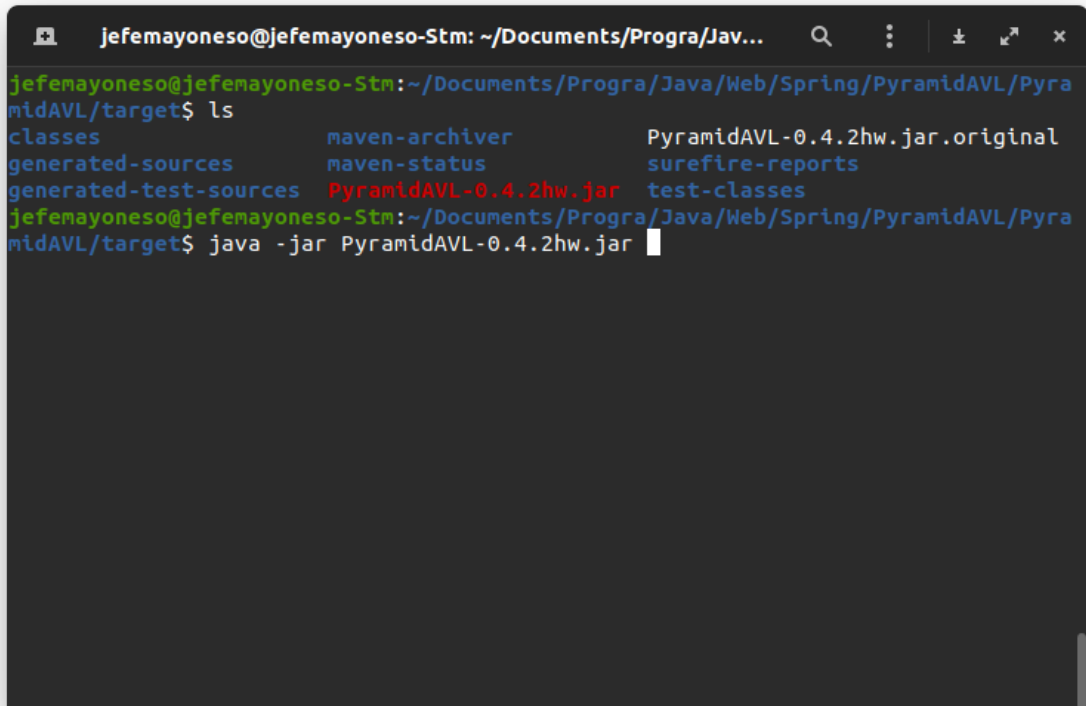
# Manual de aplicación.

1. Descargar ngrok

A terminal window with a dark background and light green text. The window title bar shows 'jefemayoneso@jefemayoneso-Stm: ~/Other Apps/ngrok'. The terminal content shows the user running 'ls' and then 'ngrok' in the directory ~/Other Apps/ngrok.

```
jefemayoneso@jefemayoneso-Stm: ~/Other Apps/ngrok
jefemayoneso@jefemayoneso-Stm:~/Other Apps/ngrok$ ls
ngrok
jefemayoneso@jefemayoneso-Stm:~/Other Apps/ngrok$
```

2. Correr la aplicacion .jar, esta aplicación abrirá tomcat, debemos tener instalado tomcat. Hemos usado spring, y usamos el puerto 8080, así que si existe un proceso en progreso en el puerto 8080, la aplicación no iniciará.



```
jefemayoneso@jefemayoneso-Stm: ~/Documents/Progra/Jav...
jefemayoneso@jefemayoneso-Stm:~/Documents/Progra/Java/Web/Spring/PyramidAVL/PyramidAVL/target$ ls
classes                               maven-archiver                    PyramidAVL-0.4.2hw.jar.original
generated-sources                    maven-status                      surefire-reports
generated-test-sources               PyramidAVL-0.4.2hw.jar            test-classes
jefemayoneso@jefemayoneso-Stm:~/Documents/Progra/Java/Web/Spring/PyramidAVL/PyramidAVL/target$ java -jar PyramidAVL-0.4.2hw.jar
```

```
jefemayoneso@jefemayoneso-Stm: ~/Documents/Progra/Jav...
2022-05-09 01:13:00.184 INFO 110231 --- [main] Application.PyramidAvlApplication : No active profile set, falling back to 1 default profile: "default"
2022-05-09 01:13:01.281 INFO 110231 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-05-09 01:13:01.292 INFO 110231 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-05-09 01:13:01.292 INFO 110231 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.60]
2022-05-09 01:13:01.375 INFO 110231 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-05-09 01:13:01.375 INFO 110231 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1127 ms
2022-05-09 01:13:01.789 WARN 110231 --- [main] ion$DefaultTemplateResolverConfiguration : Cannot find template location: classpath:/templates/ (please add some templates or check your Thymeleaf configuration)
2022-05-09 01:13:01.844 INFO 110231 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path '/'
2022-05-09 01:13:01.854 INFO 110231 --- [main] Application.PyramidAvlApplication : Started PyramidAvlApplication in 2.129 seconds (JVM running for 2.548)
```

3. Correr el comando `./ngrok http 8080` <- si queremos conectarnos mediante http
  - a. Alternativamente podemos correr `./ngrok tcp 8080` si queremos conectarnos mediante ssh ton tcp



```
jefemayoneso@jefemayoneso-Stm: ~/Other Apps/ngrok
jefemayoneso@jefemayoneso-Stm:~/Other Apps/ngrok$ ls
ngrok
jefemayoneso@jefemayoneso-Stm:~/Other Apps/ngrok$ ./ngrok http 8080
```



```
ngrok (Ctrl+C to quit)
Session Status      online
Account             danielbautista201930588@cunoc.edu.gt (Plan: Free)
Version             3.0.3
Region             United States (us)
Latency             calculating...
Web Interface       http://127.0.0.1:4040
Forwarding           https://4dc9-45-188-3-63.ngrok.io -> http://localhost:8080

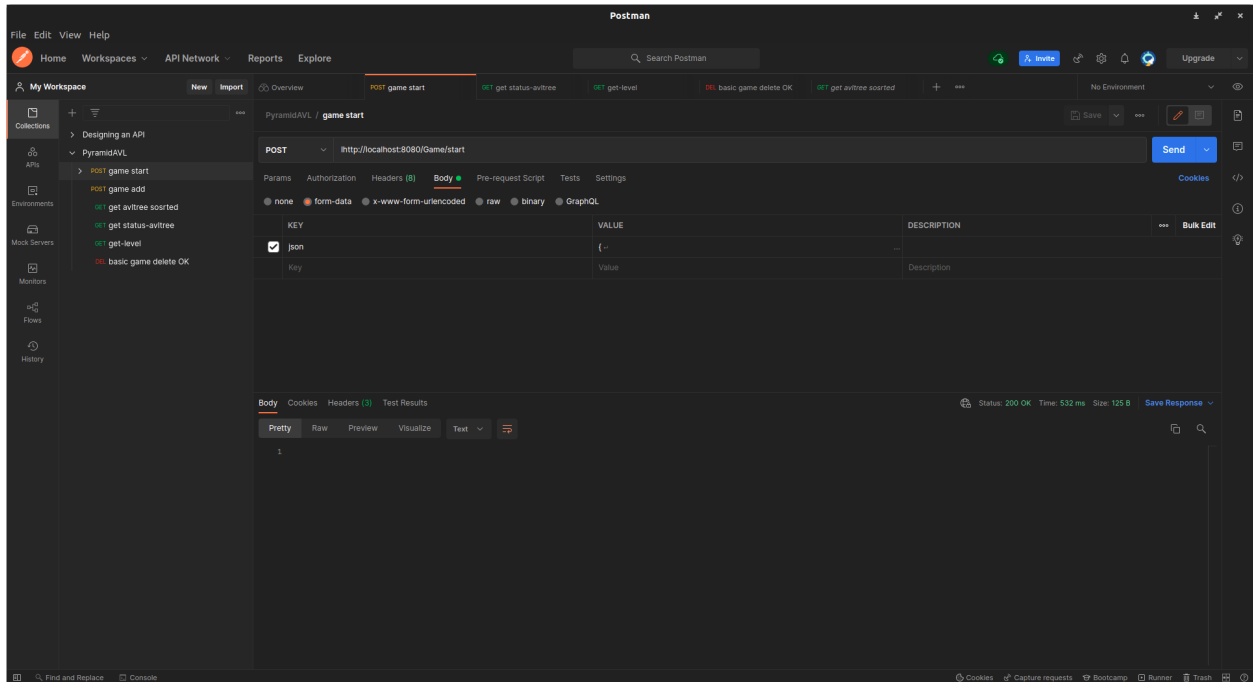
Connections      ttl    opn    rt1    rt5    p50    p90
0               0      0.00   0.00   0.00   0.00
```

4. Una vez iniciado, obtendremos una IP temporal, esta es la IP que compartiremos para que se conecten remotamente a nuestro servidor local.

`https://4dc9-45-188-3-63.ngrok.io -> http://localhost:8080`

5. Preparamos las peticiones, en este caso he usado postman. Las rutas son las siguientes
  - a. Game/start: enviamos una petición POST con una variable de nombre json que contenga las cartas a insertar para iniciar el juego.

```
{  
  "0": "2♣",  
  "1": "3♦",  
  "2": "7♣",  
  "3": "4♦",  
  "4": "8♣",  
  "5": "9♥",  
  "6": "10♣",  
  "8": "2♦",  
  "9": "8♦",  
  "10": "K♣",  
}
```



Obtendremos la siguiente respuesta



Y en caso de error veríamos lo siguiente



Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON ↕

```
1 {  
2   "message": "Valor invalido en JSON, llave=3 valor=♦",  
3   "status": "BAD_REQUEST",  
4   "timestamp": "2022-05-09T07:27:12.03273Z"  
5 }
```

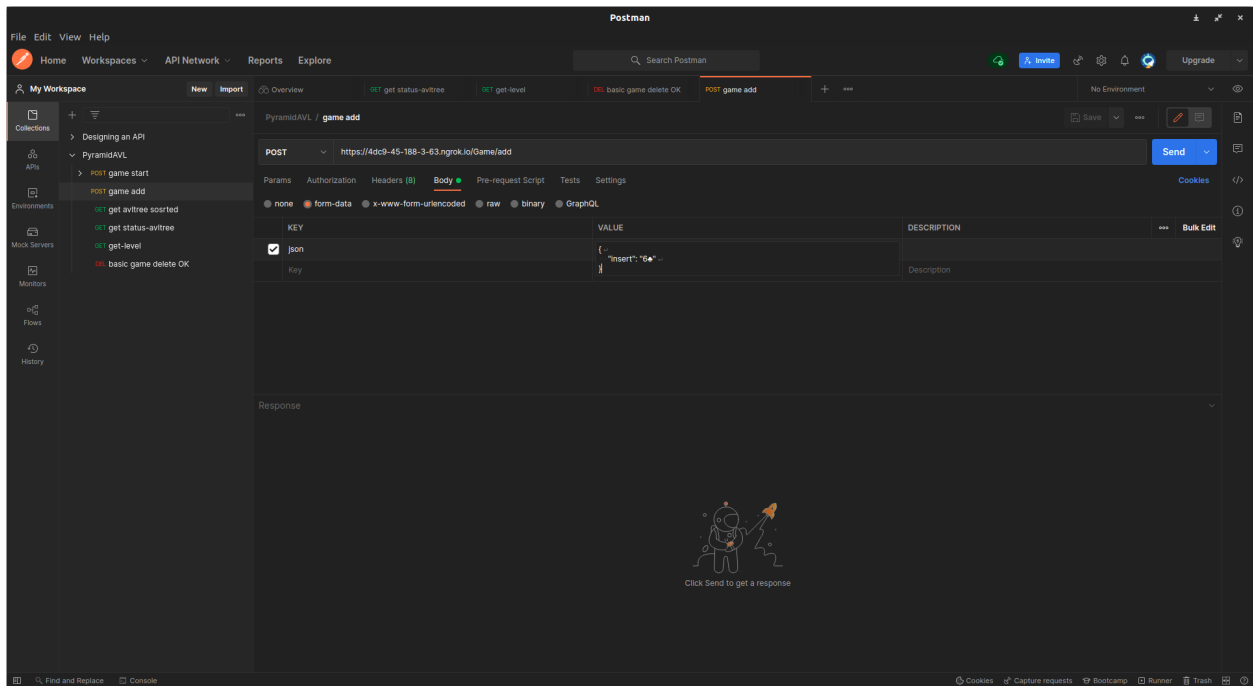
Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ↕

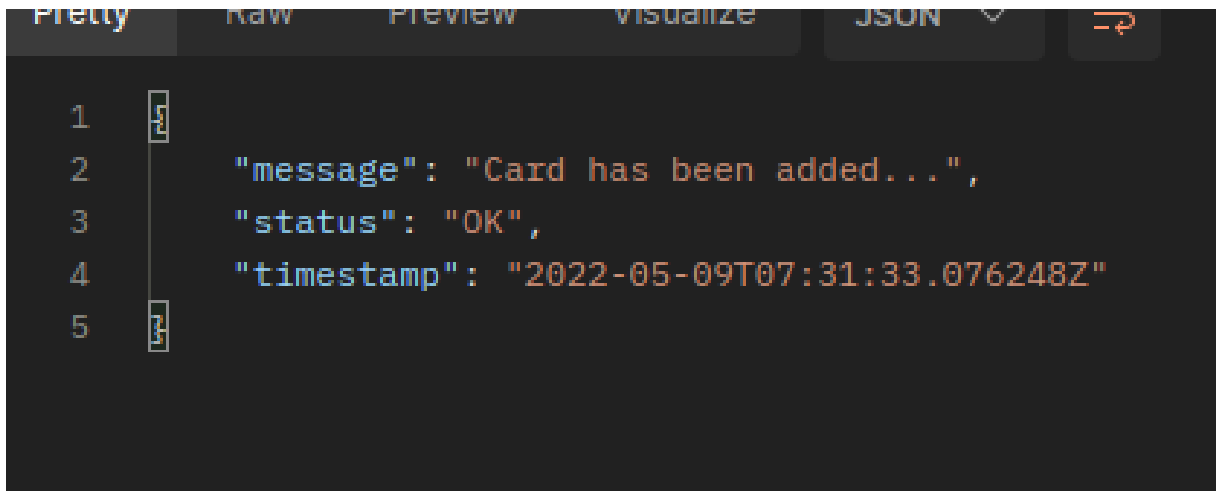
```
1 {  
2   "message": "La carta 4♦ ya ha sido insertada ",  
3   "status": "NOT_ACCEPTABLE",  
4   "timestamp": "2022-05-09T07:28:09.046434Z"  
5 }
```

b. Game/add: Mandaremos un archivo JSON con el nombre de “json”

```
{  
  "insert": "6♣"  
}
```

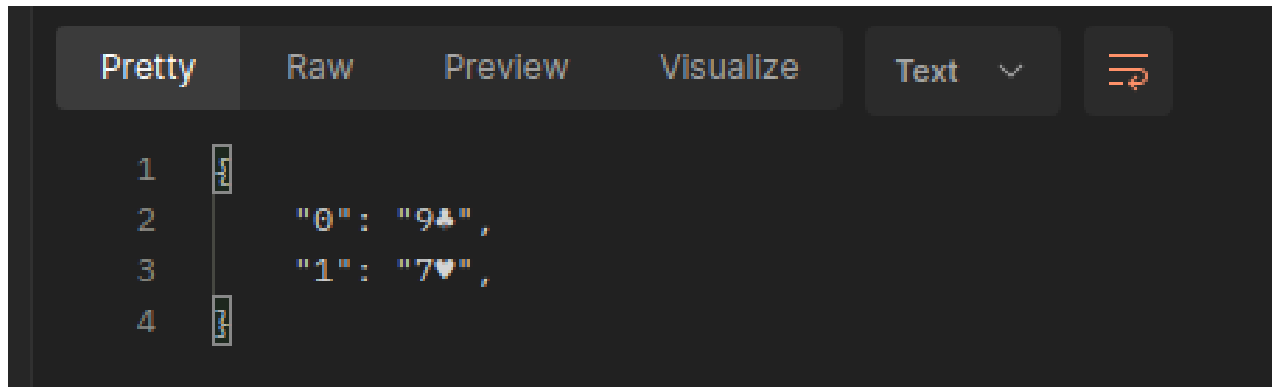


Obtendremos la siguiente respuesta



c. Game/get-level?level=NUMERO

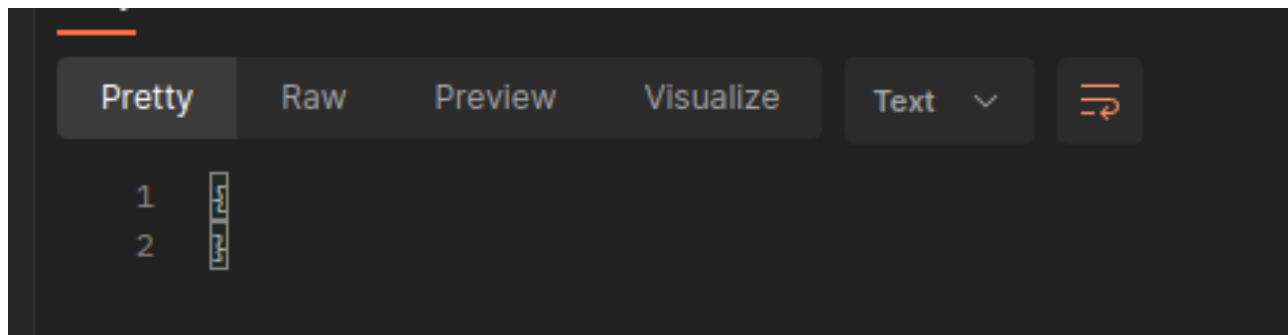
Mandaremos una petición GET con un NÚMERO entero, si existe alguna carta en el nivel veremos lo siguiente.



The screenshot shows a JSON viewer interface with tabs for 'Pretty', 'Raw', 'Preview', and 'Visualize'. The 'Pretty' tab is selected. The JSON data is displayed as follows:

```
1 {  
2   "0": "9♠",  
3   "1": "7♥",  
4 }
```

Si el nivel no existe veremos lo siguiente



The screenshot shows a JSON viewer interface with tabs for 'Pretty', 'Raw', 'Preview', and 'Visualize'. The 'Pretty' tab is selected. The JSON data is displayed as follows:

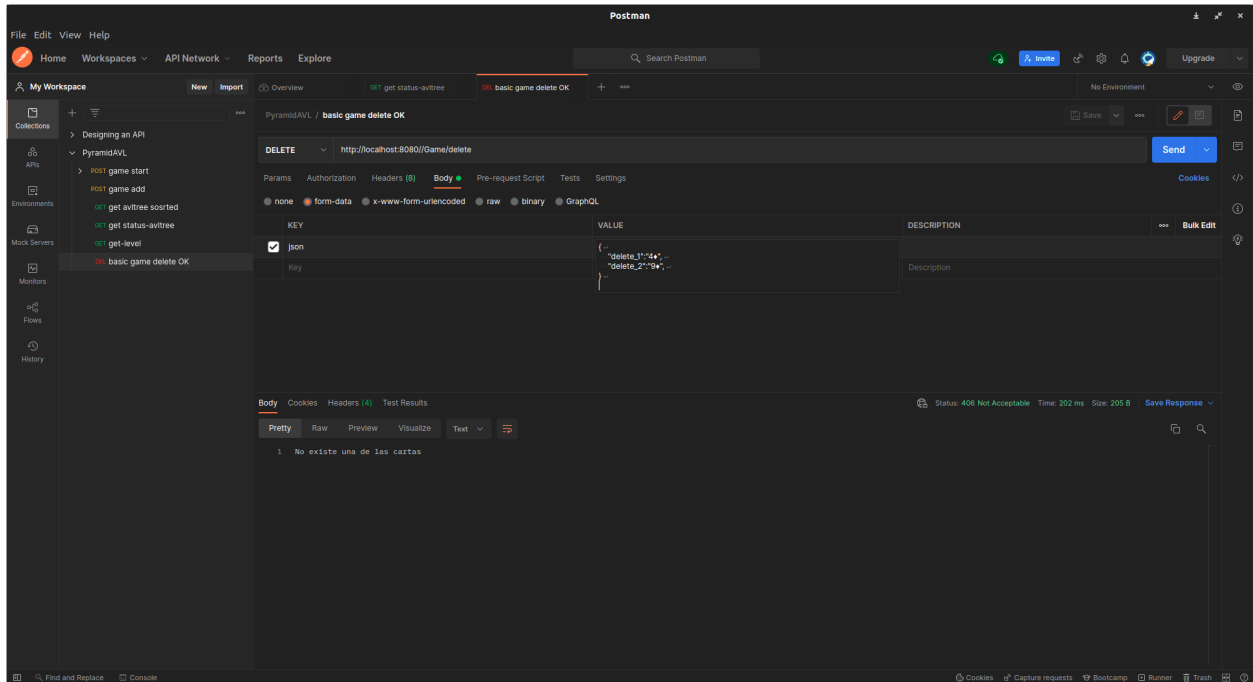
```
1 {  
2   "error": "Level not found",  
3 }
```

d. Game/delete

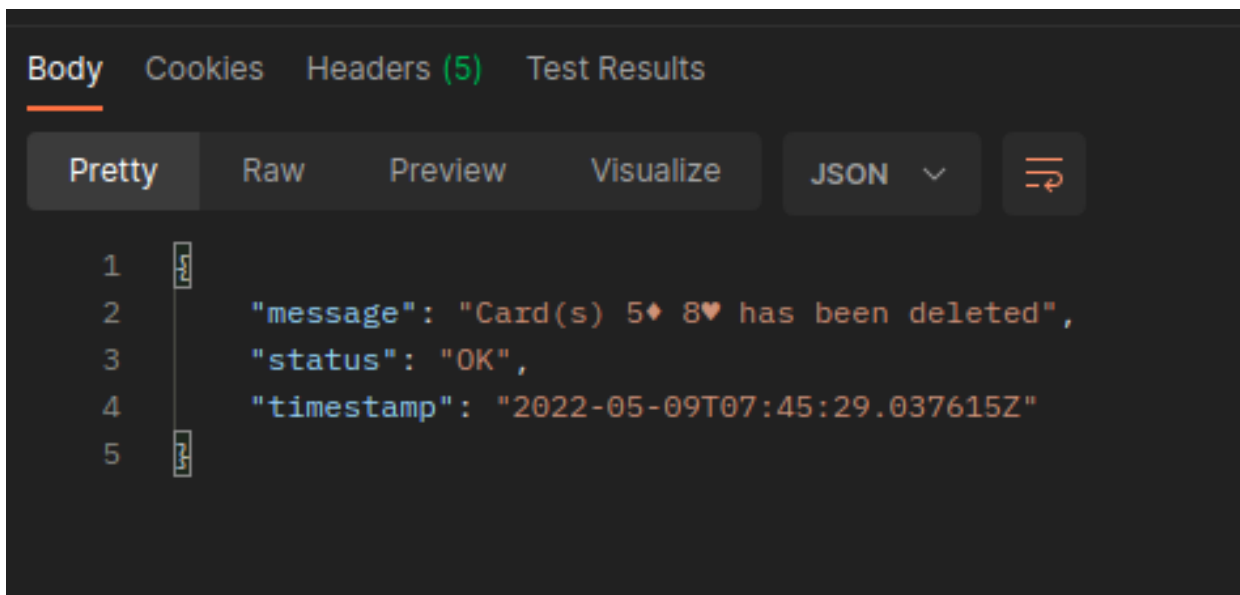
Enviaremos una petición POST con un archivo que se llame JSON, el cual debe tener la siguiente estructura

```
{  
  "delete_1": "4♦",  
  "delete_2": "9♦",  
}
```

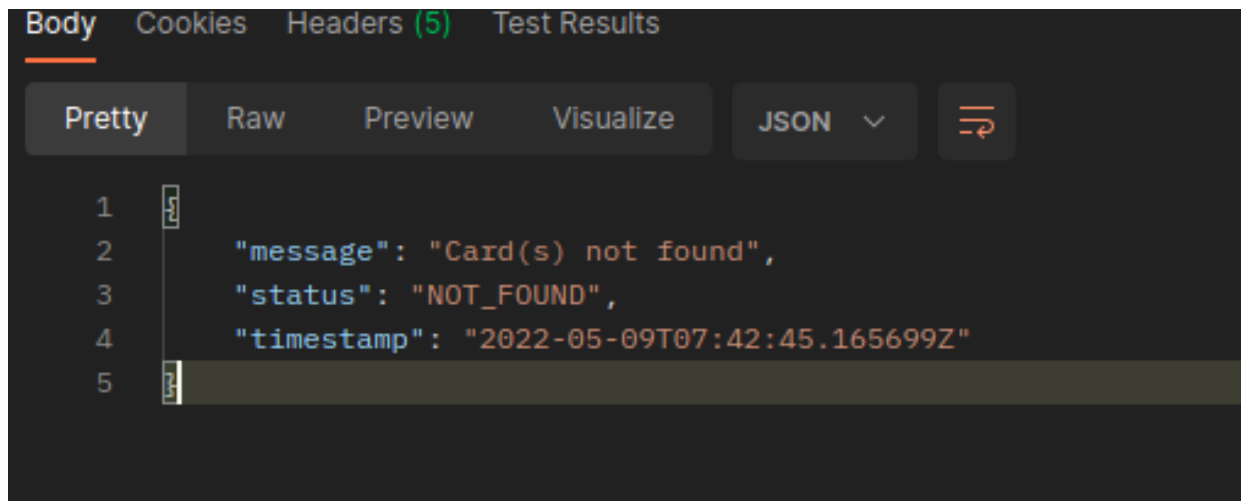
La suma de las cartas debe ser de 13 o de lo contrario retornará un mensaje de solicitud inválida.



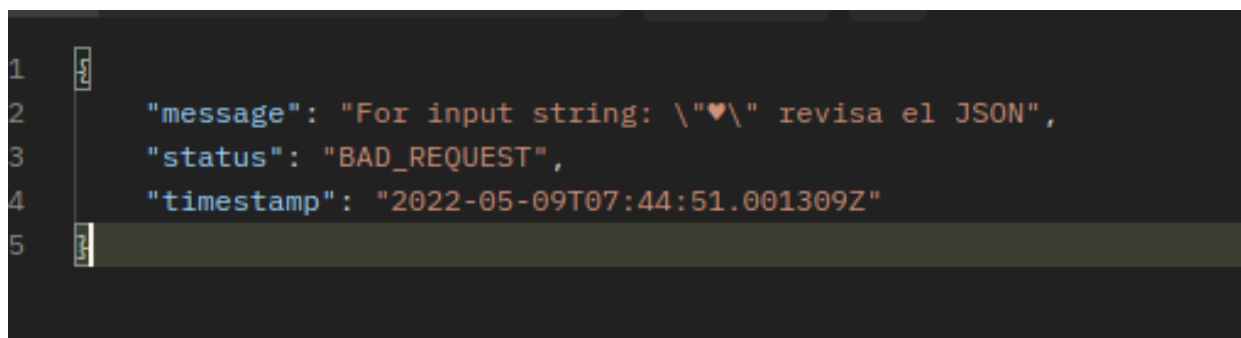
Obtendremos la siguiente respuesta.



De lo contrario obtendremos mensajes informando del error sucedido.



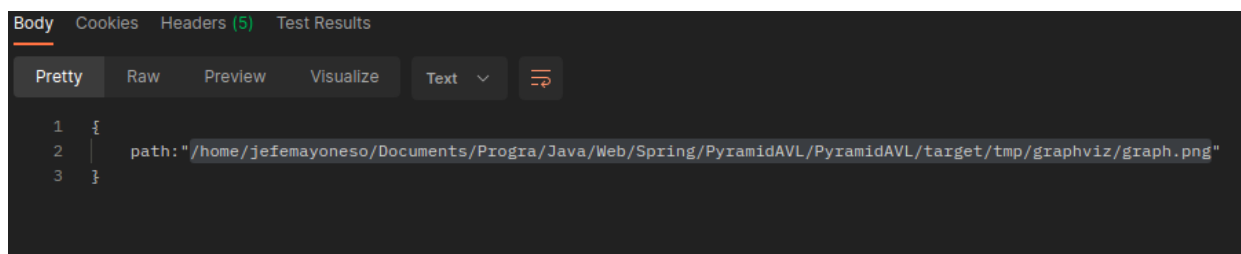
```
Body Cookies Headers (5) Test Results
Pretty Raw Preview Visualize JSON
1 {
2   "message": "Card(s) not found",
3   "status": "NOT_FOUND",
4   "timestamp": "2022-05-09T07:42:45.165699Z"
5 }
```



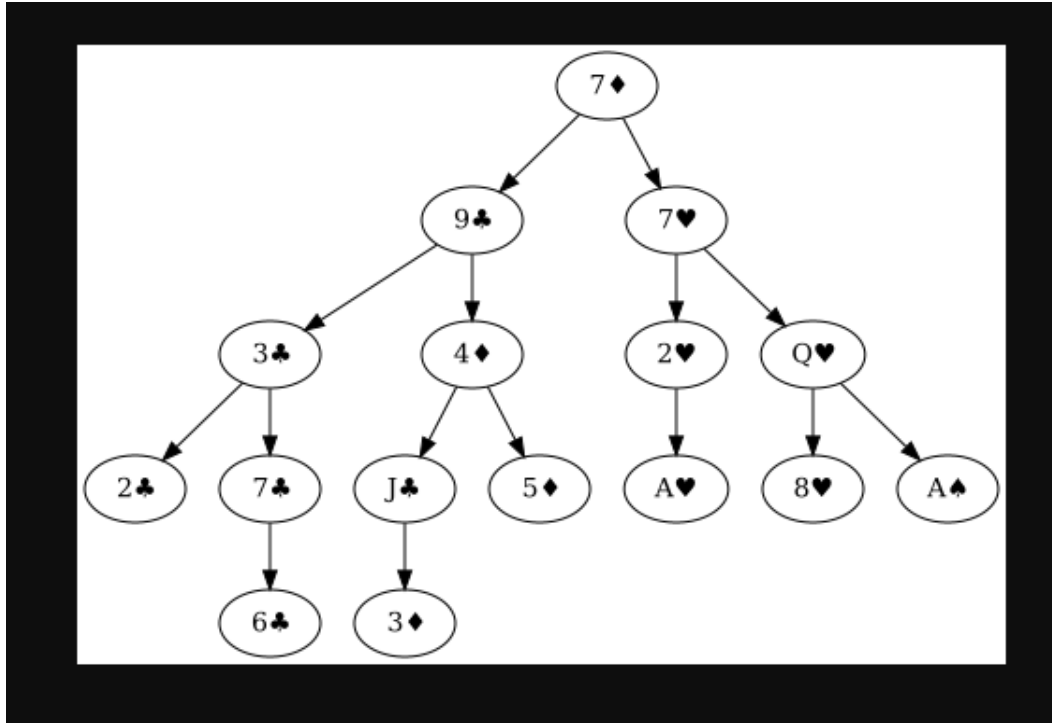
```
1 {
2   "message": "For input string: \"♥\" revisa el JSON",
3   "status": "BAD_REQUEST",
4   "timestamp": "2022-05-09T07:44:51.001309Z"
5 }
```

e. Game/status-avltree

Enviaremos una petición GET para obtener una imagen con el status del árbol.



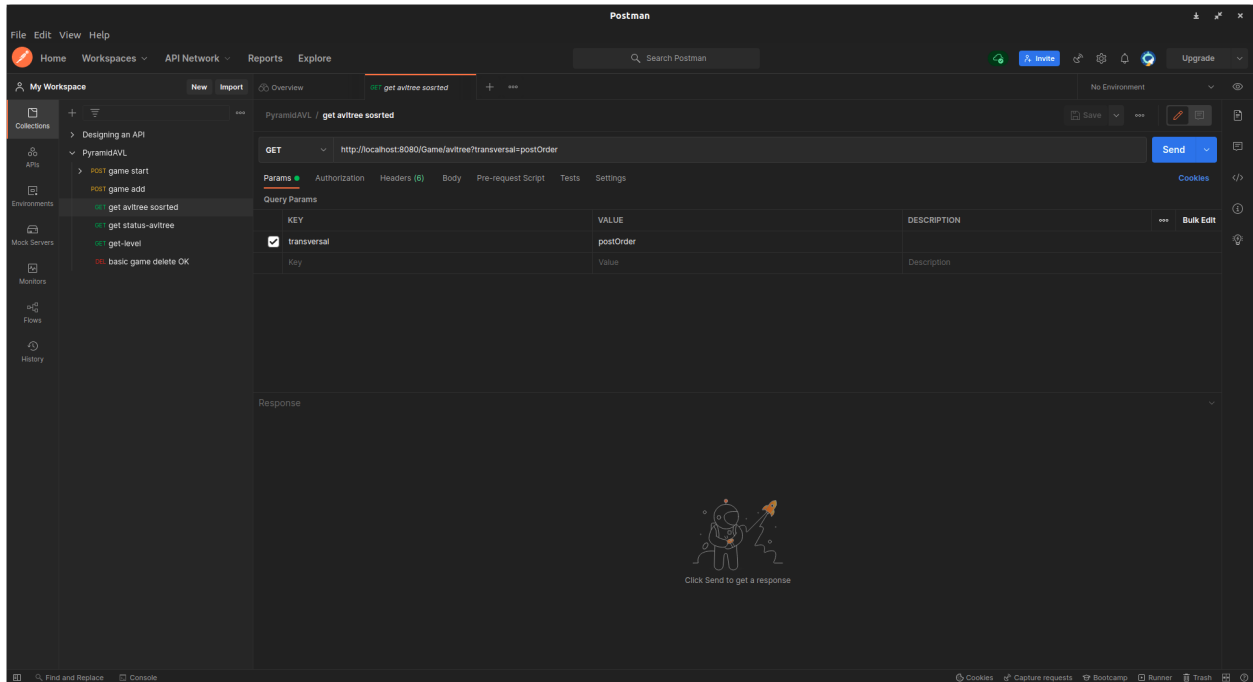
```
Body Cookies Headers (5) Test Results
Pretty Raw Preview Visualize Text
1 {
2   path: "/home/jefemayoneso/Documents/Progra/Java/Web/Spring/PyramidAVL/PyramidAVL/target/tmp/graphviz/graph.png"
3 }
```



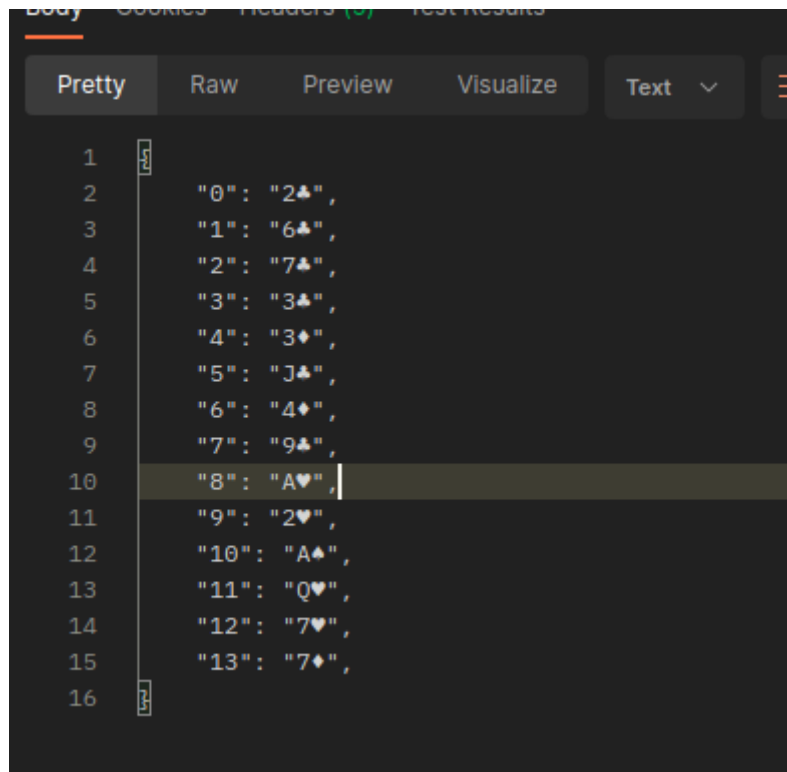
f. `Game/avltree?transversal=ACTION`

ACTION puede ser `preOrder`, `postOrder` o `inOrder`.

Enviaremos una petición GET con un atributo `transversal` y un valor ACTION.



Obtendremos la siguiente respuesta



# Estructura AVL

Un árbol AVL es un árbol de derivación que cuenta únicamente con 2 derivaciones, 1 por la derecha y una por la izquierda. El objetivo de utilizar un árbol AVL es la optimización al momento de realizar búsquedas o eliminaciones. Pues el árbol siempre debe estar balanceado, o al menos, la mayor parte del tiempo.

Se dice que un árbol AVL está balanceado si la resta del factor de equilibrio (FE) del hijo izquierdo con el FE del hijo derecho son 0, y está recargado hacia un lado si

1. El árbol está cargado hacia la derecha si la resta de FE es de 2

## Insertar nodos

Para poder insertar en un árbol AVL usamos una “carga”, esta carga nos sirve para saber si insertar a la izquierda o derecha de un nodo. Si el valor a insertar es menor al valor del nodo actual, nos movemos hacia la izquierda. Si es mayor, a la derecha. Y así hasta encontrar un punto en el que un nodo hijo  $k$  tiene un hijo  $k+1$  nulo que su valor sea mayor o menor que el peso del nodo a insertar.

Es importante aclarar, que una vez realizada una inserción, debemos volver a checar si el árbol está desbalanceado, y en caso de estarlo, balanceamos el árbol.



## Rotaciones en nodos

Ahora bien, para saber cómo balancear el árbol AVL tenemos 4 métodos

1. Rotación simple a la izquierda (RI)
2. Rotación simple a la derecha (RD)
3. Rotación izquierda derecha ó doble rotación a la derecha (DI)
4. Rotación derecha izquierda ó doble rotación a la izquierda (ID)

Rotación simple a la izquierda (RI)

Rotación simple a la derecha (RD)

Si el FE del hijo izquierdo es 2 veces mayor que el FE del hijo derecho y la raíz del subárbol izquierdo tiene un FE de 1, significa que el árbol está cargado a la izquierda, por lo tanto, se realiza una rotación a la izquierda.

Proceso:

1.  $K2 = \text{hijo derecho de } K1$
2.  $\text{Hijo derecho de } K1 = \text{Hijo izquierdo de } K2$
3.  $\text{Hijo izquierdo de } K2 = K1$
4. Retornar  $K2$  para tomar el antiguo lugar de  $K1$

Rotación derecha izquierda (RI)

Si el FE del hijo derecho es 2 veces más alto que el FE izquierdo y la raíz del subárbol derecho tiene un FE de -1 está cargado a la derecha y se corre un RI.

Proceso:

1.  $K1 = \text{hijo izquierdo de } K2$
2.  $\text{Hijo izquierdo de } K2 = \text{Hijo derecho de } K1$
3.  $\text{Hijo derecho de } K1 = K2$
4. Retornamos  $K1$  para que tome el valor anterior de  $K2$

Rotación simple a la izquierda (ID)

Si el FE del hijo derecho es 2 veces más alto que el FE izquierdo y la raíz del subárbol derecho tiene un FE de -2 está cargado a la derecha y se corre un RI.

Proceso:

1.  $K2 = \text{Hijo derecho de } K1$
2. RI de  $K2$
3. RD de  $K1$

Rotación simple a la izquierda (DI)

Si el FE del hijo derecho es 2 veces más alto que el FE izquierdo y la raíz del subárbol derecho tiene un FE de 2 está cargado a la derecha y se corre un RI.

Proceso:

4.  $K1 = \text{Hijo izquierdo de } K2$
5. RD de  $K1$
6. RI de  $K2$

## **Búsqueda de nodo**

Cómo conocemos el peso de cada nodo. Partimos de una raíz, si el peso es mayor al nodo actual nos movemos al hijo derecho, si es menor nos movemos al hijo izquierdo. Repetimos recursivamente hasta encontrar un nodo que tenga un peso igual al que buscamos. Al encontrarlo, hemos encontrado el nodo buscado.

## **Eliminar nodos**

Primero debemos buscar un nodo, al encontrarlo, nos aseguramos que los hijos del nodo a borrar, en caso existan, apunten al padre del nodo a borrar, luego borramos el nodo y equilibramos el árbol.