

Universidad de San Carlos de Guatemala

Centro Universitario de Occidente

División Ciencias de la Ingeniería

Estructura de datos

Ing. Oliver Ernesto Sierra Pac

### **Segundo Proyecto**

**Nombre:**

Pedro Ricardo Gordillo González

**Registro Académico:**

202031683

Quetzaltenango, mayo 2022

## Manual de usuario

### Requisitos del sistema

Requerimientos de hardware

- Equipo, teclado, mouse, monitor.

Requerimientos de software

- Sistema operativo (Windows 7 en adelante)
- Java 8.0 mínimo Herramientas utilizadas JAVA

### Java

El lenguaje de programación de Java es una herramienta de desarrollo orientada a objetos, fue diseñado para que no dependieran en muchas implementaciones, el cual permite a los desarrolladores ejecutar en cualquier dispositivo sin necesidad de recompilar el código, el cual se considera multiplataforma.

Instalación de software JAVA desde el siguiente link: <https://www.java.com/es/>

En el caso de Windows, su forma de instalación es sencilla ya que solamente se descarga un archivo .exe y se siguen las instrucciones que muestra la pantalla. En caso de ser otro sistema operativo, su forma de instalación varia.

### Git Bash

Git Bash es una aplicación para entornos de Microsoft Windows que ofrece una capa de emulación para una experiencia de líneas de comandos de Git. Bash es el acrónimo en inglés de Bourne Again Shell. Una shell es una aplicación de terminal que se utiliza como interfaz con un sistema operativo mediante comandos escritos.

### Spring Boot

Spring Boot es una tecnología que nos permite crear aplicaciones autocontenidas, con esto nos podemos olvidar de la arquitectura y enfocarnos únicamente en desarrollo, delegando a Spring Boot labores como configuración de dependencias, desplegar nuestro servicio o aplicación a un servidor de aplicaciones y enfocarnos únicamente en crear nuestro código.

Para esto Spring Boot utiliza internamente un servidor de aplicaciones embebido, por defecto utiliza Tomcat, pero también lo podemos hacer con jetty o undertow.

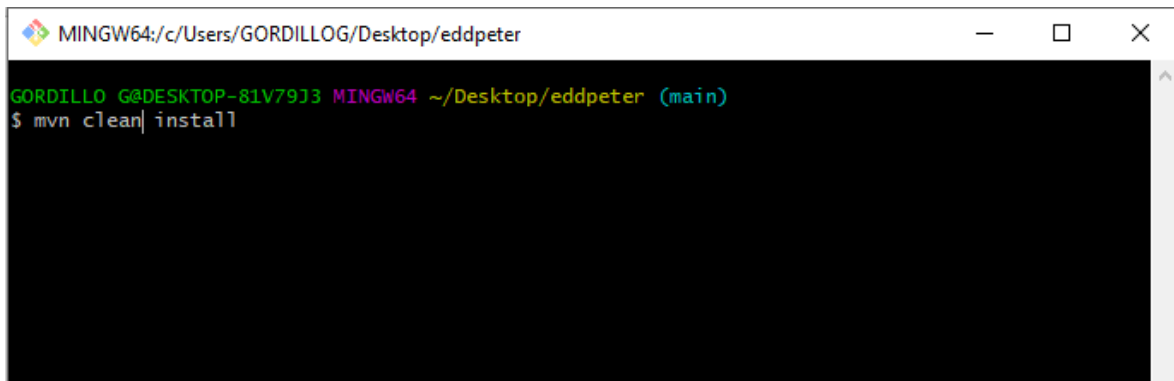
## Ngrok

Ngrok es un servicio o herramienta que te permite convertir tu servidor local en un servidor accesible mediante un subdominio generado aleatoriamente por ngrok y así poder visualizarlo desde cualquier computadora con acceso a internet en el mundo.

### Ejecutar servidor

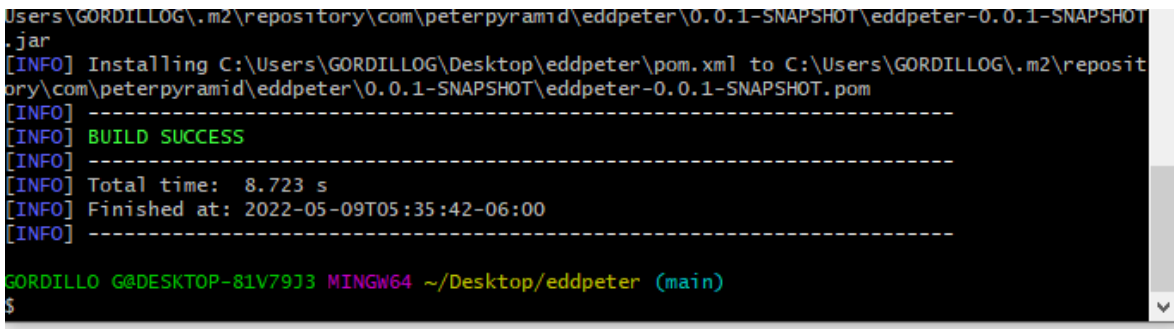
#### Ejecutar de forma local

Para ejecutar la api se hace uso del símbolo de sistema (cmd) o de la Git Bash, para ello se ingresa a la carpeta raíz que contiene el proyecto y se ejecuta el comando “mvn clean install”, enseguida se empezara el proceso que compila el código.



```
MINGW64:/c:/Users/GORDILLOG/Desktop/eddpeter
GORDILLO G@DESKTOP-81V79J3 MINGW64 ~/Desktop/eddpeter (main)
$ mvn clean install
```

Al presionar la tecla “enter” y si no se encuentra ningún error se muestra lo siguiente.



```
Users\GORDILLOG\.m2\repository\com\peterpyramid\eddpeter\0.0.1-SNAPSHOT\eddpeter-0.0.1-SNAPSHOT.jar
[INFO] Installing C:\Users\GORDILLOG\Desktop\eddpeter\pom.xml to C:\Users\GORDILLOG\.m2\repository\com\peterpyramid\eddpeter\0.0.1-SNAPSHOT\eddpeter-0.0.1-SNAPSHOT.pom
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 8.723 s
[INFO] Finished at: 2022-05-09T05:35:42-06:00
[INFO] -----
GORDILLO G@DESKTOP-81V79J3 MINGW64 ~/Desktop/eddpeter (main)
$
```

A continuación, se ingresa el comando “cd target/” para ingresar a la carpeta que contiene el jar generado.

Se ejecuta a continuación el comando “java -jar eddpeter-0.0.1-SNAPSHOT.jar”. Al realizar esto se ejecutará el servidor toomcat que se encuentra integrado a la aplicación ya que se utiliza Spring Boot.

```
MINGW64:/c/Users/GORDILLOG/Desktop/eddpeter/target
GORDILLO G@DESKTOP-81V79J3 MINGW64 ~/Desktop/eddpeter (main)
$ cd target/
GORDILLO G@DESKTOP-81V79J3 MINGW64 ~/Desktop/eddpeter/target (main)
$ java -jar eddpeter-0.0.1-SNAPSHOT.jar
```

```
MINGW64:/c/Users/GORDILLOG/Desktop/eddpeter/target
GORDILLO G@DESKTOP-81V79J3 MINGW64 ~/Desktop/eddpeter/target (main)
$ java -jar eddpeter-0.0.1-SNAPSHOT.jar

  ____ _
 / ___ \
/ /   \ \
/_/     \_\
:: Spring Boot :: (v2.6.7)

2022-05-09 05:38:19.418 INFO 7944 --- [main] c.p.eddpeter.EddpeterApplication
: Starting EddpeterApplication v0.0.1-SNAPSHOT using Java 13.0.2 on DESKTOP-81V79J3 with PID
7944 (C:\Users\GORDILLOG\Desktop\eddpeter\target\eddpeter-0.0.1-SNAPSHOT.jar started by GORDIL
LO G in C:\Users\GORDILLOG\Desktop\eddpeter\target)
2022-05-09 05:38:19.422 INFO 7944 --- [main] c.p.eddpeter.EddpeterApplication
: No active profile set, falling back to 1 default profile: "default"
2022-05-09 05:38:20.609 INFO 7944 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServe
r : Tomcat initialized with port(s): 8080 (http)
2022-05-09 05:38:20.624 INFO 7944 --- [main] o.apache.catalina.core.StandardService
: Starting service [Tomcat]
2022-05-09 05:38:20.625 INFO 7944 --- [main] org.apache.catalina.core.StandardEngin
e : Starting Servlet engine: [Apache Tomcat/9.0.62]
2022-05-09 05:38:20.735 INFO 7944 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/]
: Initializing Spring embedded WebApplicationContext
2022-05-09 05:38:20.735 INFO 7944 --- [main] w.s.c.ServletWebServerApplicationConte
xt : Root WebApplicationContext: initialization completed in 1246 ms
2022-05-09 05:38:21.297 INFO 7944 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServe
r : Tomcat started on port(s): 8080 (http) with context path ''
2022-05-09 05:38:21.307 INFO 7944 --- [main] c.p.eddpeter.EddpeterApplication
: Started EddpeterApplication in 2.373 seconds (JVM running for 2.744)
```

Al mostrarse esto, significa que se ejecutó con éxito el servidor de forma local.

## Ejecutar de forma remota

Para ejecutar de forma remota utilizando ngrok se debe de tener iniciado el servidor de forma local siguiendo los pasos anteriormente mencionados.

Ya que se tiene ejecutando de forma local el servidor se ingresa a la terminal de preferencia y se navega hasta la carpeta donde se tienen los archivos de ngrok. A continuación, se ejecuta el código “ngrok http 8080”, donde el puerto puede variar según donde se ejecute la aplicación de forma local.

```
Símbolo del sistema
C:\Pruebas>ngrok http 8080
```

```
Símbolo del sistema - ngrok: http 8080
ngrok
(CTRL+C to quit)

Session Status      online
Account             Pedro Gordillo (Plan: Free)
Update              update available (version 3.0.3, Ctrl-U to update)
Version             3.0.2
Region              United States (us)
Latency              calculating...
Web Interface        http://127.0.0.1:4040
Forwarding            https://bb27-2803-d100-e000-1a3f-79dd-7cdd-998c-1054.ngrok.io -> http://localhost:8080

Connections
t1l    opn    rt1    rt5    p50    p90
0       0       0.00   0.00   0.00   0.00
```

De esta manera se esta ejecutando el servidor de forma remota.

## Pyramid

### Codigos de error mostrados

Error	Código de respuesta
La carta no se encuentra en el árbol avl (eliminar)	Status Code 404
Los valores de las cartas no suman 13	Status Code 406
La carta no se puede eliminar ya que cuenta con hijos	Status Code 409
La carta a insertar esta duplicada	Status Code 406
Cualquier otro error	Status Code 400

### Iniciar partida

Para iniciar una partida, utilizando postman, se ingresa la url ya sea de forma local o la que brinda ngrok.

Url: <https://bb27-2803-d100-e000-1a3f-79dd-7cdd-998c-1054.ngrok.io/Game/start>

Se insertará al árbol las siguientes cartas en formato json:

```
{
  "0": "8♣",
  "1": "10♣",
  "2": "3♥"
}
```



## Agregar carta

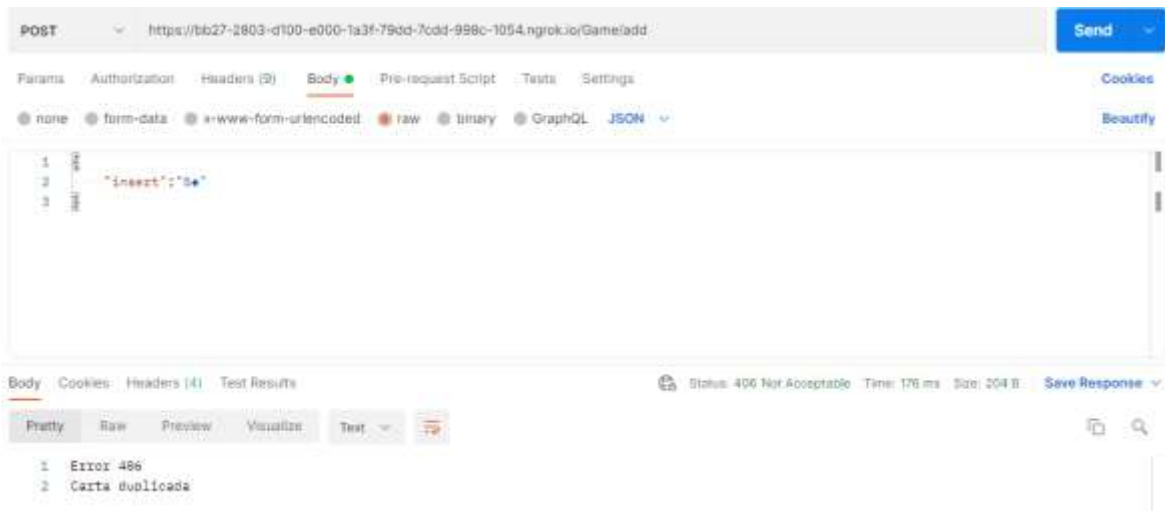
Para agregar una carta al árbol se ingresa por formato json la petición la cual es de tipo Post a la siguiente url:

<https://bb27-2803-d100-e000-1a3f-79dd-7cdd-998c-1054.ngrok.io/Game/add>



## Carta repetida

Si se intenta ingresar una carta que ya está dentro del árbol se muestra el siguiente mensaje.

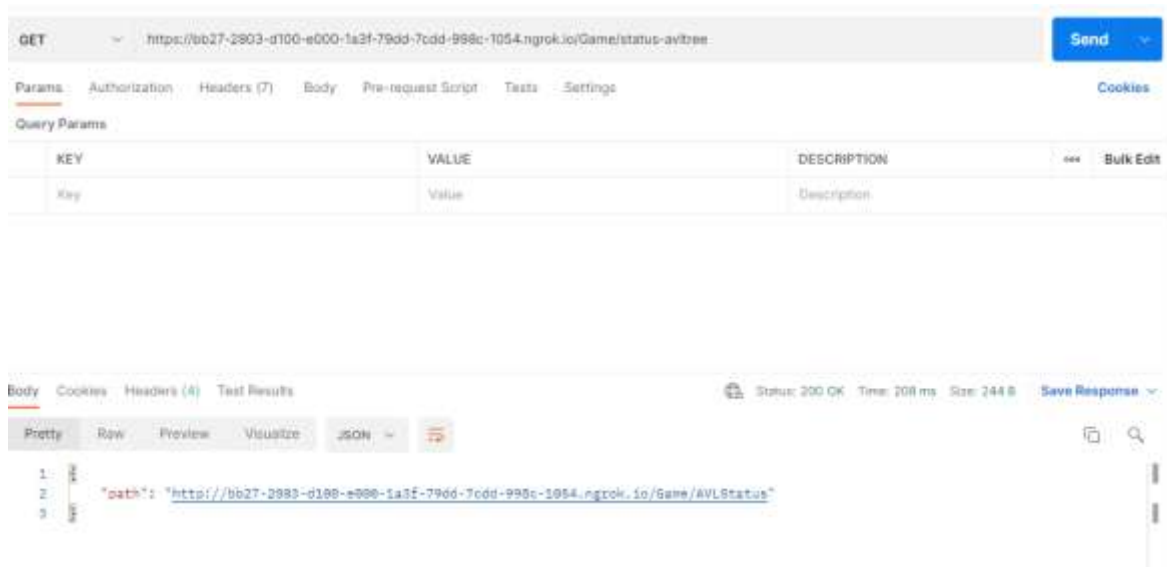


## Obtener estado del árbol

Para obtener el estado gráfico del árbol se realiza una petición de tipo get la siguiente dirección:

<https://bb27-2803-d100-e000-1a3f-79dd-7cdd-998c-1054.ngrok.io/Game/status-avltree>

La cual retorna un url en formato json donde puede observarse la imagen del estado del árbol.



Path retornado:

```
{
  "path": "http://bb27-2803-d100-e000-1a3f-79dd-7cdd-998c-1054.ngrok.io/Game/AVLStatus"
}
```

Al ingresar el path desde Postman se observa la imagen del estado del árbol.

GET <http://bb27-2803-d100-e000-1a3f-79dd-7cdd-998c-1054.ngrok.io/Game/AVLStatus>

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 183 ms

10♣

8♣ 3♥

5♠

Estado Gráfico Árbol AVL

Pedro Gordillo 202031686

## Eliminar carta

Para eliminar una carta se tiene dos condiciones, una que el valor o los dos valores de las cartas ingresadas sea igual a 13.

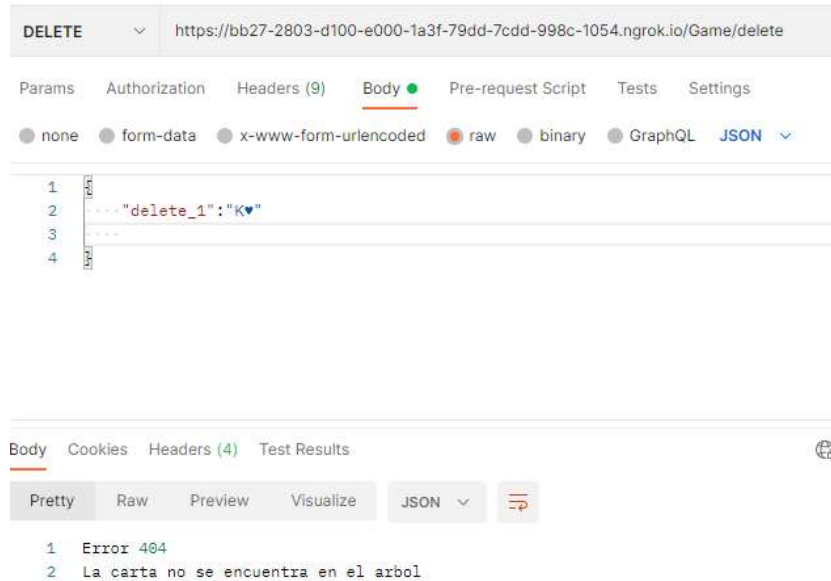
Se realiza una petición del tipo Delete a la siguiente url:

<https://bb27-2803-d100-e000-1a3f-79dd-7cdd-998c-1054.ngrok.io>

Para el ejemplo, se intentará eliminar la carta "K♥" del árbol.

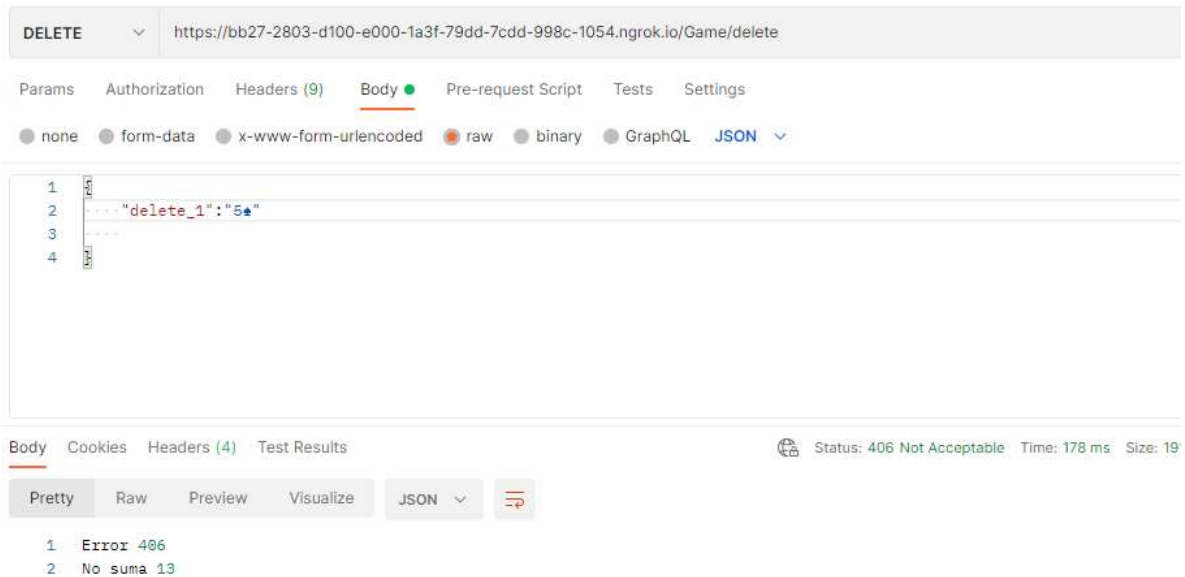
Para realizar la eliminación se utiliza un json que contienen la información de la carta a eliminar.



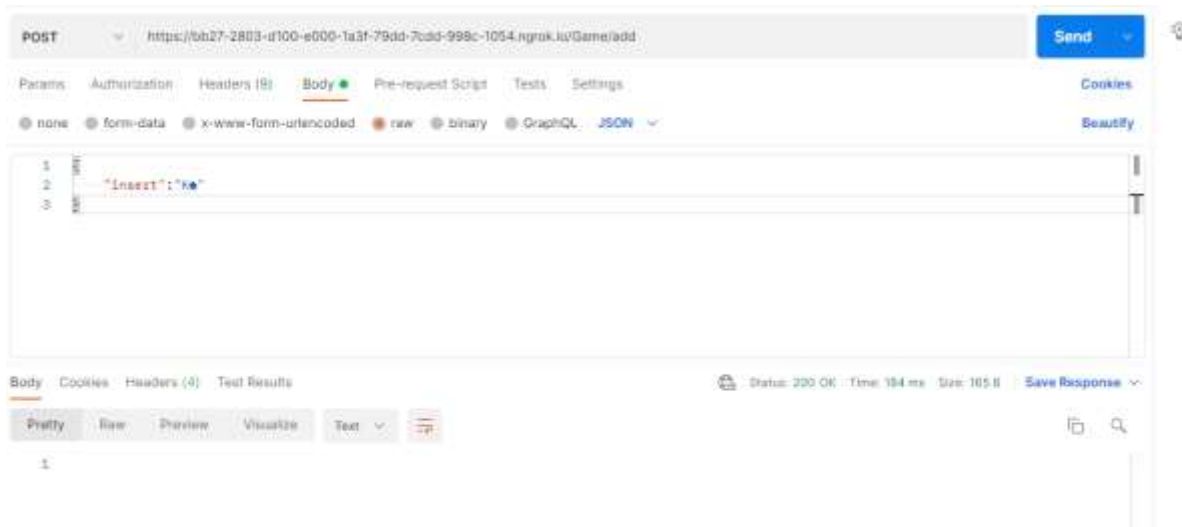


Ya que la carta no existe dentro del árbol se retorna el error y mensaje indicando que la carta no se encuentra dentro del árbol.

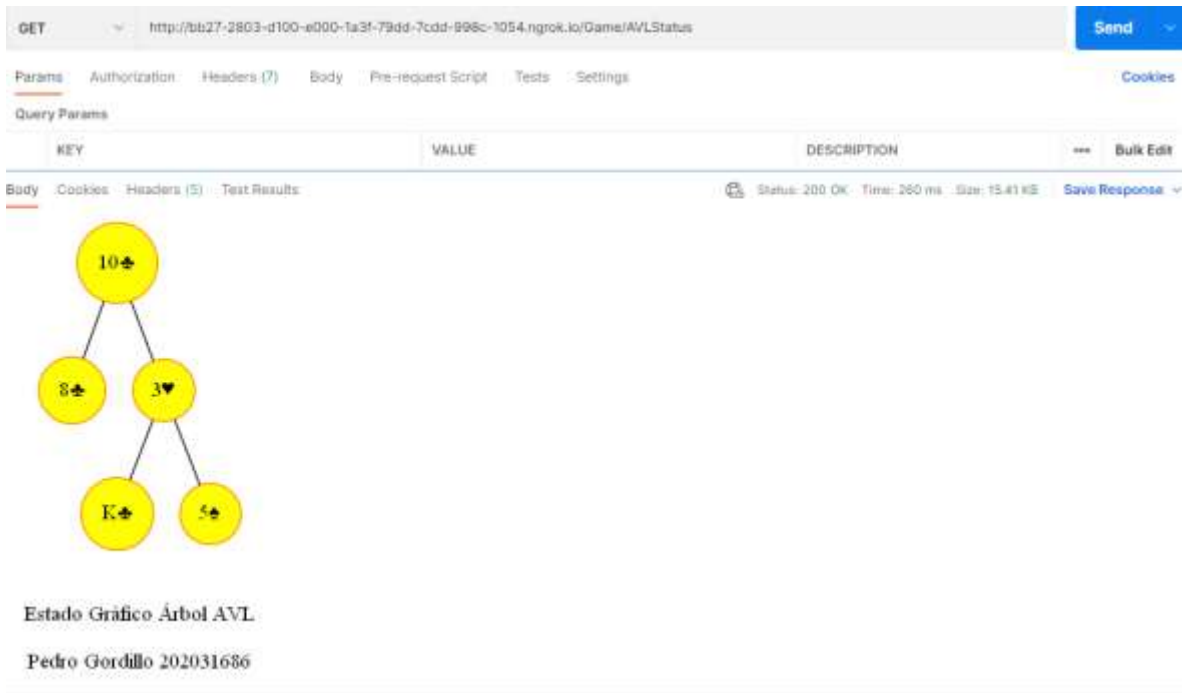
Si la carta que se se ingresa esta dentro del árbol, pero no tiene un valor real de 13 se muestra lo siguiente.



Para poder eliminar un nodo de forma correcta se agregará la carta "K♣".



Ya que no se muestran errores, se mostrará el estado de árbol para verificar que se ingresó correctamente.



Para eliminar el nodo "K♣" se utiliza el método Delete donde a través de un json se especifica la carta a eliminar. En este caso se envía el json de la siguiente forma

```

{
  "delete_1": "K♣"
}

```



Ya que no se muestra algún error de parte del servidor, se visualizará el estado del árbol para verifica que se eliminó correctamente.

GET http://bb27-2803-d100-e000-1a3f-79dd-7cdd-998c-1054.ngrok.io/Game/AVLStatus			
Params Authorization Headers (7) Body Pre-request Script Tests Settings			
Query Params			
KEY	VALUE	DESC	
Key	Value	Desc	

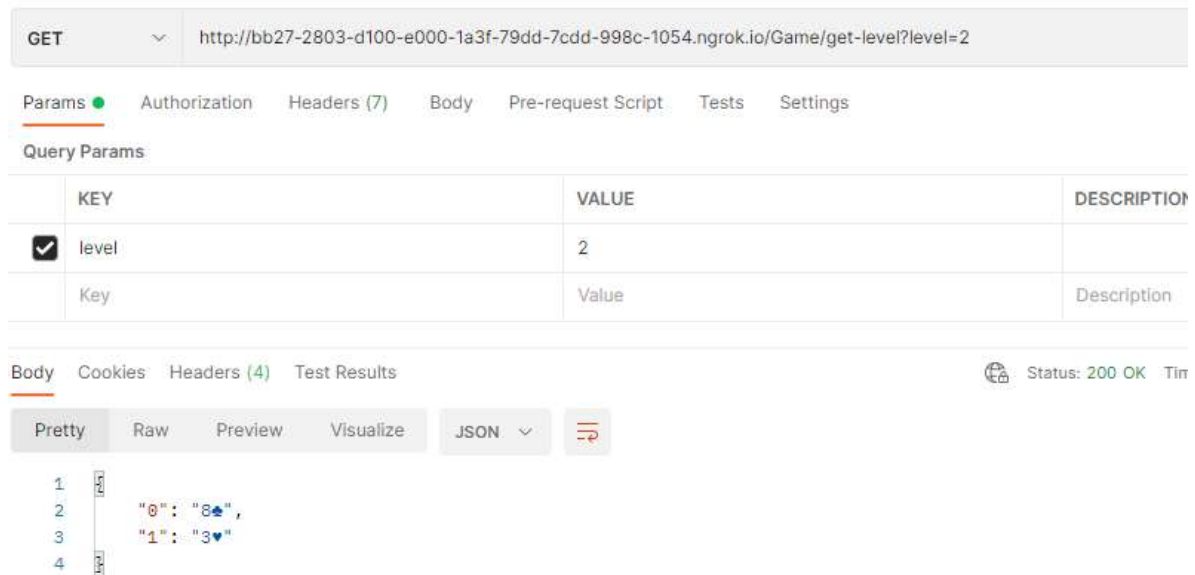


De esta forma se observa que la eliminación fue exitosa.

## Obtener nivel

Para obtener los nodos que se encuentran en el nivel solicitado Se envía una solicitud GET a la ruta <http://bb27-2803-d100-e000-1a3f-79dd-7cdd-998c-1054.ngrok.io/Game/get-level?level={nivel}> dónde se obtendrá un json con la información solicitada.

En este caso se desea el nivel 2, por lo cual se ingresa el nivel a obtener como 2.



The screenshot shows a Postman interface for a GET request to `http://bb27-2803-d100-e000-1a3f-79dd-7cdd-998c-1054.ngrok.io/Game/get-level?level=2`. The 'Query Params' section shows a parameter 'level' with value '2'. The 'Body' tab is selected, showing a JSON response in 'Pretty' format:

```
{
  "0": "8♣",
  "1": "8♣",
  "2": "3♥"
}
```

Con ello se obtienen los nodos que se encuentran en el nivel solicitado.

## Obtener recorrido del árbol

Para obtener el recorrido solicitado, se envía una solicitud de tipo Get la cual retorna un json con el recorrido especificado, esto se realiza a travez de la siguiente url:

<https://bb27-2803-d100-e000-1a3f-79dd-7cdd-998c-1054.ngrok.io/Game/avltree?transversal={recorrido}>

Es posible obtener el recorrido preOrden, postOrden e inOrden, para ello se debe de cambiar el valor recorrido por el valor deseado.

## PreOrden

Para obtener el recorrido preOrden del arbol se ingresa la siguiente url: <https://bb27-2803-d100-e000-1a3f-79dd-7cdd-998c-1054.ngrok.io/Game/avltree?transversal=preOrder>

Al ingresar la url por medio de postman se obtiene el siguiente resultado en formato json:

```
{
  "0": "10♣",
  "1": "8♣",
  "2": "3♥",
}
```

```
}
  "3": "5♠"
}
```

GET <https://bb27-2803-d100-e000-1a3f-79dd-7cdd-998c-1054.ngrok.io/Game/avltree?transversal=preOrder>

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	transversal	preOrder	
	Key	Value	Description

Body Cookies Headers (4) Test Results Status: 200 OK Time: 175 ms

Pretty Raw Preview Visualize JSON

```
1
2  "0": "10♣",
3  "1": "8♣",
4  "2": "3♥",
5  "3": "5♠"
6
```

## PostOrden

Para obtener el recorrido postOrden del arbol se ingresa la siguiente url: <https://bb27-2803-d100-e000-1a3f-79dd-7cdd-998c-1054.ngrok.io/Game/avltree?transversal=postOrder>

Al ingresar la url por medio de postman se obtiene el siguiente resultado en formato json:

```
{
  "0": "8♣",
  "1": "5♠",
  "2": "3♥",
  "3": "10♣"
}
```

GET <https://bb27-2803-d100-e000-1a3f-79dd-7cdd-998c-1054.ngrok.io/Game/avltree?transversal=postOrder>

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	transversal	postOrder	
	Key	Value	Description

Body Cookies Headers (4) Test Results Status: 200 OK Time: 176 ms

Pretty Raw Preview Visualize JSON

```
1
2  "0": "8♣",
3  "1": "5♠",
4  "2": "3♥",
5  "3": "10♣"
6
```

## inOrden

Para obtener el recorrido inOrden del arbol se ingresa la siguiente url: <https://bb27-2803-d100-e000-1a3f-79dd-7cdd-998c-1054.ngrok.io/Game/avltree?transversal=inOrder>

Al ingresar la url por medio de postman se obtiene el siguiente resultado en formato json:

```
{
  "0": "8♣",
  "1": "10♣",
  "2": "3♥",
  "3": "5♠"
}
```

The screenshot shows a Postman interface for a GET request to the URL `https://bb27-2803-d100-e000-1a3f-79dd-7cdd-998c-1054.ngrok.io/Game/avltree?transversal=inOrder`. The 'Query Params' section shows a table with one parameter: 'transversal' with the value 'inOrder'. The 'Body' tab is selected, showing the JSON response in 'Pretty' format. The status is '200 OK'.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> transversal	inOrder	
Key	Value	Description

Body	Cookies	Headers (4)	Test Results
Status: 200 OK			

JSON response (Pretty):

```
1 {
2   "0": "8♣",
3   "1": "10♣",
4   "2": "3♥",
5   "3": "5♠"
6 }
```