



# USAC

## TRICENTENARIA

Universidad de San Carlos de Guatemala

# INGENIERIA

# CUNOC

Laboratorio de Estructura de Datos

### **Proyecto 2: Documentación técnica**

Sección: A

Nombre:

Registro académico:

Mariano Francisco Camposeco Camposeco

202030987

Quetzaltenango, 09 de mayo de 2022.

## **Introducción**

En esta documentación encontrará que funcionalidad tiene alguna clase, método y controlador creado, para asimilar del cómo puede llegar a funcionar determinada instrucción, siendo entre lenguaje de alto y bajo nivel, con el fin de que no sea ni complejo ni tan sencillo que deje huecos en las explicaciones, así mismo contando con la lógica del funcionamiento del programa, detallando el funcionamiento de la API.

## **OBJETIVOS**

### **Objetivo general:**

Explicar las diferentes funciones realizadas en el código.

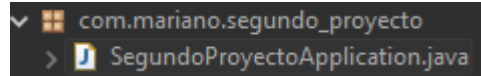
### **Objetivos específicos:**

1. Entender el qué funcionalidad se tiene en la API.
2. Aprender para qué se realizó un método.
3. Asimilar la lógica en el programa.

## Lógica de la API Juego de Cartas Pirámide

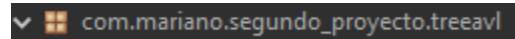
Esta API se basará específicamente en el juego de cartas solitario pirámide el cuál consiste en seleccionar dos cartas, que sumen 13 para reducir el monto que se tiene, o también se podría enviar una carta que equivalga 13, con eso ya funcionaría correctamente.

### Paquete segundo\_proyecto



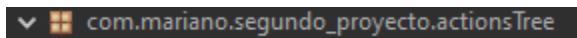
Este únicamente contiene la clase principal, con la cual inicia a correr la API.

### Paquete de treeavl

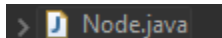


Este paquete contiene las clases esenciales para crear un arbolavl, sus nodos y el árbol en si.

### Paquete actionsTree



Este paquete cuenta con las cosas esenciales para un árbol avl, eliminar, ingresar, obtener nivel, y los recorridos.



```
//creamos constructor
```

```
public Node(String card, String cardType, int data)
```

```
//metodos get y set, para llamado y modificación de datos
```

```
public int getData()
```

```
public void setData(int data)
```

```
public int getBalance()
```

```
public void setBalance(int balance)
```


```
public String getCard()
```

```
public void setCard(String card)
```

```

public String getCardType()
public void setCardType(String cardType)
public Node getChildRight()
public void setChildRight(Node childRight)
public Node getChildLeft()
public void setChildLeft(Node childLeft)
//creamos una clase con la cual podremos crear nuestra grafica
public String createUserGraphic()

```

```
>  TreeAvl.java
```

```

//creamos arbol y raiz
public TreeAvl()
//Metodo para buscar si un nodo se encuentra ya establecido
public Boolean searchInTree(int data,Node root)
//Realizamos un metodo para lograr obtener el fe(Balance)
private int getBalanceData(Node node)
//realizar una rotacion para la izquierda
private Node rotationLeft(Node nodeRotation)
//realizar una rotacion para la derecha
private Node rotacionDerecha(Node nodeRotation)
//realizar una rotacion doble para la izquierda
private Node rotationDoubleLeft(Node nodeRotation)
//realizar una rotacion doble para la derecha
private Node rotationDoubleRight(Node nodeRotation)
//Agregamos datos a nuestro arbol avl
private Node insertTreeAvl(Node avlNewNode, Node avlSubTree)
//agregar dato a arbol
public void insertData(String card, String cardType, int data)
/*realizar recorridos
recorrido pre orden*/
public String routePreOrden(Node root)

```

```
//recorrido in orden
public String routeInOrder(Node root)

//recorrido post orden
public String routePostOrder(Node root)

//obtenemos datos de nivel solicitado
public String getLevel(Node root, int levelUser)

//obtenemos altura
public int getHeight(Node root)

//limpeamos nuestros string de recorridos
public void cleanAll()

//limpeamos el string de nivel
public void cleanLevel()

//regresamos errores
public int getErrores()

//creamos un metodo para la eliminacion dentro del nodo
private Node deleteTreeAvl(Node avlNewNode, Node avlSubTree)

//Metodo para eliminar dentro del arbol creado
public void deleteData(String card, String cardType, int data)

//escritura de archivo Graphviz
public String realizeGraphicGraphviz()

//creamos archivo Dot
private void getFileDot(String getRouteFile, String data)

//Creamos imagen jpg
public void createImageJpg()

//metodos get y set
public int getDuplicateCard()
public void setDuplicateCard(int duplicateCard)
public int getStatusCode404()
public void setStatusCode404(int statusCode404)
public int getStatusCode409()
```

```
public void setStatus409(int statusCode409)
```

```
> Delete.java
```

```
//obtenemos los valores que tendría una carta
```

```
public int getCardValue(String cardType, String card)
```

```
//agregamos un dato a la carta con un json y un arbol,esto luego de haber iniciado
```

```
public TreeAvl deleteCard(String dataJson, TreeAvl treeUser)
```

```
//obtenemos el tipo de carta
```

```
public String getCardType(String treeCard)
```

```
//establecemos el dato principal de la cara, la letra
```

```
public String getPrincipalCard(String treeCard)
```

```
//metodos get y set, para llamado y modificación de datos
```

```
public int getErrorCardDelete()
```

```
public void setErrorCardDelete()
```

```
public int getStatus406()
```

```
public void setStatus406(int status406)
```

```
> Enter.java
```

```
//obtenemos los valores que tendría una carta
```

```
public int getCardValue(String cardType, String card)
```

```
//verificamos que contenga cierto caracter para darle corrimiento
```

```
switch (cardType)
```

```
//agregamos carta con un json, esto al inicio
```

```
public TreeAvl addCardStart(String dataJson)
```

```
//agregamos un dato a la carta con un json y un arbol,esto luego de haber iniciado
```

```
public TreeAvl addCardAfterStart(String dataJson, TreeAvl treeUser)
```

```
//obtenemos el tipo de carta
```

```
public String getCardType(String treeCard)
```

```
//establecemos el dato principal de la cara, la letra
```

```
public String getPrincipalCard(String treeCard)
```

```
//metodos get y set, para llamado y modificación de datos
```

```
public int getErrorCardEnter()
public void setErrorCardEnter()
```

```
> Level.java
```

```
//creamos un map para obtener los nodos de un nivel determinado
public Map<String, String> getLevel(TreeAvl arbol,int levelUser)
```

```
> Route.java
```

```
//creamos un map para obtener la ruta en preOrden
public Map<String, String> routePreOrder(TreeAvl arbol)
//creamos un map para obtener la ruta en inOrden
public Map<String, String> routeInOrder(TreeAvl arbol)
//creamos un map para obtener la ruta en postOrden
public Map<String, String> routePostOrder(TreeAvl arbol)
```

## Paquete controllers

```
com.mariano.segundo_proyecto.controllers
> SegundoProyectoControlador.java
```

Este paquete cuenta con la clase que establece los métodos POST, DELETE, GET de la API en funcionamiento.

```
//creamos un arbol, metodo insertar, metodo eliminar
TreeAvl treeAvlUser;
Enter addElementTree = new Enter();
Delete deleteElementTree = new Delete();
/*Metodo POST
 * Recibimiento de un arbol inicial
 * */
@RequestMapping(value = "/Game/start", method = RequestMethod.POST, consumes =
"application/json")
public ResponseEntity<String> insertarStart(@RequestBody String node)
/*Metodo POST
 * Recibimiento para adiccion de carta
 * */
```



```
@RequestMapping(value = "/Game/add", method = RequestMethod.POST, consumes = "application/json")
```

```
public ResponseEntity<String> insertar(@RequestBody String node)
```

```
/*Metodo GET
```

```
 * Enviado a usuario los datos en preOrden,postOrden e inOrden
```

```
 * */
```

```
@GetMapping(value = "/Game/avltree", produces = "application/json")
```

```
public ResponseEntity<String> recorrido(@RequestParam(name = "transversal") String tipo)
```

```
/*Metodo GET
```

```
 * Procesamiento de imagen y enviado de datos al usuario respecto a donde observar dicha imagen
```

```
 * */
```

```
@GetMapping(value = "/Game/status-avltree", produces = "application/json")
```

```
public ResponseEntity<String> graficar(HttpServletRequest solicitudServidor)
```

```
/*Metodo GET
```

```
 * Devolucion de imagen
```

```
 * */
```

```
@GetMapping(value = "/Game/status-avltree/state-tree", produces = MediaType.IMAGE_JPEG_VALUE)
```

```
public ResponseEntity<Resource> stateTree() throws IOException
```

```
/*Metodo GET
```

```
 * Enviado a usuario los datos de los nodos que se encuentran en un determinado nivel
```

```
 * */
```

```
@GetMapping(value = "/Game/get-level", produces = "application/json")
```

```
public ResponseEntity<String> getLevel(@RequestParam(name = "level") int levelUser)
```

```
/*Metodo DELETE
```

```
 * Recibimos una peticion para poder eliminar un nodo
```

```
 * */
```

```
@DeleteMapping(value = "/Game/delete", consumes = "application/json", produces = "application/json")
```

```
public ResponseEntity<String> deleteCard(@RequestBody String nodes)
```