ALGORITMOS Y ESTRUCUTRA DE DATOS 2

**Documentación - Obligatorio**



Grupo N4A

# Docente:

* Francisco Bouza

# Integrantes:

* Sebastián Bonacci (214587)



* Vicente Bermúdez (214831)



Índice

[ESTRUCTURA DE DATOS Y ALGORITMOS 1](file:///C:\Users\vincentes\Downloads\Documentación%20-%20Template.docx#_Toc359097655)

[Obligatorio 1](file:///C:\Users\vincentes\Downloads\Documentación%20-%20Template.docx#_Toc359097656)

[Interfaz Sistema: Pre y post condiciones 3](file:///C:\Users\vincentes\Downloads\Documentación%20-%20Template.docx#_Toc359097657)

[Solución escogida 3](file:///C:\Users\vincentes\Downloads\Documentación%20-%20Template.docx#_Toc359097658)

[Diagrama de la estructura de datos 3](file:///C:\Users\vincentes\Downloads\Documentación%20-%20Template.docx#_Toc359097659)

[Justificación 3](file:///C:\Users\vincentes\Downloads\Documentación%20-%20Template.docx#_Toc359097660)

[Testing 3](file:///C:\Users\vincentes\Downloads\Documentación%20-%20Template.docx#_Toc359097661)

# Interfaz Sistema: Pre y post condiciones

// PRE: Recibe una deseada cantidad de puntos (cantPuntos) para el sistema

// POST: Inicializa y resetea estructuras, asignando como tope a cantPuntos

**public** Retorno inicializarSistema (**int** cantPuntos);

// PRE: -

// POST: Pone en null a todas las estructuras del sistema, libera memoria.

**public** Retorno destruirSistema();

// PRE: Recibe info del productor a registrar: una cedula válida, un nombre, una direccion, un email válido, y un celular válido.

// POST: Registra al productor en el sistema

**public** Retorno registrarProductor(String cedula, String nombre, String direccion, String email, String celular);

// PRE: Recibe info de la ciudad a registrar: nombre, y coordenadas no tomadas

// POST: Registra a la ciudad en el sistema

**public** Retorno registrarCiudad(String nombre, Double coordX, Double coordY);

// PRE: Recibe info de la plantacion a registrar: nombre, coordenadas no tomadas, una cédula de un productor registrado, y una capacidad mayor a 0.

// POST: Registra a la plantacion en el sistema

**public** Retorno registrarPlantacion(String nombre, Double coordX, Double coordY, String cedula\_productor, **int** capacidad);

// PRE: Recibe info del silo a registrar: nombre, coordenadas no alojadas, y una capacidad mayor a 0

// POST: Registra el silo en el sistema

**public** Retorno registrarSilo(String nombre, Double coordX, Double coordY, **int** capacidad);

// PRE: Recibe dos pares de coordenadas alojadas en el mapa, y un peso del tramo a registrar

// POST: Registra el tramo en el sistema

**public** Retorno registrarTramo(Double coordXi, Double coordYi, Double coordXf, Double coordYf, **int** peso);

// PRE: Recibe dos pares de coordenadas como identificadores del tramo a eliminar

// POST: Elimina el tramo del sistema

**public** Retorno eliminarTramo(Double coordXi, Double coordYi, Double coordXf, Double coordYf);

// PRE: Recibe un par de coordenadas como identificador del punto a eliminar

// POST: Elimina el punto del sistema

**public** Retorno eliminarPunto(Double coordX, Double coordY);

// PRE: -

// POST: Muestra en el navegador un mapa con los puntos marcados con colores identificadores

**public** Retorno mapaEstado();

// PRE: Recibe el par de coordenadas de un silo

// POST: Retorna el recorrido mas corto a tomar para llegar al silo mas cercano para satisfacer al productor localizado en (coordX, coordY)

**public** Retorno rutaASiloMasCercano(Double coordX, Double coordY);

// PRE: Recibe el par de coordenadas de una ciudad

// POST: Retorna una lista de las plantaciones que sean alcanzables recorriendo 20 kms por tramos definidos en el mapa

**public** Retorno listadoDePlantacionesEnCiudad(Double coordX, Double coordY);

// PRE: -

// POST: Retorna la lista de todos los silos registrados en el sistema con su capacidad original y su capacidad remanente

**public** Retorno listadoDeSilos();

// PRE: -

// POST: Retorna la lista de todos los productores ordenados por cédula en manera ascendente, en forma de String

# public Retorno listadoProductores();Solución escogida

## Diagrama de la estructura de datos

## 

## Justificación

Justificación de las estructuras elegidas para modelar las entidades del problema.

|  |  |
| --- | --- |
| **Productor.java** | |
| Estructura | Arbol Binario de Búsqueda |
| Justificación | Debido a la probable gran cantidad de productores que se necesitarán tener registrados y mantenidos en el sistema, se decidió utilizar la nombrada estructura. Los productores son buscados cada vez que se crea una plantación; con esta estructura fomentamos la eficiencia de búsqueda en el sistema. |

|  |  |
| --- | --- |
| **Grafo.java (Sistema.mapa)** | |
| Estructura | Grafo |
| Justificación | Los requerimientos del sistema nos obligan a utilizar el grafo como estructura de mapeo. Identificamos las ciudades, los silos, y las plantaciones mediante puntos en un plano interconectado. No hay otra con más coherencia, eficiencia, y sencillez que ésta. Por lo tanto, se nos hace monumental incorporar este diseño. |

## Testing

Resumen de las pruebas realizadas y los resultados obtenidos.