# PROYECTO 1

ESTRUCTURAS DE DATOS
VACACIONES DE JUNIO 2022
FACULTAD DE INGENIERIA, ESCUELA DE CIENCIAS Y SISTEMAS
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

## Manual Técnico


**Realizado por**

Rodrigo Hernández

**Carnet**

201900042


# ÍNDICE

# I. Introducción

La aplicación web de la librería CatsBooks fue realizado en el lenguaje de programación JavaScript junto con el apoyo de los lenguajes HTML y CSS donde la estructura visual fue apoyada por medio del uso de una plantilla donde se adaptó a los requerimientos del programa y de las estructuras de datos implementadas.

# II. Objetivo

El objetivo primordial del presente manual es orientar al programador acerca de la explicación de la forma en que fue creada la aplicación desde el código fuente, la ubicación de la aplicación de las estructuras de datos y la importante utilización de la Programación Orientada a Objetos en JavaScript para poder crear los distintos TDA para la solución de los requerimientos del sistema.

# III. Dirigido

Este manual va orientado a los distintos programadores interesados en el conocimiento de las estructuras de datos dentro de la programación web y la implementación del mismo con herramientas como D3 para mostrar las estructuras creadas en el lenguaje de programación.

# IV. Especificación técnica

## 1. Requisitos de Hardware

- Computadora de escritorio o portátil.
- Mínimo 4GB de Memoria RAM.
- Procesador Intel Core i3 o superior.
- Resolución gráfica máxima de 1900 x 1070 píxeles.

## 2. Requisitos de Software

- Sistema Operativo con que pueda contar con un navegador web y un Editor de Texto.
- Visual Studio Code u otro editor de texto.
- Librería D3.
- Graphviz.
- Navegador Web para el manejo de la aplicación web.

# V. Lógica de la Aplicación

## 1. Clases

### a. Clase Usuario

En esta clase será utilizada para almacenar objetos de tipo Usuario y poder interactuar con las estructuras de datos.

```
class Usuario{
    constructor(_dpi,_nombre,_username,_correo,_rol,_contrasenia,_telefono){
        this.dpi = _dpi;
        this.nombre = _nombre
        this.username = _username
        this.correo = _correo
        this.rol = _rol
        this.contrasenia = _contrasenia
        this.telefono = _telefono
    }
}
```

### b. Clase Libro

En esta clase será utilizada para almacenar objetos de tipo Libro y poder interactuar con las estructuras de datos.

```
class Libro{

 constructor(_isbn,_autor,_nombre,_cantidad,_fila,_columna,_paginas,_categoria){
        this.isbn = _isbn
        this.autor = _autor
        this.nombre = _nombre
        this.cantidad = _cantidad
        this.fila = _fila
        this.columna = _columna
        this.paginas = _paginas
        this.categoria = _categoria
    }
}
```

### c. Clase Autor

En esta clase será utilizada para almacenar objetos de tipo Autor y poder interactuar con las estructuras de datos.

```
class Autor{
    constructor(_dpi,_nombre,_correo,_telefono,_direccion,_biografia){
        this.dpi = _dpi
        this.nombre = _nombre
        this.correo = _correo
        this.telefono = _telefono
        this.direccion = _direccion
        this.biografia = _biografia
    }
}
```

# 2. Estructuras de Datos

## a. Lista de Listas

Esta estructura es una lista simple circular donde también cada nodo tendrá una lista simple de libros y está hecha para almacenar objetos de tipo Usuario y objetos de tipo Libro donde se necesitaron 2 nodos:

### i. Nodo CU

Este nodo se encarga de almacenar un usuario y almacenara un siguiente y un abajo.

```
class NodoCU{
    constructor(_usuario){
        this.usuario = _usuario;
        this.siguiente = null;
        this.abajo = null;
    }
}
```

### ii. Nodo Libro U

Este nodo se encarga de almacenar únicamente los nombres de los libros y tendrá un apuntador hacia abajo.

```
class NodoLibroU{
    constructor(_nombre){
        this.nombre = _nombre
        this.abajo = null
    }
}
```

### iii. Lista Usuarios

En esta clase contendrá todas las operaciones con la lista de Usuarios.

```
class ListaUsuarios{
    constructor(){
        this.primero = null;
        this.ultimo = null;
        this.tamanio = 0;
    }
    insertarusuario(_cliente){
        //Inserta al usuario a la lista circular
    }
    retornarusuario(_dpi){
        //Retorna el usuario por medio de su DPI
    }
    insertarlibro(_libro, _dpi){
        //Inserta el nombre del libro en la lista de listas al usuario
    }
    mostrarusuarios(){
        //Muestra en consola todos los usuarios
    }
    graficar(){
        //Grafica la lista de listas utilizando el codigo dot
    }
    verificarusuario(username, contrasenia){
        //Verifica si el usuario es Administrador o Usuario o ninguno
    }
    validarususuario(usuario,contrasenia){
        //Valida si existe el usuario
    }
    retornarusuariologin(username,contrasenia){
        //Retorna el objeto usuario por medio del username y contraseña
    }
    retornarCantidadLibros(_dpi){
        //Retorna la cantidad de libros por cada usuario
    }
    retornarlibro(_dpi,_libro){
        //Retorna los libros del usaurio
    }
    retornarcantlibro(_dpi,_libro){
        //Retorna la cantidad de ese libro comprado por ese usuario
    }
}
```

## b. Lista Doblemente Enlazada

Esta estructura se implementa para poder generar el top 5 de usuarios con mas compras de libros donde únicamente se almacena la cantidad de libros por usuario y los datos del usuario.

### i. Nodo DU

Este nodo se encarga de almacenar los objetos de tipo Usuario y la cantidad de libros comprados por ese usuario.

```
class NodoDU{
    constructor(_usuario, _cantidad){
        this.usuario = _usuario
        this.cantidad = _cantidad
        this.siguiente = null
        this.anterior = null
    }
}
```

### ii. Lista Doble

En esta clase se hacen las diferentes operaciones que requiere la lista doblemente enlazada para poder cumplir con la operación del top 5 de usuarios.

```
class ListaDoble{
    constructor(){
        this.primero = null
        this.ultimo = null
        this.tamanio = 0
    }
    insertar(_usuario, _cantidad){
        //Inserta a los usuarios y su cantidad de libros al nodo y a la lista doble
    }
    ordenar(){
        //Ordena a los Usuarios de manera Descendente utilizando el Ordenamiento Burbuja
    }
    graficar(){
        //Grafica la lista doblemente enlazada hasat 5 nodos
    }
}
```

## c. Matriz Dispersa

En esta estructura se enfoca en almacenar objetos de tipo libro y específicamente que sea de categoría Thriller.

### i. Nodo Cabecera MD

En esta clase se encarga para las cabeceras de la matriz dispersa y almacenar el id(número de posición) de ese nodo. Y contiene 3 apuntadores que son: anterior, siguiente y acceso.

```
class NodoCabeceraMD{
    constructor(id){
        this.id = id
        this.siguiente = null
        this.anterior = null
        this.acceso = null
    }
}
```

### ii. Lista Cabecera MD

Esta clase se encarga de manejar las cabeceras de la matriz dispersa utilizando la lista doblemente enlazada.

```
class ListaCabeceraMD{
    constructor(tipo){
        this.tipo = tipo
        this.primero = null
        this.ultimo = null
        this.tamanio = 0
    }

    insertar_nodoCabecera(nuevo){
      /*
        Inserta el nodo a la cabecera del nodo y verifica si:
        -- Es Nulo
        -- El id es menor al primer nodo
        -- El id es mayor al primer nodo y menor al ultimo nodo
        -- El id es mayor al ultimo nodo
      */
    }
    obtenercabecera(id){
        //Retorna el nodo cabecera por medio de su id
    }
}
```

### iii. Nodo Celda

En este nodo se almacenan los libros, sus coordenadas en X y Y, y tiene 4 apuntadores: arriba, abajo, izquierda y derecha.

```
class NodoCelda{
    constructor(x,y,libro){
        this.x = x
        this.y = y
        this.libro = libro
        this.arriba = null
        this.abajo = null
        this.izquierda = null
        this.derecha = null
    }
}
```

### iv. Matriz Dispersa

En esta clase es necesario tener las clases de arriba para poder utilizarla y donde se hacen los procedimientos para insertar los libros a los nodos de la matriz.

```
class MatrizDispersa{
    constructor(){
        this.filas = new ListaCabeceraMD('fila')
        this.columnas = new ListaCabeceraMD('columna')
    }
    insertar(coor_x,coor_y,libro){
        /*
        Inserta los nodos manteniendo los siguientes casos:
        --Si la fila es null
        --Si la columna es null
        --Si existe la columna pero no la fila
        --Si existe la fila pero no la columna
        */
    }
    retornarNodo(fila,columna){
        //Retorna el nodo por medio de sus parametros de la ubicacion de la fila y
          columna
    }

    graficar(){
        //Grafica la estructura por medio del codigo dot.
    }
    graficarlibrera(){
        //Grafica por medio de ayuda del codigo dot para adaptarlo a una librera.
    }
}
```

## d. Matriz Ortogonal (Lista Ortogonal)

En esta estructura se almacenan objetos de tipo Libro y específicamente que sean de categoría Fantasía.

### i. Nodo CO

En esta clase se encarga de guardar los libros, sus coordenadas en X y Y y contiene dos apuntadores que son hacia la derecha(siguiente) y hacia abajo.

```
class NodoCO{
    constructor(_libro,_x,_y){
        this.libro = _libro
        this.x = _x
        this.y = _y
        this.abajo = null
        this.siguiente = null
    }
}
```

### ii. Lista MO

En esta clase se encarga de crear las columnas de la matriz, específicamente de 25 columnas.

```
class ListaMO{
    constructor(){
        this.raiz = null
        this.ultimo = null
    }
    insertarlista(_libro,_y){
        //Se insertan los nodos que pueden ser de tipo libro o de tipo null ya que es
          necesario para rellenar 25 espacios asi como cumple la definicion de una matriz
          ortogonal.
    }

    retornarlista(_y){
        //Retorna toda la columna de la matriz ortogonal.
    }
}
```

### iii. Matriz Ortogonal

En esta clase se realizan las operaciones para la estructura de la matriz ortogonal.

```
class MatrizOrtogonal{
    constructor(){
        this.lista_x = new ListaMO()
    }

    llenarmatriz(){
        //Llena todos los nodos de la matriz con valores nulos
    }

    insertar(_coorx,_coory,_libro){
        //Inserta cada nodo haciendo la sustitucion de la casilla nula al objeto Libro
    }

    retornarlibro(_nombre){
        //Retorna el objeto Libro por medio del nombre
    }

    retornarnodolibro(_x,_y){
        //Retorna el nodo del libro por medio de sus coordenadas
    }

    graficar(){
         //Grafica con el uso del lenguaje dot la matriz como una estructura
    }
    graficarlibrera(){
         //Grafica la matriz con el uso del lenguaje dot la matriz adaptandolo a una
           librera.
    }
}
```

## e. Pila

Esta estructura esta hecha para poder mostrar la cantidad de ejemplares de cada libro.

### i. Nodo Pila

En este nodo se encarga de almacenar el numero y el apuntador hacia abajo.

```
class NodoPila{
    constructor(_numero){
        this.numero = _numero
        this.abajo = null
    }
}
```

### ii. Pila

Aquí se encuentran todos los métodos para operar la pila.

```
class Pila{
    constructor(){
        this.cima = null
        this.tamanio = 0
    }
    insertar(_numero){
        //Metodo para agregar los numeros a la pila aplicando LIFO(push)
    }
    sacar(){
        //Metodo para quitar los numeros de la pila aplicando LIFO(pop)
    }
    graficar(){
        //Grafica la pila de ejemplares con ayuda del lenguaje dot.
    }
}
```

## f. Cola

Esta estructura se encarga de almacenar todos los libros que ya no cuentan con disponibilidad a la hora de la compra, almacenando al usuario y la cantidad de ejemplares que ya no logro conseguir.

### i. Pendiente

Objeto que almacena al usuario y la cantidad de libros pendientes.

```
class Pendiente{
    constructor(_usuario,_libro, _cantidad){
        this.usuario = _usuario
        this.libro = _libro
        this.cantidad = _cantidad
    }
}
```

### ii. Nodo Cola

Objeto que almacena al objeto pendiente y su apuntador siguiente.

```
class NodoCola{
    constructor(_pendiente){
        this.pendiente = _pendiente
        this.siguiente = null
    }
}
```

### iii. Cola

Clase donde se realizan las distintas operaciones con la cola de disponibilidad.

```
class Cola{
    constructor(){
        this.primero = null
        this.tamanio = 0
    }
    insertar(_pendiente){
        //Inserta el nodo aplicando la metodologia FIFO(enqueue)
    }
    graficar(){
        //Grafica la estructura de la cola con ayuda del lenguaje dot.
    }
}
```

## g. Lista Simple

Esta estructura se encarga de almacenar todos los libros para poder hacer las distintas operaciones de ordenamiento.

### 1. Nodo Lib

En este nodo se almacena el objeto de tipo libro y el apuntador siguiente.

```
class NodoLib{
    constructor(_libro){
        this.libro = _libro
        this.siguiente = null
    }
}
```

### 2. Lista Simple

En esta clase se encuentra todas las operaciones con la lista simple junto con los ordenamientos de los libros.

```
class ListaSimple{
    constructor(){
        this.cabeza = null
        this.tamanio = 0
    }
    insertar(_libro){
        //Se encarga de insertar los nodos a la lista
    }
    ordenamientoascendente(){
        //Ordena la lista de forma ascendente(A-Z) utilizando el
        ordenamiento burbuja el cual muestra el codigo en la vista de
        libros
    }
    ordenamientodescendente(izquierda, derecha){
        //Ordena la lista de forma descendente(Z-A) utilizando el
        ordenamiento quicksort de manera recursiva el cual muestra el
        codigo en la vista de libros
    }
}
```

## h. Árbol Binario

Esta estructura se encarga de guardar a los autores ordenándolos e insertándolos por nombre.

### 1. Nodo Autor

Este nodo se encarga de almacenar el objeto de tipo Autor y cuenta con dos apuntadores que son izquierda y derecha.

```
class NodoAutor{
    constructor(_autor){
        this.autor = _autor
        this.izquierda = null
        this.derecha = null
    }
}
```

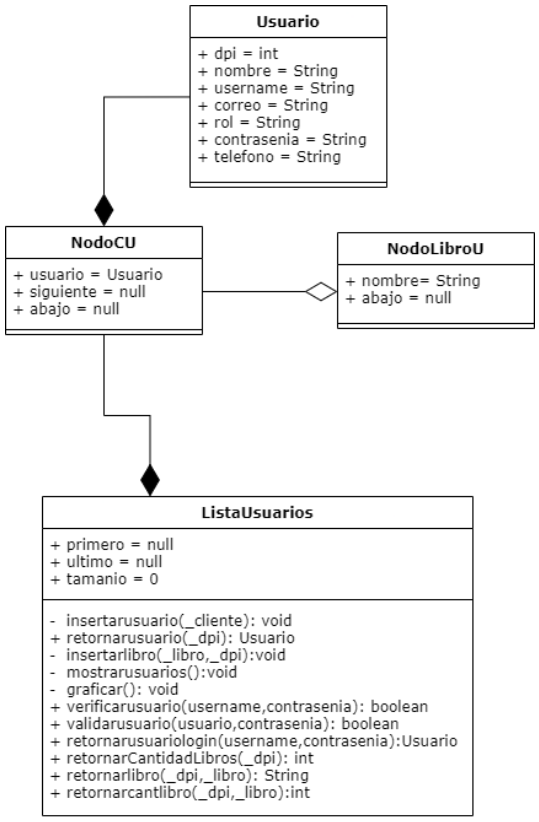
2. **Árbol Binario**

Esta clase se encarga de realizar todas las operaciones de la estructura del árbol binario.

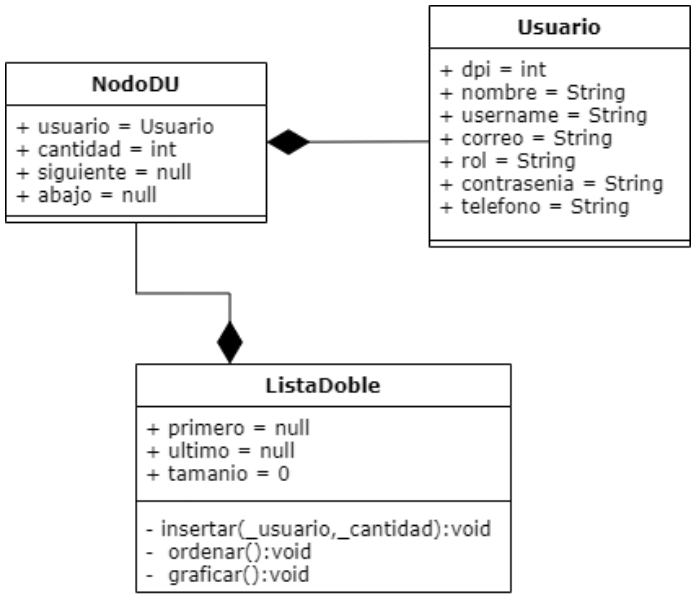
```
class ArbolBinario{
    constructor(){
        this.raiz = null
        this.codigodot = ""
    }
    insertarautor(_autor){
        //Inserta al Autor al árbol y llama a su metodo recursivo
    }

    add(_autor, nodo){
        //Funcion recursiva para añadir al autor en cada nodo desde su raiz
    }

    preordenD(){
        //Hace el llamado a su metodo recursivo para poder graficar el árbol
    }
    pre_ordenD(nodo){
        //Añade los nodos del codigo dot y los enlaza para almacenarlos en un
          string
    }

    graficar(){
        //Arma toda la estructura del codigo dot para la grafica del árbol
    }

    buscar(nodo,_nombre){
        //Metodo recursivo para buscar en cada nodo el autor teniendo como
        parametro el nombre
    }

}
```

# VI. Diagramas de Clase

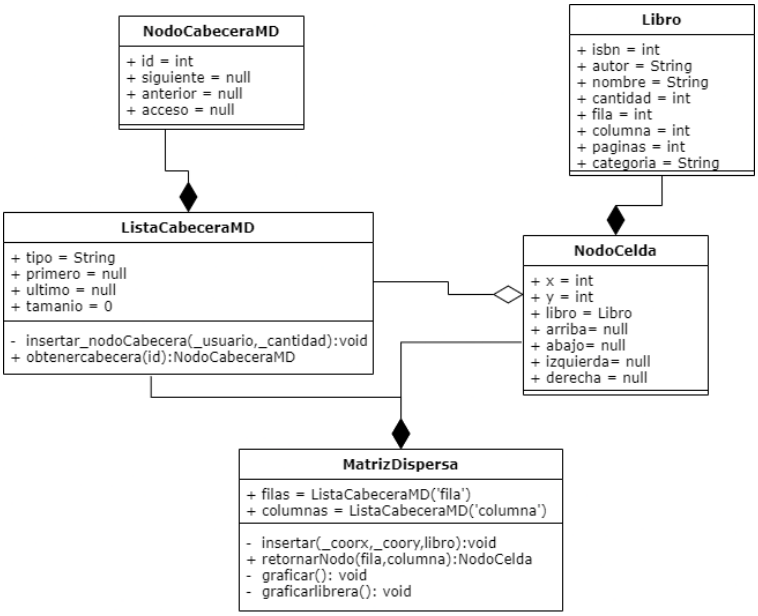
## 1. Lista de Listas



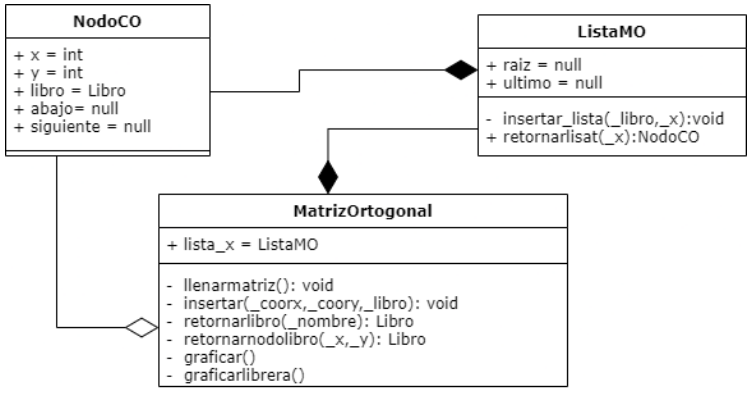

## 2. Lista Doblemente Enlazada



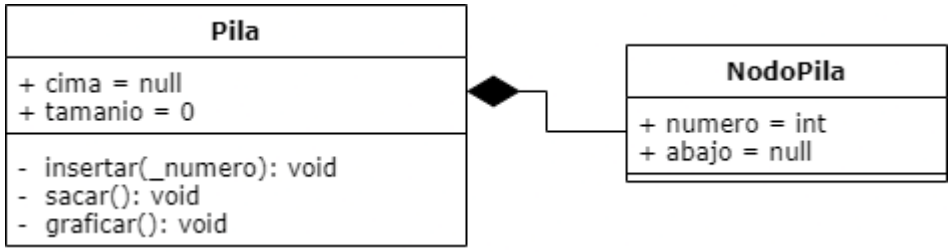
# 3. Matriz Dispersa



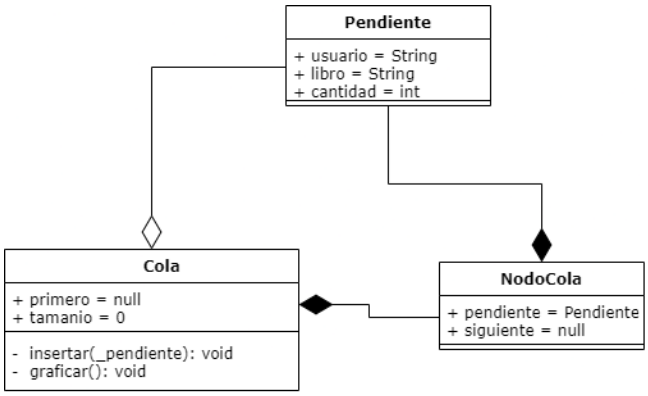

# 4. Matriz Ortogonal



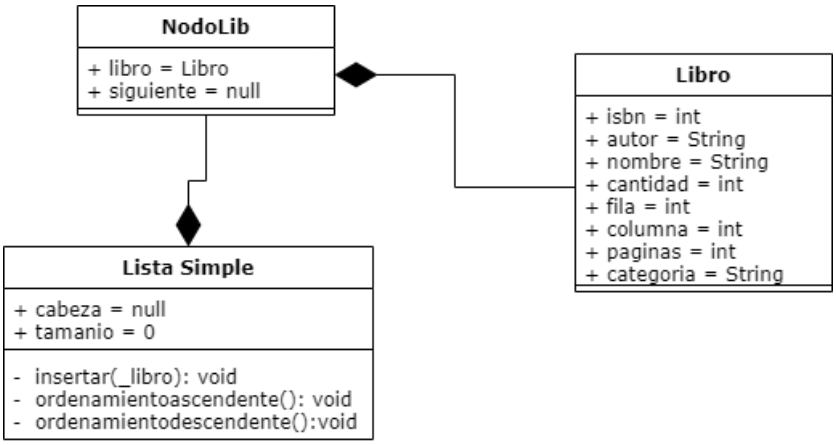

# 5. Pila



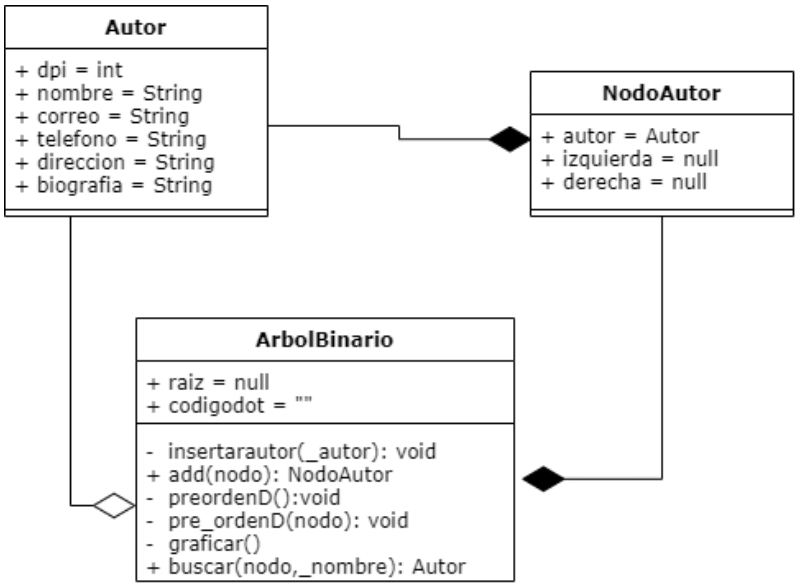
# 6. Cola




# 7. Lista Simple




# 8. Árbol Binario



# 9. Diagrama de todo el sistema