

Complejidad Temporal, Estructuras de Datos y Algoritmos

CARRANZA BRAIAN
HERRERA FRANCO

PRESENTACIÓN TRABAJO FINAL

UNIVERSIDAD NACIONAL ARTURO JAURETCHE
INGENIERÍA EN INFORMÁTICA

Índice

- 01** Diagrama UML
 - 02** Detalles de la Implementación
 - 03** Ideas y/o Sugerencias de Mejora
 - 04** Reflexiones
 - 05** Conclusion
-

Integrantes



Carranza

BRAIAN

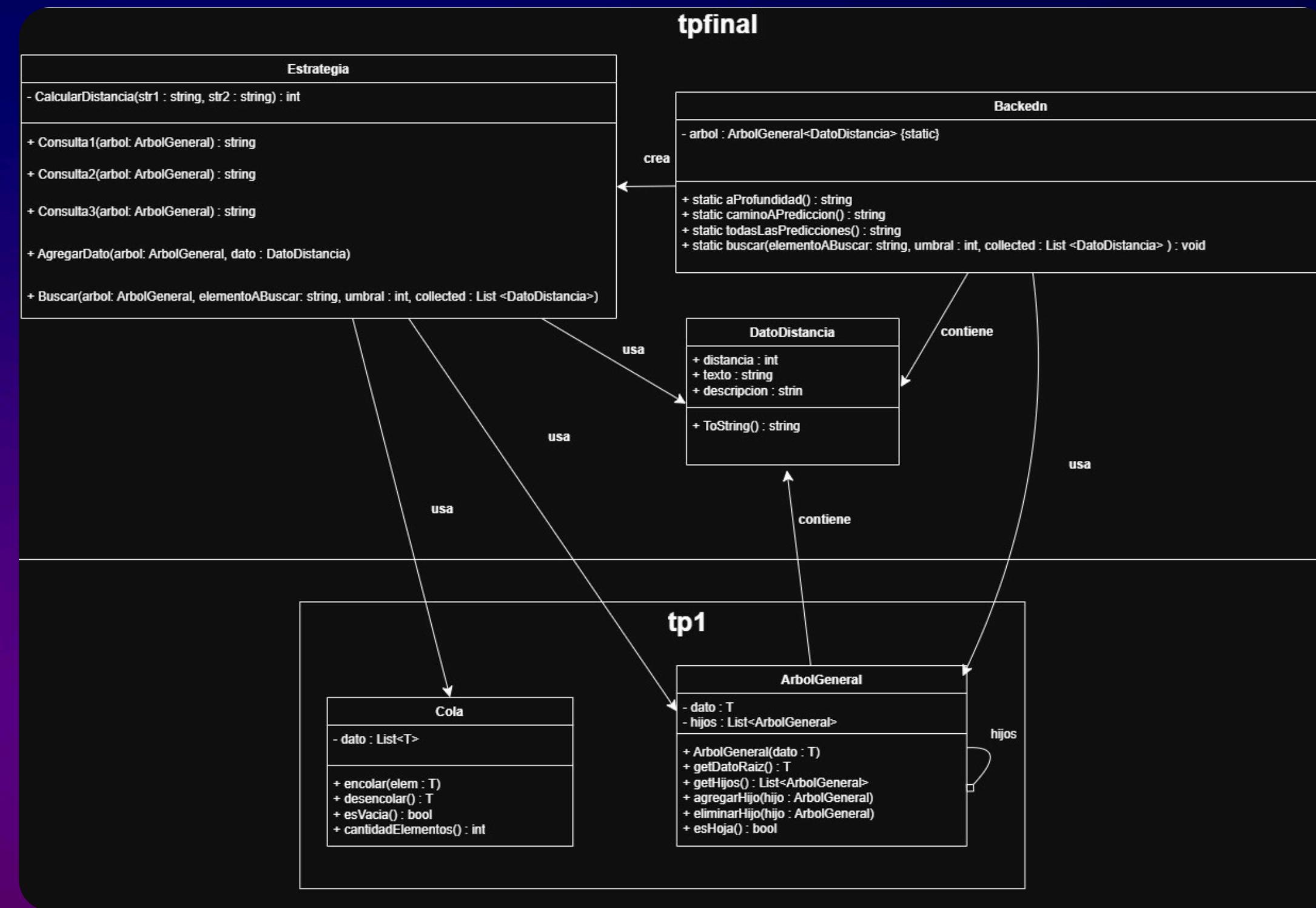


Herrera

FRANCO

Desarrollo de los métodos
AgregarDatos, Buscar y la Consulta1.
Creación del repositorio remoto,
inicialización del proyecto.

Desarrollo de la Consulta2 y la
Consulta3.
Creación de la presentación.
Edición del Video.



Detalles de la Implementación

- Resolución de “AgregarDatos”
- Resolución de “Buscar”
- Resolución de la “Consulta1”
- Resolución de la “Consulta2”
- Resolución de la “Consulta3”



AgregarDato

```
2 references
public void AgregarDato(ArbolGeneral<DatoDistancia> arbol, DatoDistancia dato)
{
    int distancia = CalcularDistancia(arbol.getDataRaiz().texto, dato.texto);

    ArbolGeneral<DatoDistancia> hijoExistente = null;

    foreach (var hijo in arbol.getHijos()) {
        if (CalcularDistancia(arbol.getDataRaiz().texto, hijo.getDataRaiz().texto) == distancia) {
            hijoExistente = hijo;
            break;
        }
    }

    if (hijoExistente == null) {
        arbol.agregarHijo(new ArbolGeneral<DatoDistancia>(dato));
    }
    else
    {
        AgregarDato(hijoExistente, dato);
    }
}
```

Buscar

```
2 references
public void Buscar(ArbolGeneral<DatoDistancia> arbol, string elementoABuscar, int umbral, List<DatoDistancia> collected)
{
    if (arbol == null)
    {
        return;
    }

    int distancia = CalcularDistancia(arbol.getDatoRaiz().texto, elementoABuscar);

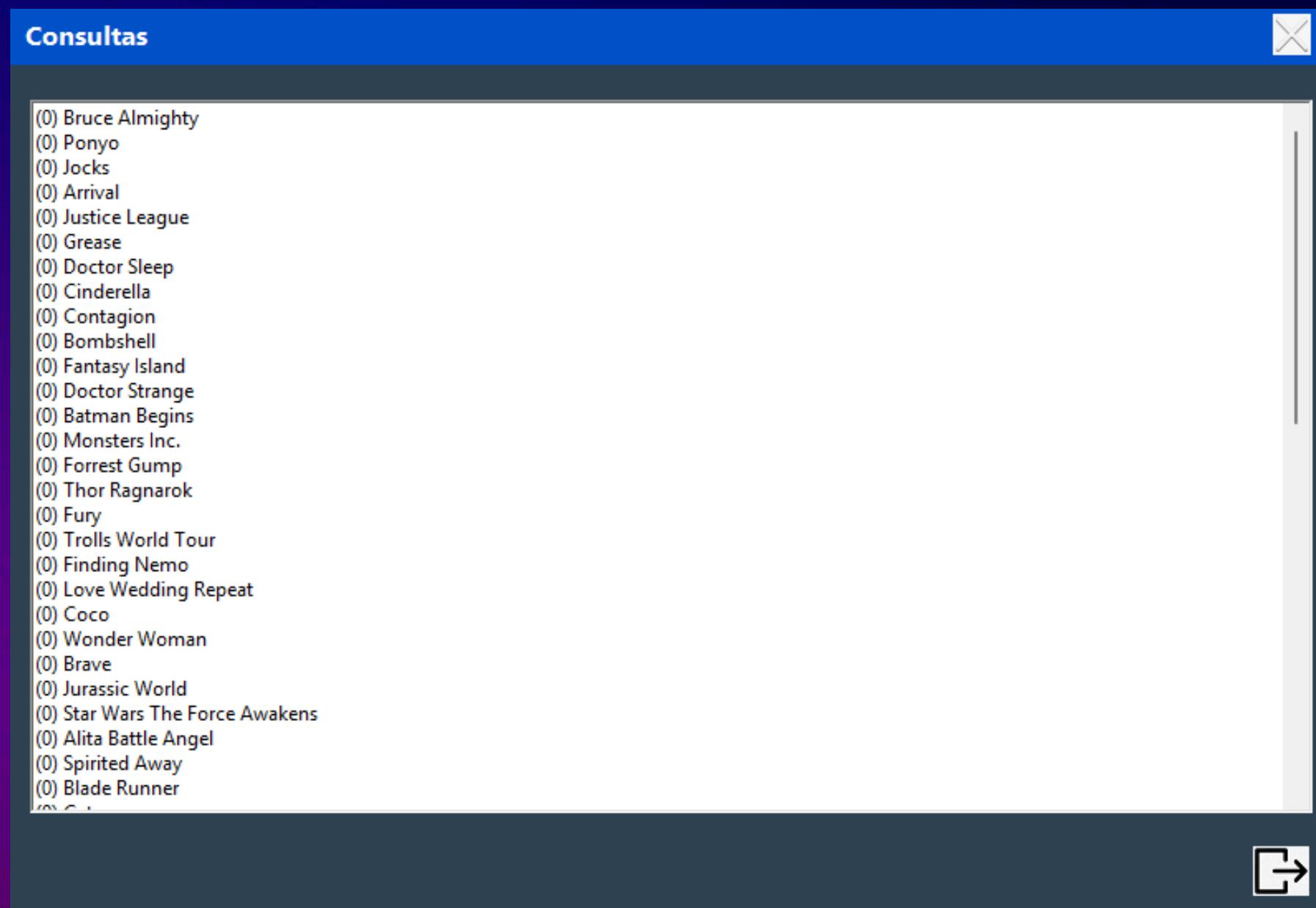
    if(distancia <= umbral)
    {
        collected.Add(arbol.getDatoRaiz());
    }

    foreach(var hijo in arbol.getHijos())
    {
        int distanciaHijo = CalcularDistancia(arbol.getDatoRaiz().texto, hijo.getDatoRaiz().texto);
        if(distanciaHijo >= distancia - umbral && distanciaHijo <= distancia + umbral)
        {
            Buscar(hijo, elementoABuscar, umbral, collected);
        }
    }
}
```

Consulta1

```
2 references
public String Consulta1(ArbolGeneral<DatoDistancia> arbol)
{
    string resultado;
    if (arbol.esHoja())
    {
        return arbol.getDataRaiz().ToString() + "\n";
    }
    else
    {
        resultado = "";
        foreach(ArbolGeneral<DatoDistancia> hijo in arbol.getHijos())
        {
            resultado = resultado + Consulta1(hijo);
        }
        return resultado;
    }
}
```

Resultados



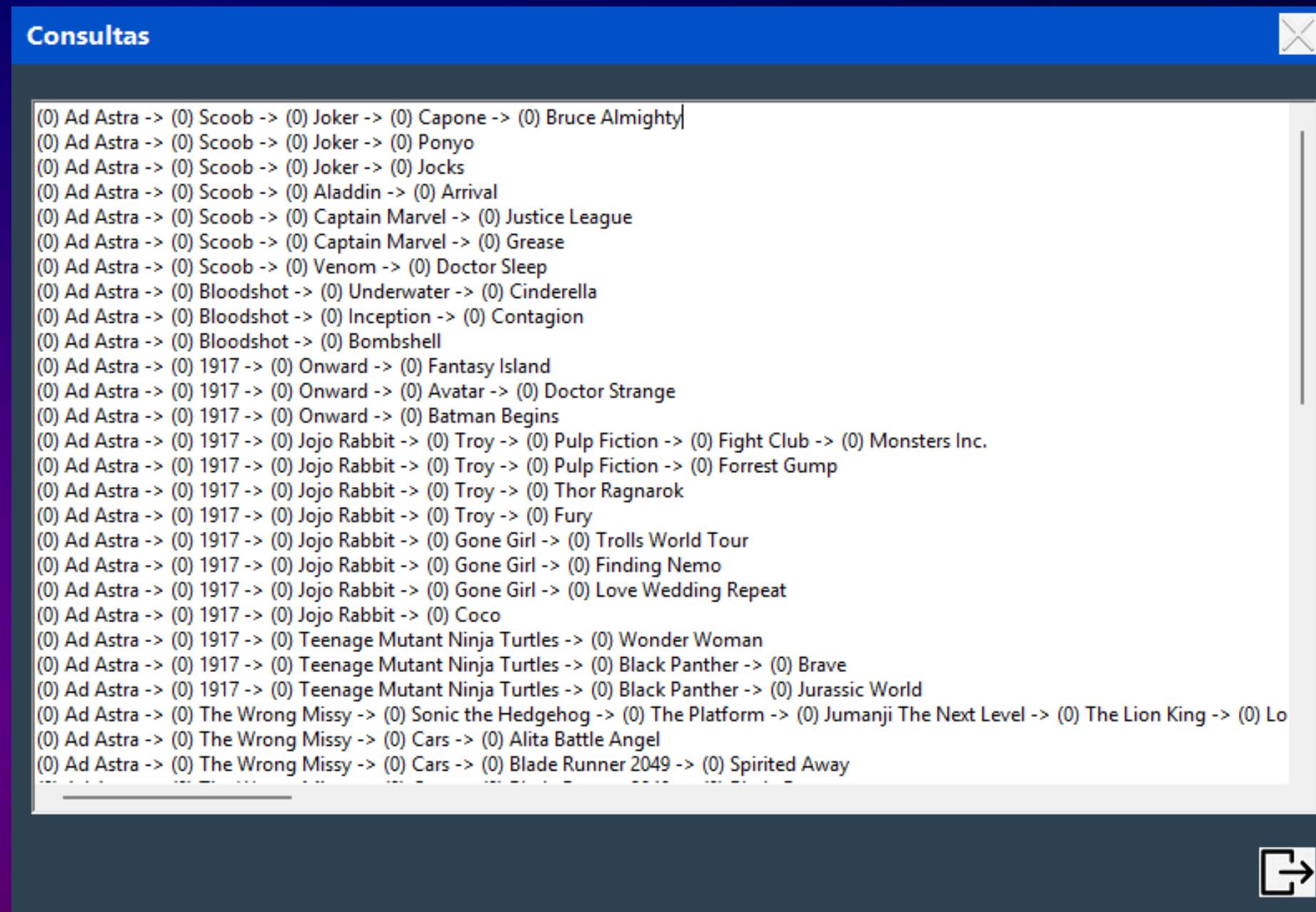
Consulta2

```
1 reference
public String Consulta2(ArbolGeneral<DatoDistancia> arbol)
{
    List<string> todosLosCaminos = new List<string>();
    RecorrerCaminos(arbol, arbol.getDatoRaiz().ToString(), todosLosCaminos);
    return string.Join("\n", todosLosCaminos);
}

2 references
private void RecorrerCaminos(ArbolGeneral<DatoDistancia> nodoActual, string caminoActual, List<string> todosLosCaminos)
{
    if (nodoActual.esHoja())
    {
        todosLosCaminos.Add(caminoActual);
        return;
    }

    foreach (ArbolGeneral<DatoDistancia> hijo in nodoActual.getHijos())
    {
        string nuevoCamino = caminoActual + " -> " + hijo.getDatoRaiz().ToString();
        RecorrerCaminos(hijo, nuevoCamino, todosLosCaminos);
    }
}
```

Resultados



Consulta3

```
1 reference
public String Consulta3(ArbolGeneral<DatoDistancia> arbol) {
    if (arbol == null) {
        return "El arbol esta vacio.";
    }

    string resultado = "";

    Cola<ArbolGeneral<DatoDistancia>> cola = new Cola<ArbolGeneral<DatoDistancia>>();

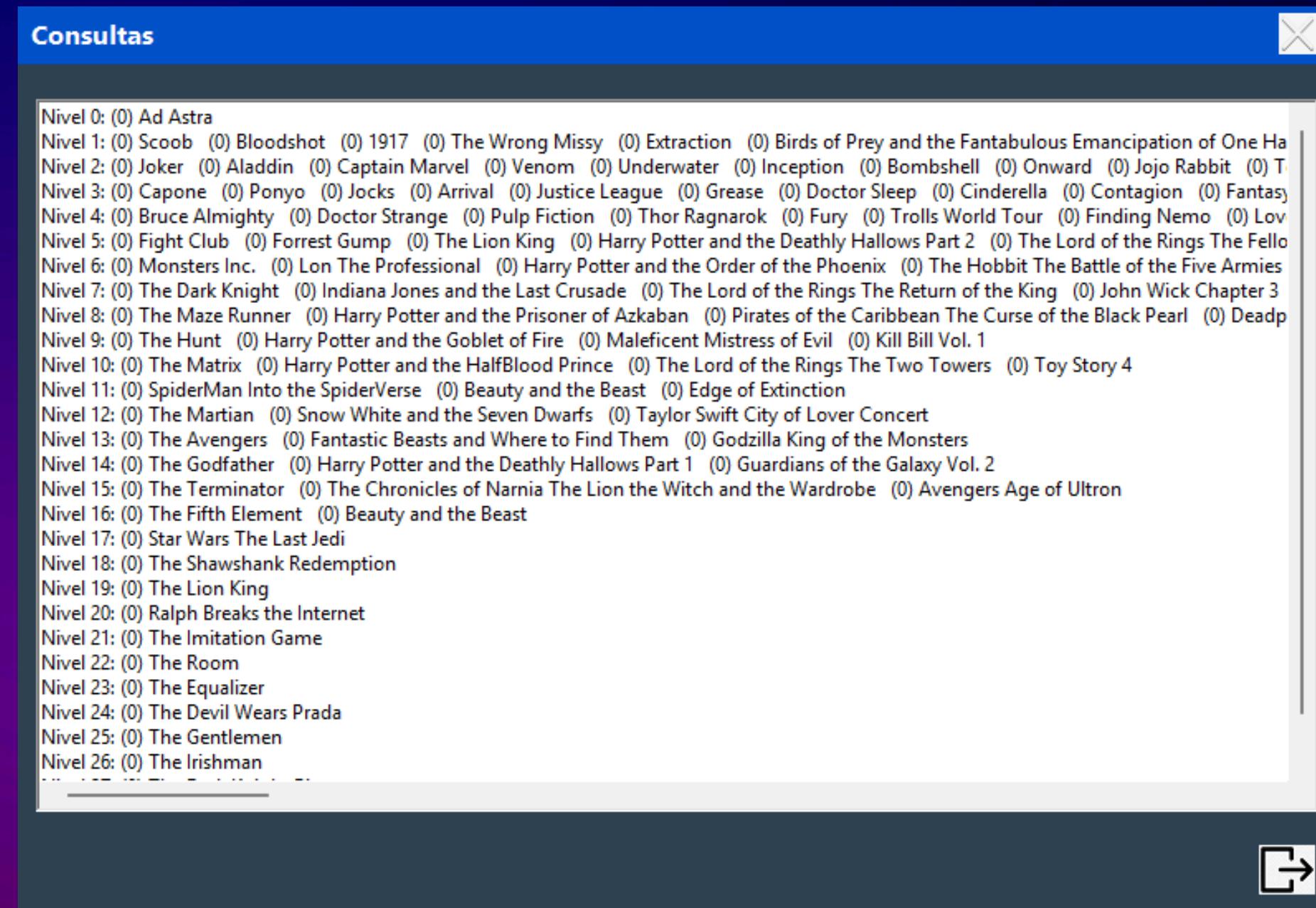
    cola.encolar(arbol);
    cola.encolar(null);

    int nivelActual = 0;
    resultado += "Nivel " + nivelActual + ": ";

    while (!cola.esVacia()){
        ArbolGeneral<DatoDistancia> nodoActual = cola.desencolar();

        if (nodoActual == null) {
            if (!cola.esVacia()){
                nivelActual++;
                resultado += "\nNivel " + nivelActual + ": ";
                cola.encolar(null);
            }
        } else {
            resultado += nodoActual.getDataRaiz().ToString() + "  ";
            foreach (ArbolGeneral<DatoDistancia> hijo in nodoActual.getHijos())
            {
                cola.encolar(hijo);
            }
        }
    }
    return resultado;
}
```

Resultados



Ideas y/o Sugerencias

01

Mejor diseño

Encontrarle la forma por medio de diseño UX una solucion para que todos los resultados de las consultas se vean mas claro de un vistazo y no sea tan dificil leer toda la informacion que tiene el arbol.

02

Optimizaciones

Hacer el codigo mas completo y optimizado para lograr mejores tiempos de respuesta si es que esta aplicacion llegara a ser real y utilizada por usuarios reales.

03

Base de datos

Estos datos al estar mockeados en un archivo .csv es claramente una forma facil y rapida de levantar la aplicacion y probar. Pero esto en un entorno real es impracticable.

04

Mejor legibilidad

Si algun nuevo trabajador ve el codigo entienda de lo que va o lo que quiere intentar hacer el codigo.

Nota: Ayudar con comentarios

Reflexiones y Conclusiones

Este trabajo final, nos enseño como funcionan los algoritmos que todos los dias utilizamos y que por razones que no nos incumben no conocemos a fondo. Por estas razones, creemos que este trabajo aporto mucho valor al conocimiento y entendimiento de como funcionan los algoritmo ahí fuera. En la vida real.