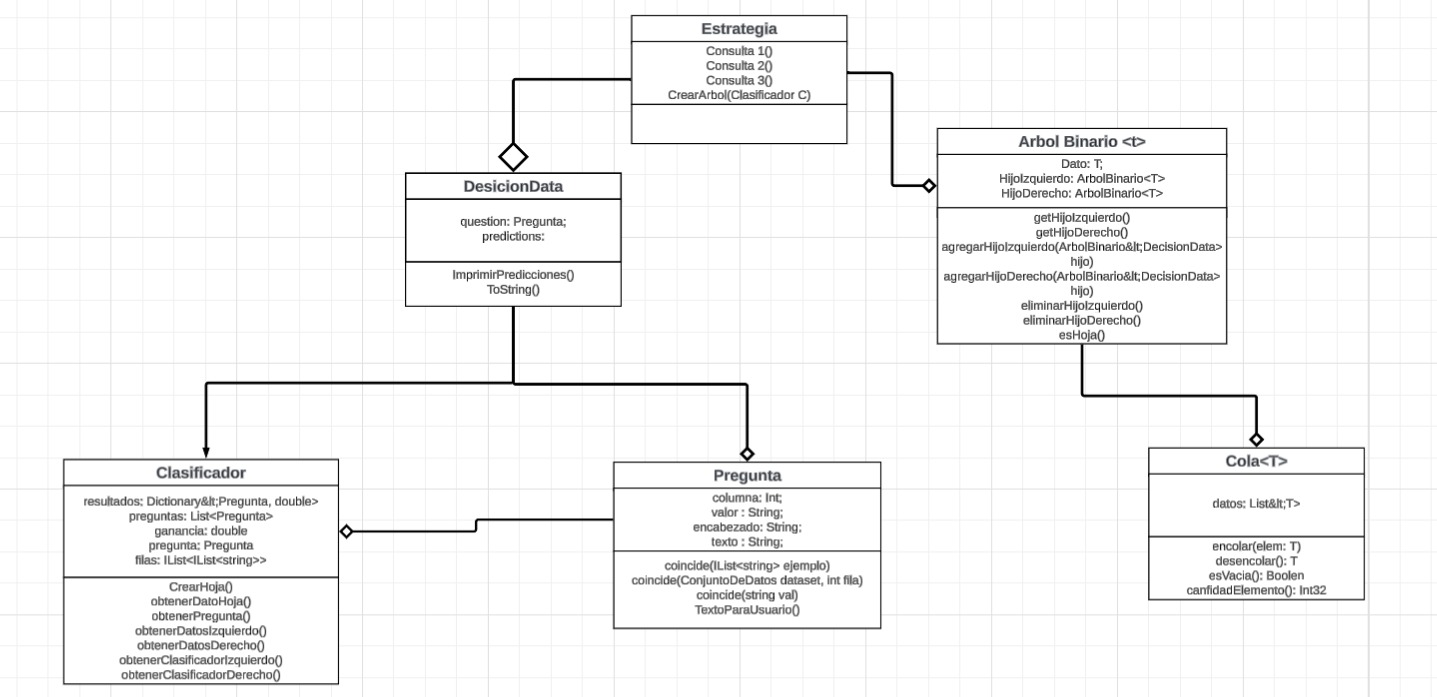
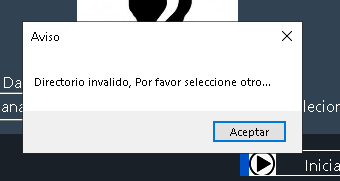
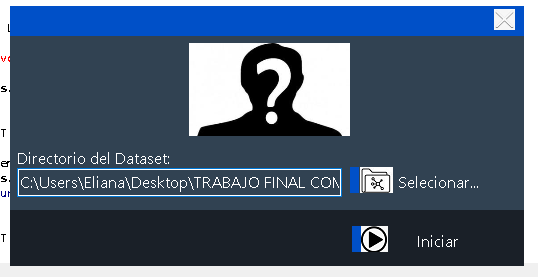
|  |  |  |
| --- | --- | --- |
|  | Complejidad temporal, estructura de datos y Algoritmos. | ALUMNO: Vázquez Eliana, Giménez Augusto, Luna Candela  COMISIÓN: 02  Docente: Guzmán Jesica |
| TRABAJO FINAL |  |

# Diagrama UML:

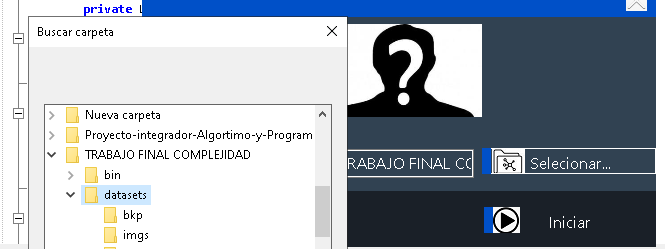


# Pantallas del programa (Forms):

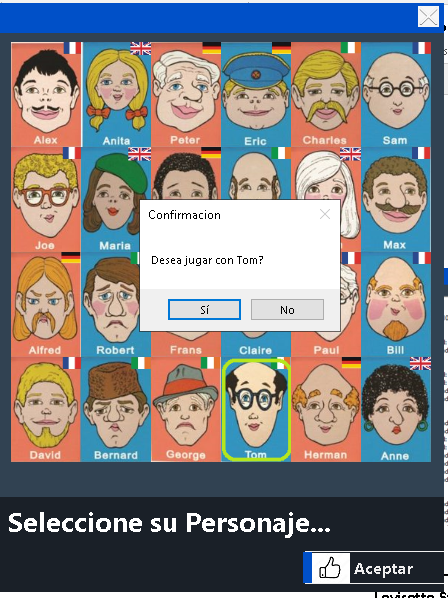
Ni bien ejecutamos el programa nos encontramos con una pantalla inicial que nos permite seleccionar el archivo dataset en caso de seleccionar un dato invalido nos sale otra pantalla de alerta:

*Pantalla de selección de archivo Pantalla de error*

**¡Primer error encontrado!** Al querer iniciar el juego nos salta un aviso de error, para solucionar esto lo que hicimos fue seleccionar la carpeta donde guardaríamos el trabajo practico y seleccionar la carpeta “datasets”, con esto por fin podemos proceder a jugar



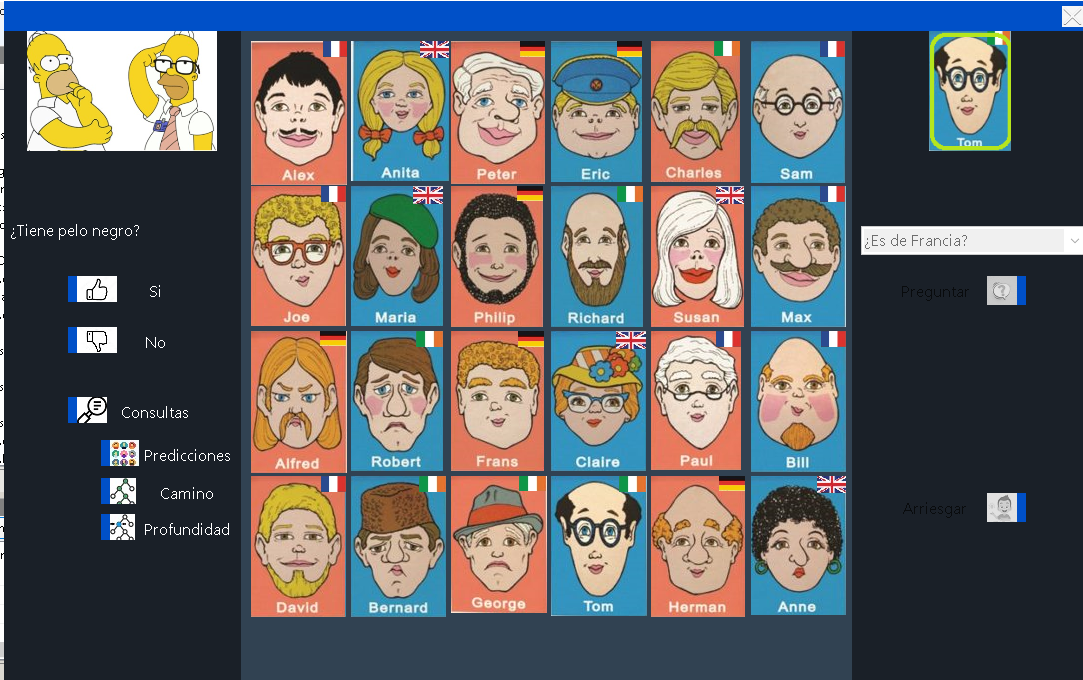
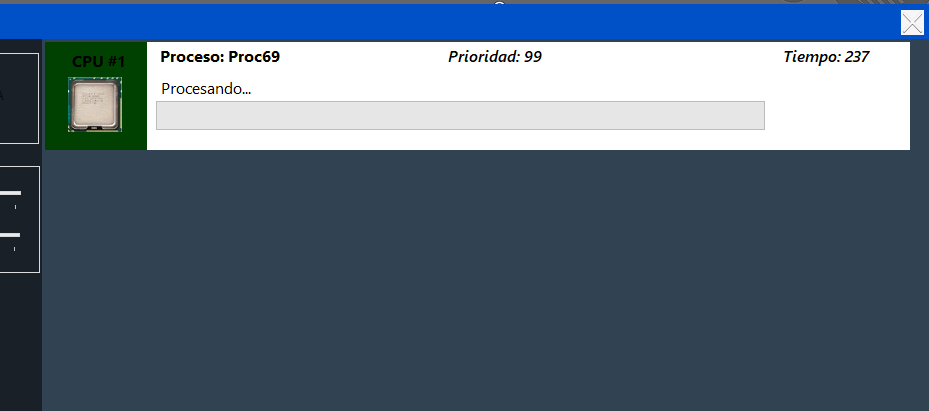
Luego, entramos a la pantalla principal donde nos permite elegir a uno de los personajes disponibles, el cual la maquina va a tratar de adivinarnos

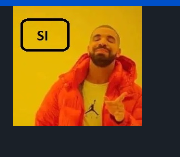
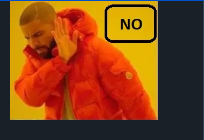


*Pantalla de Inicio*

Posteriormente arrancara haciendo preguntas el cual debemos de negar o afirmar acorde nuestro personaje, una vez que hayamos respondido procederemos nosotros a preguntar, en la parte superior del lado izquierdo si la afirmación es correcta nos aparecerá una foto diciendo que si, y en el caso contrario dirá que no

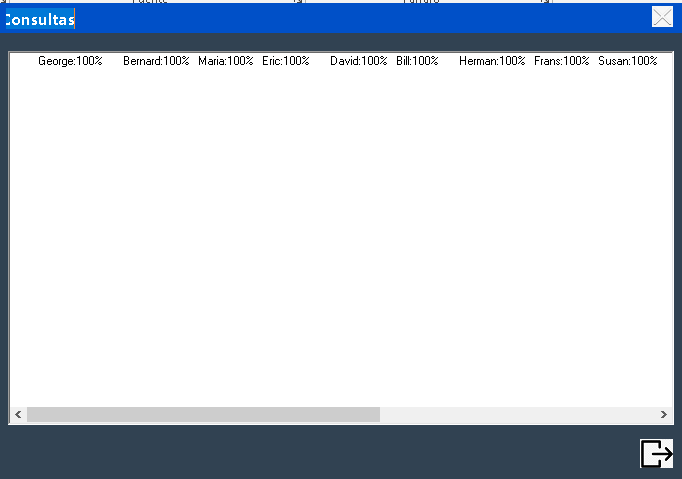
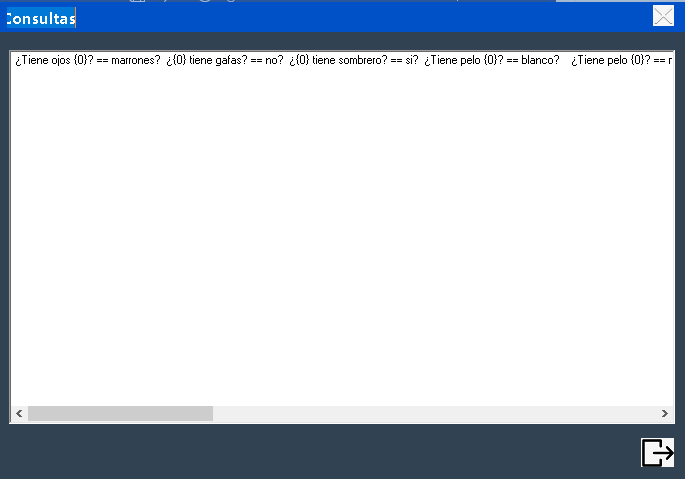
*Pregunta realizada por el bot*

**

*Hemos acertado a la característica del personaje Hemos errado a la característica del personaje*

Luego, veremos una pantalla distinta por cada Consulta:

***Consulta 1 “Predicciones”***

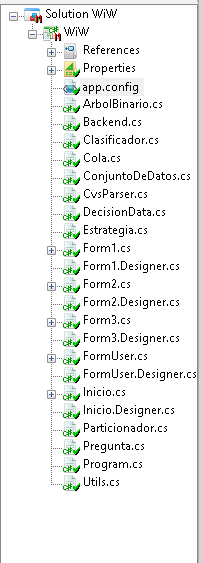
***Consulta 2 “Caminos”***

## 

***Consulta 3 “Profundida***

**Análisis previo a el desarrollo:**

El presente trabajo fue desarrollado sobre una base de clases provistas por la cátedra en el marco de la asignatura *Complejidad Algorítmica*. En particular, se nos suministró un conjunto de clases ya definidas y funcionales, entre las cuales se destacan:



Nuestro rol, como estudiantes, consistió en **comprender y utilizar adecuadamente dichas clases**, respetando su lógica interna, e **implementar la clase Estrategia**, que constituye el núcleo de la lógica del bot para el juego “Who is Who”.

La clase Estrategia fue completamente desarrollada por el equipo, incluyendo:

* El método CrearArbol, que construye el árbol de decisión en base a un clasificador.
* Los métodos Consulta1, Consulta2 y Consulta3, que permiten recorrer y analizar el árbol de decisión bajo distintos criterios (hojas, preguntas y recorrido por niveles).

Esta implementación fue realizada respetando la estructura general del sistema, sin modificar las clases base provistas, y asegurando su correcta integración con los formularios y demás componentes del proyecto.

# Seguimiento del desarrollo:

## Durante el proceso de implementación del trabajo práctico, se siguió una planificación progresiva basada en la comprensión del sistema, la implementación modular de los métodos requeridos y su posterior integración y prueba. A continuación, se detalla paso a paso cómo se desarrolló la clase Estrategia, que constituye el núcleo de la lógica del sistema basado en árboles de decisión.

## 1. Análisis del material provisto por la cátedra

## La cátedra proveyó una serie de clases fundamentales que conforman la base del sistema:

## DecisionData: Encargada de encapsular datos correspondientes a decisiones, ya sean preguntas internas del árbol o predicciones (hojas). Esta clase se utiliza como tipo genérico en el árbol de decisión.

## Clasificador: Proporciona los métodos necesarios para construir el árbol de decisión. Permite identificar si el nodo actual debe ser una hoja (crearHoja()), acceder a su predicción (obtenerDatoHoja()), o construir preguntas con sus respectivos clasificadores izquierdo y derecho.

## ArbolBinario<T>: Estructura de datos que permite representar el árbol de decisión de manera binaria, incluyendo operaciones para agregar hijos, obtener hijos y acceder al dato raíz.

## Cola<T>: Estructura auxiliar utilizada para recorrer el árbol por niveles, fundamental para la implementación de la consulta nivel por nivel.

## Se analizó la lógica de cada clase provista sin modificar ninguna de ellas, tal como lo indicaban las consignas del trabajo.

## 2. Implementación del método CrearArbol

## El método CrearArbol(Clasificador clasificador) es el encargado de construir recursivamente el árbol de decisión, usando los datos y métodos del Clasificador provisto. Se definió la lógica de la siguiente forma:

## Si el clasificador indica que se debe crear una hoja (crearHoja() devuelve true), se crea un nuevo nodo con un objeto DecisionData que contiene una predicción final (por ejemplo, "Es Ana").

## Si se trata de un nodo interno, se crea un nuevo nodo con la pregunta obtenida mediante obtenerPregunta().

## Luego, se construyen recursivamente los subárboles izquierdo y derecho usando los clasificadores correspondientes, y se agregan como hijos del nodo actual.

## Este proceso garantiza que el árbol se construya de manera completa y coherente, respetando la estructura lógica de un árbol de decisiones binario.

## 3. Desarrollo de métodos de consulta

## Una vez construido el árbol, se desarrollaron tres métodos de consulta diferentes, cada uno con un propósito específico para analizar y recorrer la estructura del árbol:

## 📌 Consulta1: Listado de predicciones (hojas)

## Este método recorre todo el árbol de forma recursiva, y devuelve únicamente los nodos que son hojas (es decir, aquellos que no poseen hijos y representan una predicción final). Se usó una lógica basada en recorrido postorden.

## En cada paso, se evalúa si el nodo es hoja, y se acumula su predicción en un string resultado.

## Si el nodo posee hijos, se llama recursivamente a la función sobre ellos.

## Este método resulta útil para identificar todas las posibles salidas o conclusiones del árbol de decisión.

## 📌 Consulta2: Listado de preguntas

## Este método tiene como finalidad extraer únicamente las preguntas formuladas en los nodos internos del árbol (es decir, todos aquellos nodos que no son hojas).

## Se recorren los nodos en forma recursiva, en lo que podría considerarse un recorrido preorden modificado.

## Se evalúa si el nodo actual tiene al menos un hijo (por lo tanto, es una pregunta), y en tal caso se incluye su contenido.

## Luego se continúa el recorrido por los hijos izquierdo y derecho.

## Este método permite visualizar toda la lógica condicional que estructura el proceso de clasificación.

## 📌 Consulta3: Recorrido por niveles

## Este método implementa un recorrido por niveles (también conocido como BFS o Breadth-First Search), utilizando una cola auxiliar para mantener el orden correcto.

## Se encola el nodo raíz.

## Mientras la cola no esté vacía, se desencola un nodo, se procesa su contenido, y se encolan sus hijos (si los tiene).

## Este recorrido permite ver la evolución del árbol desde la raíz hacia las hojas, nivel por nivel, como si se "leyera de arriba hacia abajo".

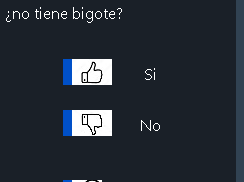
# Sugerencias de posibles mejoras:

**Sistema de entrenamiento personalizado:** Se podría permitir que el usuario final ingrese nuevos ejemplos o decisiones mediante la interfaz, y que el sistema reconstruya o expanda el árbol automáticamente.

Esto implicaría:

* Adaptar el clasificador para tomar ejemplos nuevos.
* Aplicar algún algoritmo de aprendizaje de árboles
* Guardar el nuevo árbol generado.

**Mejorar la interpretación de las preguntas:** Puede ser medio confuso la pregunta para ciertos jugadores el hecho de que la pregunta sea en negativa, como, por ejemplo: “¿NO tiene bigote?”, puede llegar a surgir un error de interpretación y la llegada a un resultado incorrecto



**Editor de personajes y preguntas:** Para que el juego sea más versátil, se puede permitir que el usuario cargue su propio dataset de personajes y preguntas. Esto daría lugar a versiones personalizadas del juego, por ejemplo:

* “¿Qué animal es?” usando preguntas sobre características biológicas.
* “¿Qué país es?” usando datos geográficos, culturales, etc.
* “¿Qué superhéroe es?” para versiones temáticas.

Esto convertiría al sistema en una plataforma de juegos basada en árboles de decisión, no limitada solo al juego clásico.

**Que al final de cada partida se muestre el jugador elegido por la maquina**

**Conclusión:**

La realización de este trabajo final nos permitió aplicar de manera concreta conceptos fundamentales de la materia, tales como la implementación y análisis de árboles binarios, la recursividad, la estructuración de datos y la interacción entre clases.

Pudimos comprender cómo un **árbol de decisión** no solo es una herramienta útil en inteligencia artificial y clasificación de datos, sino también una excelente estructura lógica para modelar el comportamiento de juegos como *Who is Who*.

La implementación de la clase Estrategia implicó un desafío importante al momento de pensar en la **recursividad, modularidad y reutilización del código**, y también al interpretar correctamente el funcionamiento de las clases provistas por la cátedra.

En resumen, este proyecto no solo reforzó nuestros conocimientos teóricos, sino que también nos permitió desarrollar una solución funcional, extensible y con potencial para futuras mejoras en un entorno lúdico y desafiante.

Enlaces:

<https://github.com/vzblea/tp-final-Complejidad>

<https://www.mycompiler.io/view/89190GunWxm> (ver y entender recorrido árbol)

<https://lucid.app/lucidchart/6f5eff2e-d7e8-4050-b546-006512386e0d/edit?viewport_loc=-1122%2C-1268%2C4231%2C2053%2C0_0&invitationId=inv_0b0cebe6-b270-47a9-b7f2-6a1fafe0310c>