TRABAJO FINAL

CTEDYA

MIRANDA IBARRA MELINA

Introducción

El presente informe tiene como objetivo integrar los contenidos vistos en la materia. Se expondrá la resolución del proyecto machine learning planteado por la catedra con el fin de desarrollar el procedimiento, exponer los componentes del programa y explicar el funcionamiento del mismo, a demás de integrar los conceptos adquiridos sobre búsqueda de caminos, recorrido de arboles binarios y hacer el uso del TAD cola.

Desarrollo

Para dar comienzo al desarrollo del proyecto se comenzó con la visualización de todas las clases que lo comprenden y se realizó un análisis de lo que se debía hacer. Durante este análisis se pudo observar que es un sistema de toma de decisiones basado principalmente en un árbol binario, el cual se deberá recorrer para hallar sus valores y así, complementar las otras funciones del proyecto para que este sea funcional.

Una vez realizado esto, se comenzó a codificar las funciones requeridas. En el proceso se encontraron los siguientes problemas:

En el comienzo del desarrollo de la función **CrearArbol**

- No saber cómo meterle datos al árbol.
- No saber crear las preguntas para meterlas en el árbol.
- No poder crear el Árbol binario.
- No insertar las hojas al árbol.

Pero luego, gracias a las consultas, al análisis parte por parte de cada elemento del enunciado y gracias a los apuntes tomados en clase se logró realizarlo.

También surgieron los soguientes en la creación de las funciones **Consulta**

- No lograr que la **Consulta1** retorne los resultados de las hojas del árbol.
- Cuando **Consulta1** está a punto de predecir el personaje sale un mensaje de error y se detiene el programa.

  Estos fueron resueltos cuando se realizó la inserción de hojas al árbol.

- En la **Consulta2** no saber cómo traer todos los caminos del árbol.

  Este problema fue resuelto chequeando si el dato procesado pertenece a una hoja, si lo era, había que agregar la lista camino a la lista caminos, si no, seguir con el proceso preorden.

- En la **Consulta3** que no traiga los datos del árbol separado por niveles.

  En esta consulta se estaba retornando 2 veces la variable string “cola” y, además, no se estaba concatenando el texto, solo se estaba reemplazando al anterior por el nuevo. Fue solucionado usando “cola += “.

Problema general

No poder imprimir los textos uno debajo del otro.

Este fue solucionado concatenando un *"\n"* en la variable que almacena los datos.

Pantallas que componen el sistema codificado

Función **CrearArbol**

```
public ArbolBinario<DecisionData>CrearArbol(Clasificador clasificador)
{
    DecisionData pregunta = new DecisionData(clasificador.obtenerPregunta());
    ArbolBinario<DecisionData> arb = new ArbolBinario<DecisionData>(pregunta);
    DecisionData nodohoja;

    if(clasificador.crearHoja()==true)
    {
        nodohoja = new DecisionData(clasificador.obtenerDatoHoja());
        arb = new ArbolBinario<DecisionData>(nodohoja);
    }
    else
    {
        pregunta = new DecisionData(clasificador.obtenerPregunta());
        arb = new ArbolBinario<DecisionData>(pregunta);
        arb.agregarHijoDerecho(CrearArbol(clasificador.obtenerClasificadorDerecho()));
        arb.agregarHijoIzquierdo(CrearArbol(clasificador.obtenerClasificadorIzquierdo()));

    }

    return arb;
}
```

En esta función perteneciente a la clase Estrategia se comenzó creando los objetos que compondrán el árbol binario de decisiones, como lo es *pregunta* (el cual almacena una pregunta brindada por el clasificador), *arb* (que almacena a *pregunta* en su contenido), y *nodohoja* (que almacena el dato de la hoja).

Luego, se pregunta si el conjunto de datos corresponde a un nodo-hoja, cuando es verdadero se almacena en el objeto *nodohoja* el dato de la hoja y este se almacena en *arb*. Cuando es falso, vuelve a almacenar una nueva pregunta en el objeto *pregunta* y la guarda nuevamente en *arb*.

Posteriormente, se agregan los hijos derechos e izquierdos a *arb* haciendo una llamada recursiva con sus respectivos clasificadores como parámetro.

Para finalizar, la función retorna el árbol binario *arb*.

Función **Consulta1**

```
public String Consulta1(ArbolBinario<DecisionData> arbol)
{
    return arbol.contenidoHoja();
}
```

Esta función también perteneciente a la clase Estrategia retorna una función contenidoHoja perteneciente a la clase ArbolBinario.

```
public string contenidoHoja()
{
    string contenido = "";

    if(esHoja())
        contenido = dato.ToString() + " \n ";

    if(hijoIzquierdo!=null)
        contenido =  contenido + hijoIzquierdo.contenidoHoja();

    if(hijoDerecho!=null)
        contenido =   contenido + hijoDerecho.contenidoHoja();


    return contenido;
}
```

Esta retorna una variable *"contenido"* donde se almacena el contenido de la raíz del árbol y luego sus hijos, esto lo hace con un recorrido preorden donde primero procesa la raíz, luego el hijo izquierdo y por último el hijo derecho, los cuales llaman recursivamente a la función.

Función **Consulta2**

```
public String Consulta2(ArbolBinario<DecisionData> arbol)
{
    return arbol.Caminos(camino,caminos,copia);

}
```

Esta función de la clase Estrategia retorna la función Caminos también perteneciente a la clase ArbolBinario.

```
public string Caminos(ArrayList camino, ArrayList caminos, ArrayList copia)
{
    string preOrden = dato.ToString() + "\n";
    if(dato!=null)
    {
        camino.Add(this);
        if(esHoja())
        {
            //guarda camino en una lista de copia
            copia.AddRange(camino);
            //copia camino en caminos
            foreach(var i in camino)
                caminos.Add(i);
        }
        else
        {
            if(hijoIzquierdo!=null)
                preOrden = preOrden + hijoIzquierdo.Caminos(camino,caminos,copia);
            if(hijoDerecho!=null)
                preOrden = preOrden + hijoDerecho.Caminos(camino,caminos,copia);
        }

    }

    return preOrden;
}
```

Esta toma como parámetro tres listas. En la lista camino almacena el árbol, copia almacena camino y guarda una copia de camino en caminos.

Tiene una variable llamada preOrden donde almacenará los datos de la raíz y los hijos.

Para que todo esto funcione primero pregunta si dato (el contenido de la raíz) no es null, si se cumple, camino guardará al árbol y preguntará si dato es hoja, si es así, entonces almacenará camino en caminos, si no, preguntará si hijo izquierdo no está vacío y almacenará en preOrden el valor anterior de preOrden concatenando la llamada recursiva a Caminos, lo mismo sucede con el hijo derecho.

Para finalizar, la función retorna preOrden.

Función **Consulta3**

```
public String Consulta3(ArbolBinario<DecisionData> arbol)
{
    ArbolBinario<DecisionData> arbolaux;
    c.encolar(arbol);
    c.encolar(null);
    string cola = "";
    while(!c.esVacia())
    {
        arbolaux=c.desencolar();
        if(arbolaux == null)
        {
            if(!c.esVacia())
                c.encolar(null);
        }
        else
        {
            cola += arbolaux.getDatoRaiz().ToString() + "\n";
            if(arbolaux.getHijoIzquierdo()!=null)
                c.encolar(arbolaux.getHijoIzquierdo());
            if(arbolaux.getHijoDerecho()!=null)
                c.encolar(arbolaux.getHijoDerecho());
        }
    }

    return  cola;
}
```

En esta función se declaró un nuevo objeto de tipo ArbolBinario<DecisionData> arbolaux.

Se inicia encolando a arbol en la cola y luego un null ya que sirve para separar por niveles.

Se declara la variable cola en la cual se almacenarán los datos de la raíz y los hijos del árbol.

Se instancia un bucle while cuya condición de corte es que la cola esté vacía, mientras esto no pase arbolaux almacenará lo que desencole la cola y se preguntará si este es null, si lo es, pregunta si c no está vacía y encola null, si está vacía guarda en cola el dato de la raíz, luego pregunta si el hijo izquierdo no es null y encola el contenido de hijo izquierdo del arbol auxiliar, lo mismo hace con el hijo derecho.

Para finalizar, retorna la variable cola.

Diagrama UML de las clases más importantes involucradas en las especificaciones del sistema.

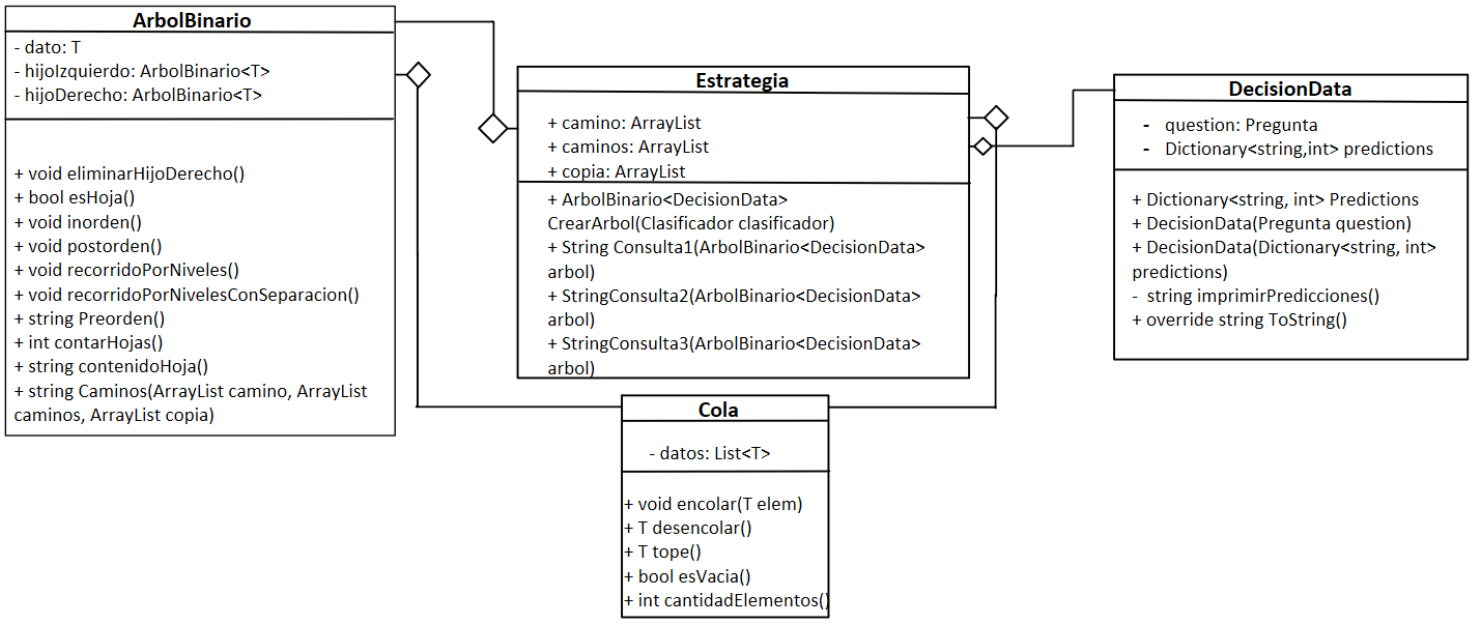


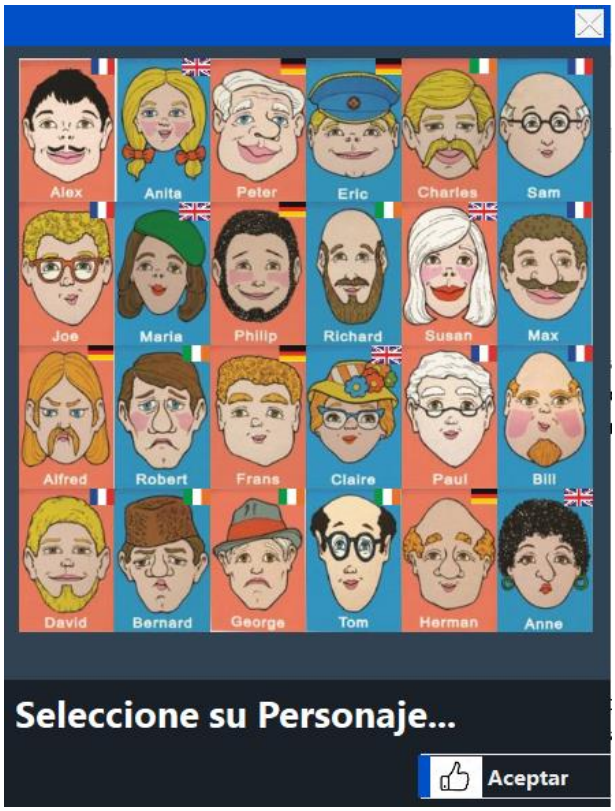
Diagrama UML

Ejecución

Para comenzar a jugar lo primero que se hará será cargar el directorio de la carpeta dataset en la pantalla de inicio del juego y click en iniciar.

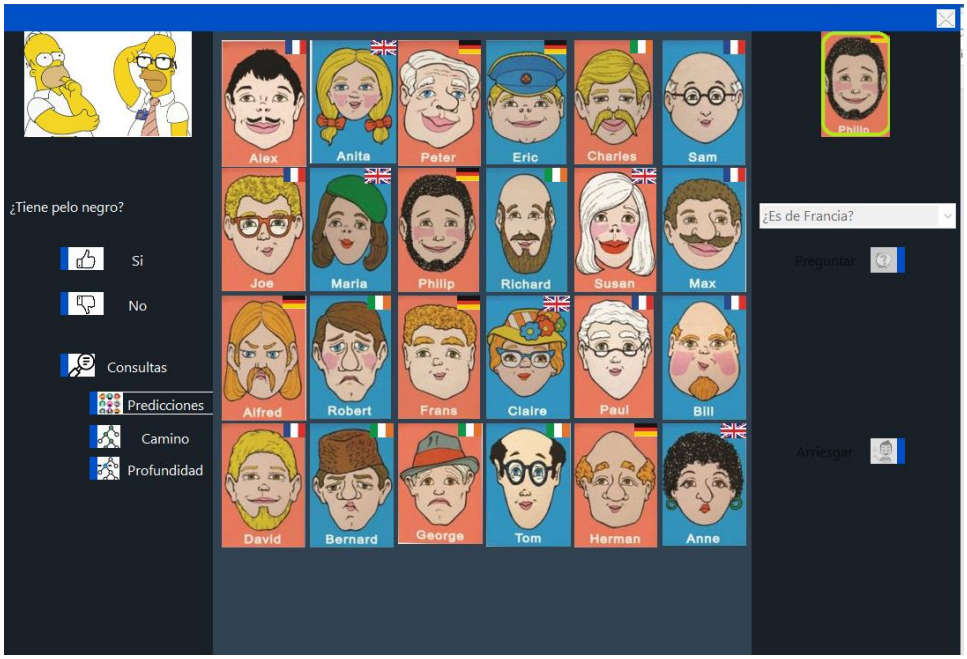
Pantalla de inicio

Una vez iniciado el juego, se elegirá el personaje con el que se va a jugar



Elección de personaje

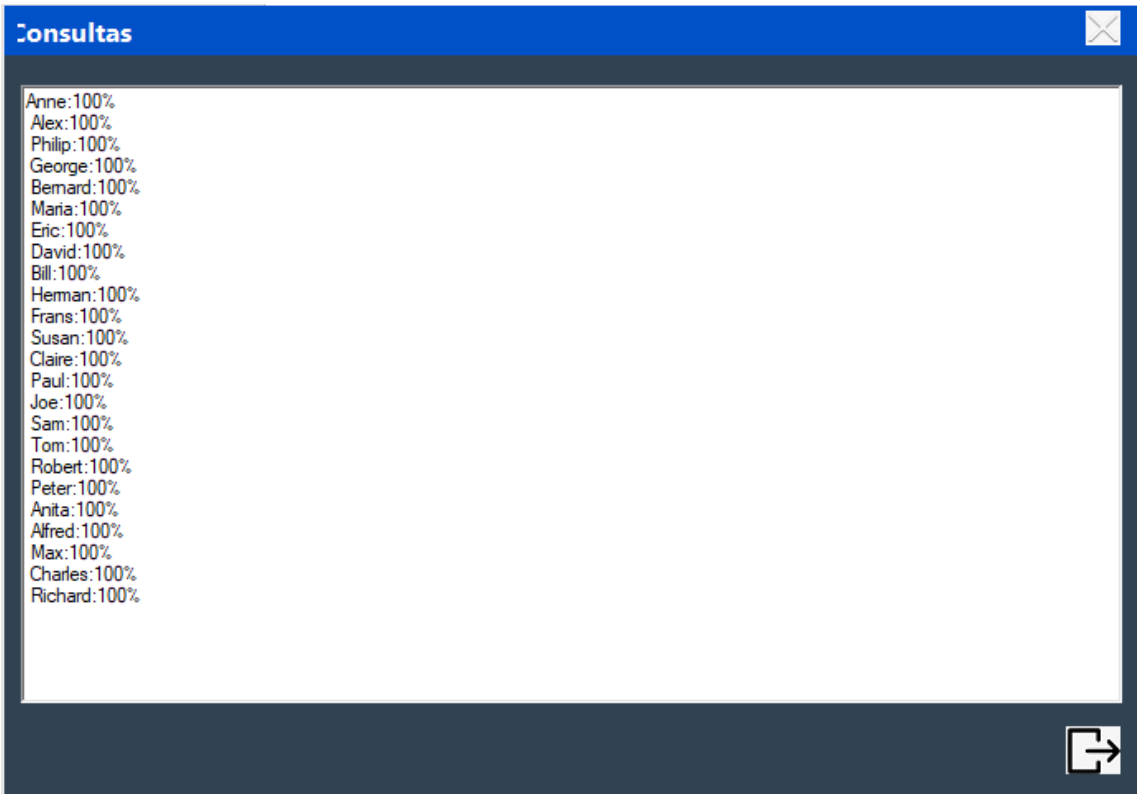
Una vez elegido, esta será la pantalla que se mostrará, donde se puede ver las preguntas que hace la máquina con respecto al personaje, las opciones de si o no para que se responda, las consultas que se pueden hacer, la opción de elegir qué preguntas hacerle sobre su personaje a la máquina y la opción de arriesgar.



Pantalla inicial

Las consultas son las siguientes:

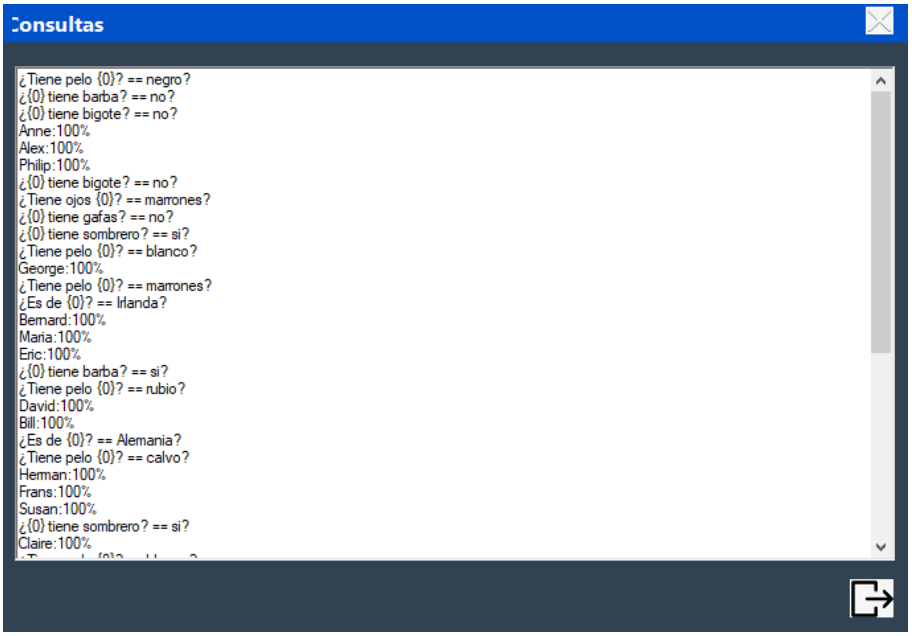
Consulta Predicciones



Predicciones

Esta consulta lo que muestra es el porcentaje de predicción de los posibles personajes que pueden llegar a ser según las preguntas que la máquina vaya haciendo, esta lista de personajes se va reduciendo pregunta tras pregunta hasta que llega a un nombre el cual es el del personaje elegido.

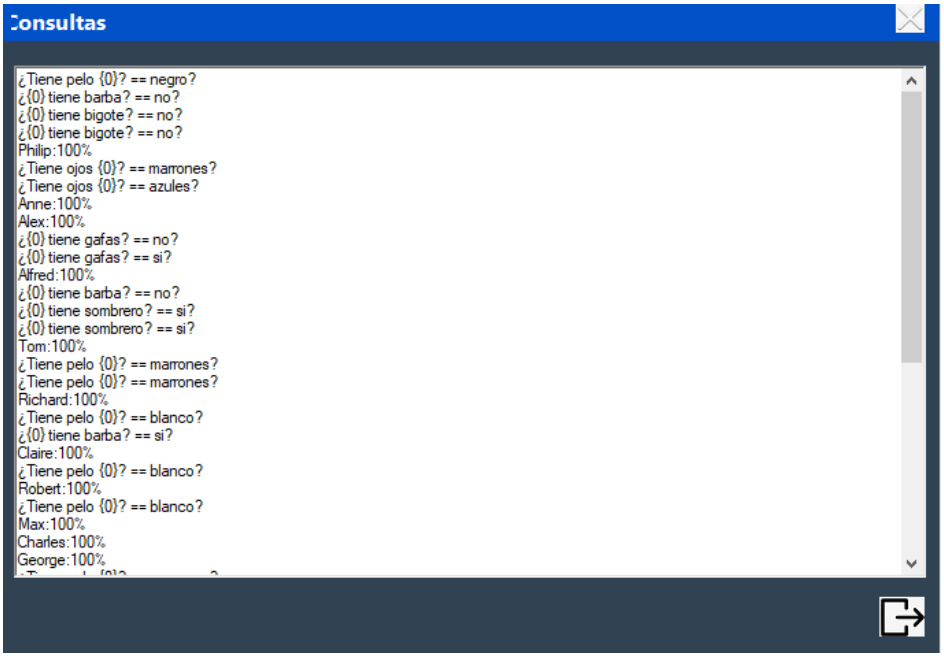
Consulta Camino



Caminos

En esta consulta se pueden observar todos los posibles caminos hacia una predicción, muestra las preguntas que llevan hacia cada uno de los personajes, y, a medida que avanza el juego estos se van reduciendo.

Consulta Profundidad

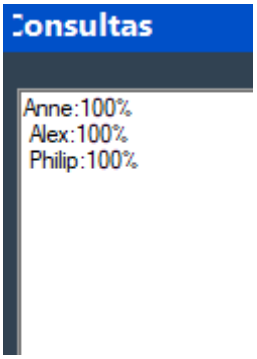


Profundidad

En esta consulta lo que se muestra son los niveles del árbol de decisión desde la raíz hasta las hojas.

A modo de ejemplo se desarrollará la explicación del juego con Phillip, esto valdrá para todos los personajes que sean seleccionados al principio del juego.

Cuando comienza el juego la máquina hace la primera pregunta *"¿Tiene pelo negro?"* el personaje elegido si lo tiene, entonces, se respondió *"Si"* por lo que generó una nueva lista de predicciones en la cual se observa el nombre del personaje seleccionado. Como se dijo anteriormente en la explicación de *Consulta Predicciones*, a medida que avanza con las preguntas esta lista se va reduciendo hasta llegar a adivinar el personaje.

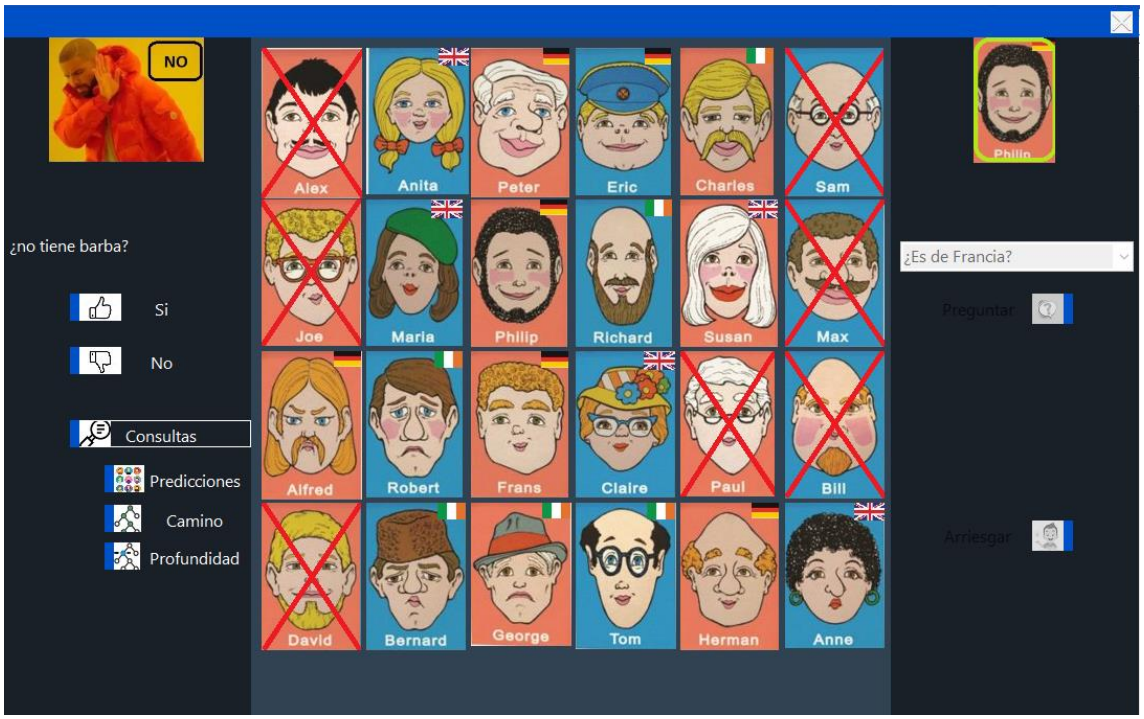


Nueva lista de predicciones

Cuando se responde la pregunta que hace la máquina, toca que se pregunte a la misma sobre su personaje con las preguntas que se muestran a continuación

Lista de preguntas

Se preguntó si su personaje era de Francia y respondió que no, entonces, se procede a tachar a los que provienen de ese país.



Pregunta a la máquina

Ella pregunta "*¿No tiene barba?"* y el personaje si tiene, entonces se responde "*No*", como en su lista de predicciones sólo hay un personaje con estas características nos da la predicción y finaliza el juego.

Predicción

Conclusiones

Si bien se tuvieron adversidades se logró el objetivo final el cual consiste en aprender más sobre el funcionamiento de los árboles binarios y su implementación en los diferentes campos de la tecnología. También así, el funcionamiento del proceso de

hallar caminos y los recorridos de estos para obtener información de diferentes maneras.