



**Instituto Tecnológico de Costa Rica.**

**Área de Ingeniería en Computadores.**

**Lenguajes Intérpretes y Compiladores.**

**Profesor: Marco Rivera.**

**Grupo 02.**

**Tarea 2: RestauranTEC.**

**Estudiantes:**

- Santiago Brenes Torres (2019063875)
- Tomás Segura Monge (2018099729)



**By Santiago Brenes,  
Tomás Segura.**

## **Documentación.**

---

Los sistemas expertos (SE) son aplicaciones de cómputo que involucran experiencia no algorítmica, para resolver cierto tipo de problema. Por ejemplo, los sistemas expertos se usan para el diagnóstico al servicio de humanos y máquinas. Existen SE que juegan ajedrez, que plantean decisiones financieras, que configuran computadoras, que supervisan sistemas de tiempo real, que deciden políticas de seguros, y llevan a cabo demás tareas que requieren de experiencia humana.

## Objetivo General

- Desarrollar una aplicación que permita reafirmar el conocimiento del paradigma de programación lógico.

## Objetivos Específicos

- Crear una aplicación que se comporte como un Experto utilizando Prolog.
- Aplicar los conceptos de programación lógico.
- Crear y manipular listas como estructuras de datos.

A continuación se presenta la documentación externa del proyecto RestaurantTEC que se puede encontrar en [GitHub](#). El esquema del documento es el siguiente:

- 1 - Documentación del código.
  - 1.1 - Descripción de los hechos y reglas implementadas.
    - 1.1.1 - Hechos.
    - 1.1.2 - Reglas.
  - 1.2 - Descripción de las estructuras de datos desarrolladas.
  - 1.3 - Descripción detallada del algoritmo general de solución.
  - 1.4 - Problemas sin solución.
  - 1.5 - Plan de Actividades.
  - 1.6 - Problemas solucionados.
  - 1.7 - Conclusiones y Recomendaciones del proyecto.
- 2 - Bitácoras.
  - 2.1 - Bitácora de reuniones.
  - 2.2 - Bitácora de Santiago.
  - 2.3 - Bitácora de Tomás.
- 3 - Referencias.

## 1.1 - Descripción de los hechos y reglas implementadas.

---

### 1.1.1 - Hechos.

 restaurante([ ])

**Descripción:** almacena la información de un restaurante individual.

**Parámetros:** recibe una lista con todos la siguiente información:

- Nombre.
- Estilo.
- Menú.
- Ubicación.
- Dirección exacta.
- Capacidad.
- Disposiciones.

**Ejemplo:**

```
restaurante(["Bella Italia",  
            "italiano",  
            ["Pizza", ["Jamon y queso", "Suprema", "Hawaina"], "Calzone", []],  
            ["vino", "Fanta"],  
            ["San Pedro", "300m Sur de la entrada principal de la Universidad  
10]]).
```



**tipoValor(valor, tipo, pregunta)**

**Descripción:** asocia los valores de la base de datos de restaurantes a un tipo y una pregunta para responder en la interfaz.

**Parámetros:**

- valor: string de la base de datos.
- tipo: indica el tipo de valor que representa.
- pregunta: respuesta del sistema experta para el usuario de modo que la conversación continúe.

**Ejemplo:**

```
tipoValor("italiano", estilo, "¿Algún plato en particular?").  
tipoValor("Comida Rápida", estilo, "¿Algún plato en particular?").
```

## 1.1.2- Reglas.

## frase\_usuario(X, N)

**Descripción:** recibe la oración que redacta el usuario y la analiza para encontrar coincidencias con la base de datos.

### Parámetros:

- X: lista de palabras generada a partir de la oración que digita el usuario.
- N: variable que almacena una lista con las palabras que tienen coincidencia en la base de datos.

### Ejemplo:

```
frase_usuario(["quiero", "comer", "italiano"], N).  
N = "italiano" .
```

## miembro(B, [X|\_])

**Descripción:** analiza si un elemento es parte de una lista.

### Parámetros:

- B: elemento para comparar.
- [X|\_]: lista para analizar.

## eliminar(A, [A|B], B)

**Descripción:** elimina un elemento de una lista.

### Parámetros:


- A: elemento para eliminar.
- B: lista resultado.
- [A|B]: lista para analizar.

## readInput

**Descripción:** Es lo que podríamos llamar un “hecho recursivo”. Consiste en el ciclo del programa. Se llama continuamente, desplegando preguntas y recibiendo respuestas hasta

encontrar una coincidencia.

**Parámetros:** Ninguno.

 repetitions(X, C)

**Descripción:** recibe la oración que redacta el usuario y cuenta la cantidad de veces que se repite en la base de datos.

**Parámetros:**

- X: lista de palabras generada a partir de la oración que digita el usuario.
- C: contador de la cantidad de veces que se ha repetido una palabra de la lista X en la lista L.

**Ejemplo:**

```
?- repetitions(["quiero", "comer", "pizza"], C).  
C = 2
```

 get\_restaurante(X, R)

**Descripción:** recibe la oración que redacta el usuario y retorna el restaurante que tenga alguna palabra de la oración en su base de datos.

**Parámetros:**

- X: lista de palabras generada a partir de la oración que digita el usuario.
- R: valor de retorno que contiene una lista con la informacion del restaurante.

**Ejemplo:**

```
?- get_restaurante(["quiero", "comer", "pizza"], C).  
C = ["Bella Italia", "italiano", "pizza", "jamon", "queso", "suprema", "hawair"]
```

 restauranTEC

**Descripción:** representa el main del programa. Se encarga de generar la conversación y ser un mediador entre el usuario y la lógica.

**Parámetros:** ninguno.

## 1.2 - Descripción de las estructuras de datos desarrolladas.

---

La información de cada restaurante se encuentra en listas. Esto debido a que consideramos que sería más sencillo manejar la base de datos de esta manera, se recorre la lista sin necesidad de recorrer otras listas dentro de la principal.

Además, la oración de entrada del usuario también se divide en palabra por palabra, como elementos de una lista. Al igual que los restaurantes, el manejo de las palabras de entrada resulta más sencillo de manejar y recorrer. Por lo tanto, cada vez que se requiere de un dato de los restaurantes entonces se recorre la lista para obtenerlo.

Por lo tanto, cada vez que se requiere de un dato de los restaurantes entonces se recorre la lista para obtenerlo.

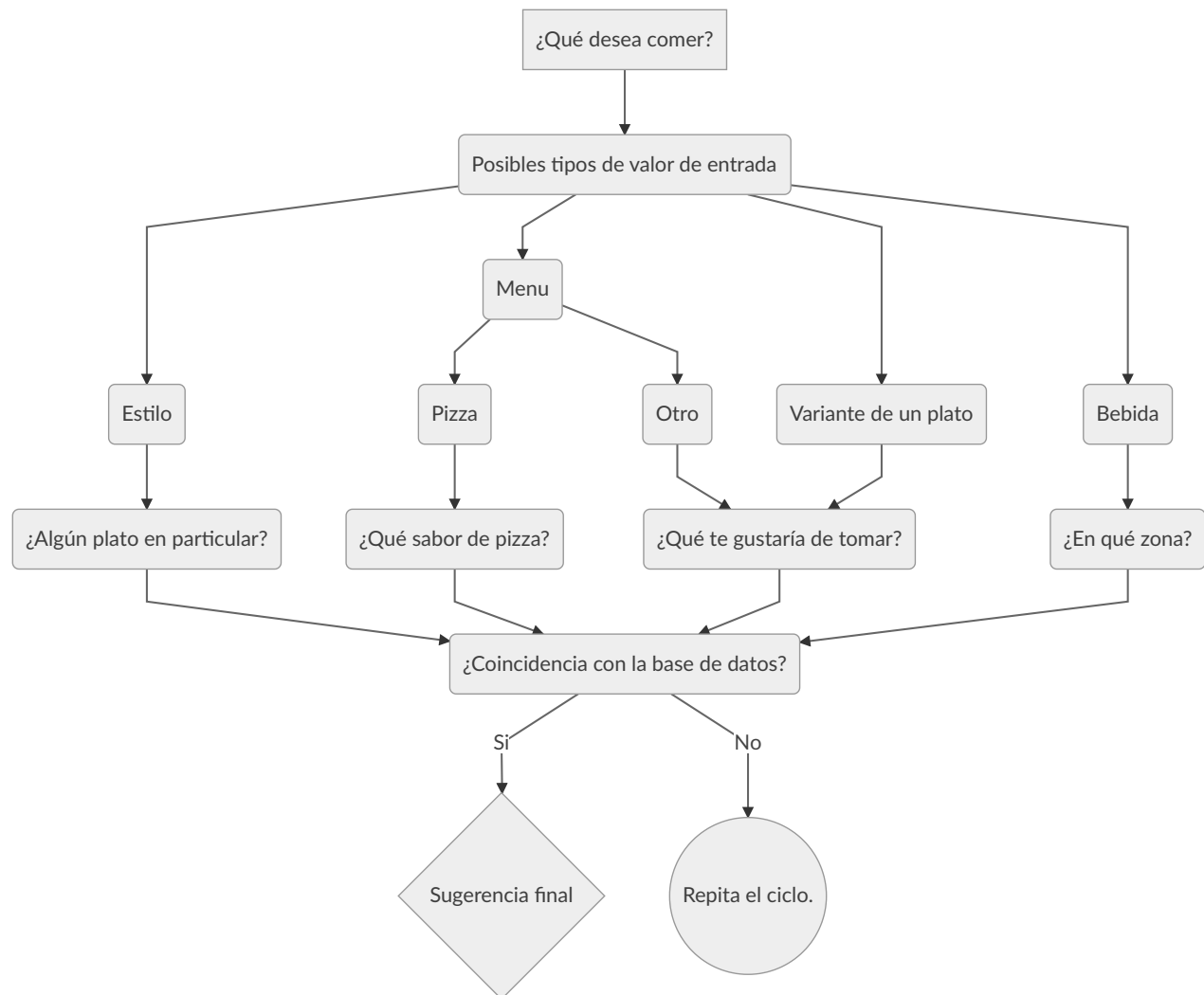
Ejemplo de algunas estructuras de restaurantes:

```
restaurante([ "Burger Fantasy",  
              "rapida",  
              "hamburguesa", "nachos", "papas",  
              "coca", "fanta",  
              "Escazu", "200m Este de Zareto.",  
              "Sólo se permiten burbujas sociales.",  
              40]).
```

```
restaurante([ "NachosCR",  
              "mexicana",  
              "nachos", "carne", "tacos",  
              "coca",  
              "chepe", "200m Este del museo de los niños.",  
              "Lleva alcohol en gel.",  
              30]).
```

## 1.3 - Descripción detallada del algoritmo general de solución.

El algoritmo es bastante sencillo.



En caso de que la primera vez encuentre solamente una coincidencia entonces no avanzará a la siguiente pregunta.

## 1.4 - Problemas sin solución.

### Conversación fluida.

El SE está programado para hacer dos preguntas máximo, lo que resulta en una conversación muy tiesa. Además, por su falta de eficacia para analizar las palabras clave de una oración es que cuesta mucho obtener resultados serios.



?- restauranTEC.

¡Hola! Espero que te encuentres súper bien

¿Qué te gustaría comer?

|: pizza.

¿Qué sabor de pizza?|: pepperoni.

Ninguna palabra coincide con las del diccionario.¿Qué sabor de pizza?|: jamon.

Ninguna palabra coincide con las del diccionario.Ninguna palabra coincide con false.

**Actualización tras la prórroga:** el sistema sí simula una conversación relativamente fluida sólo que cortará en el momento que obtenga una coincidencia con la base de datos.

## **Análisis de palabras clave.**

Cuando nos dimos cuenta del error de almacenar toda la información de los restaurantes dentro de listas, creamos una base de datos que categoriza ciertos valores por tipo y les asociamos una pregunta para realizar en caso de ser seleccionada.

Esto resultó bastante positivo ya que permitió el avance que tenemos pero no fue suficiente para solucionar todos los demás problemas como encontrar el resultado final o tener una conversación fluida.

**Actualización tras la prórroga:** sí resultó útil esta sub-base de datos ya que fue clave para facilitar la sugerencia final del sistema al usuario. Sin mencionar que el algoritmo de selección de selección de la siguiente pregunta no hubiera sido posible sin este manejo de los datos. (Ver la regla tipoValor ).

Aún así, el sistema no es capaz de detectar dos palabras clave en una misma línea por lo que continuará con el logoritmo a partir de la primer palabra que identifica.

?- restauranTEC.

¡Hola! Espero que te encuentres súper bien

¿Qué te gustaría comer?

|: una pizza de jamon y queso

¿Qué sabor de pizza?

|: jamon y queso

Te sugerimos que vayas a: Bella Italia

Dirección: Curridabat

300m Sur de la entrada principal de la Universidad de Costa Rica

Gracias por usar RestaurantTEC, ¡Lindo día!

## 1.5 - Plan de Actividades.

Tarea	Descripción	Tiempo estimado	Encargado
Investigación introductoria	Investigar sobre análisis de palabras clave en Prolog y sistemas expertos.	Uno o dos días.	Ambos
Elaboración de la base de datos	Tomar los restaurantes de ejemplo y montar una base de datos en Prolog.	2 h	Santiago
Mejorar la base de datos	Agregar bebidas y ampliar la gama de restaurantes.	2h	Tomás
Análisis de palabras clave	Escribir la lógica para el análisis de coincidencias entre una lista de palabras y la base de datos de restaurantes.	1 día	Santiago
Interfaz	Montar una interfaz de diálogo entre el usuario y el SE en la consola de Prolog.	1 día	Tomás
Acople	Implementar la base de datos con la interfaz y crear una conversación fluida con el usuario	1 día	Santiago.
Documentación	Escribir la documentación interna y externa del programa.	medio día	Tomás.

## 1.6 - Problemas solucionados.

**Utilizar la consola como interfaz con el usuario.**

Hicimos correcto uso para comunicar al usuario con el programa de una manera amigable, a pesar de la falta de fluidez y los muchos errores que presenta el programa. La conversación se siente relativamente natural mas no se descarta la percepción de estar comunicandose con un bot.

## **Montar una base de datos.**

Montamos diferentes estilos de bases de datos como:

1. A partir de listas.
2. A partir de hechos y reglas.

## **Sugerir opciones de restaurantes.**

Sí fue posible diseñar un sistema capaz de analizar las entradas de un usuario y retornarle un restaurante que cumpla con las características definidas.

# **1.7 - Conclusiones y Recomendaciones del proyecto.**

---

## **Conclusiones.**

1. Prolog tiene una curva de aprendizaje que subestimamos. Nos costó más tiempo de lo planeamos por lo que al final debimos pedir una prórroga para la entrega.
2. Existen diferentes maneras de acomodar datos en datos. Haber usado listas y no los hechos y reglas que son los tipos de dato especializados de Prolog, resultó un causante de fricción innecesaria para el desarrollo del proyecto.

## **Recomendaciones.**

1. Dedicar más tiempo al aprendizaje de las nuevas tecnologías y no subestimarlas.
2. Considerar las características especializadas de un lenguaje como primera instancia para resolver un algoritmo ya que es en lo que el lenguaje puede comportarse de mejor y más sencilla manera.

## 2 - Bitácoras.

---

### 2.1 - Bitácora de reuniones.

7 de octubre.

Nos reunimos para leer la tarea y planear. Decidimos investigar un poco por nuestra cuenta y repartir tareas la próxima reunión.

9 de octubre.

No teníamos idea de cómo repartir ideas pero decidimos que Santiago trabajaría en el análisis de las entradas y palabras mientras que Tomás comenzaría a montar una pequeña interfaz.

12 de octubre.

Tenemos la base de datos. Agregamos algunos valores como bebidas y un par de restaurantes extra. Nos ha costado que la conversación sea dinámica y natural.

14 de octubre.

Estamos apurados. Aún nos falta mejorar la fluidez de la conversación y terminar el filtro para que retorne los restaurantes correctos. Sin mencionar la documentación 🙄.

### 2.2 - Bitácora de Santiago.

8 de octubre.

Se realizó la investigación para conocer más sobre los aspectos de un sistema experto. Lo importante es entender sobre el análisis sintáctico.

11 de octubre.

He construido una pequeña base de datos de restaurantes y he realizado pruebas con Prolog para conocer mejor cómo manejar listas.

14 de octubre.

Hoy se culminó el proyecto. Realizamos la documentación.

### 2.3 - Bitácora de Tomás.

7 de octubre.

Prolog. Un lenguaje interesante, me recuerda mucho matemática discreta. He investigado un poco pero todavía no entiendo cómo hacer el análisis de palabras clave.

12 de octubre.

Amplie la base de datos para agregar bebidas y otros restaurantes. También, junto a Santiago definimos un modelo de análisis palabra por tipo de valor que debemos implementar aún.

14 de octubre.

Hemos corrido bastante. Me siento estresado porque nos hace falta la documentación y mejorar la conversación del SE.

### 3 - Referencias.

---

- Chaudhry Talha, [Tutorial: Learn Prolog Language by Creating an Expert System](#).
- Documentación de [Prolog](#).
- Especificación de la tarea.