



Inteligencia Artificial 1

Practica 2

Manual Usuario

Juan Antonio Solares Samayoa
Carnet 201800496
Primer Semestre 2023

Objetivos	3
Generales	3
Específicos	3
Descripción de la solución	5
Cantidad de recursos que utiliza el sistema experto	5
Archivos del sistema	5
Requerimientos del sistema	5
Código de la solución	5
Menú principal	5
Sistema experto	5
Reportes	8
Opinión de la solución	18
Diagrama de encadenamiento	19
Simbología Utilizada	19
Explicación de diagrama de encadenamiento	21
Conclusiones	24

Objetivos

Generales

- Familiarizarse con los conceptos básicos de la programación lógica.
- Aplicar los conceptos básicos del lenguaje de programación prolog
- Comprender cómo funcionan las bases de conocimiento y aplicarlas por medio de programación lógica.
- Crear un sistema experto como paso a una inteligencia artificial funcional

Específicos

- Se buscará comprender los fundamentos teóricos de la programación lógica, como la lógica de primer orden, la resolución de predicados y la unificación. Además, se explorarán los principales conceptos relacionados, como las reglas, los hechos y las consultas en Prolog.
- Adquirir habilidades prácticas en la programación en Prolog. Se trabajará en la creación de programas más complejos utilizando los constructos y la sintaxis del lenguaje. Se practicará la definición de predicados, la manipulación de listas y la recursión. Se resolverán problemas prácticos utilizando Prolog, como búsquedas en árboles, manipulación de bases de conocimiento.
- Comprender el concepto de una base de conocimiento en Prolog y cómo se utiliza para representar información y conocimientos. Se explorarán diferentes técnicas para la creación, modificación y consulta de bases de conocimiento. Se aprenderá a definir hechos y reglas que representen información y relaciones entre entidades. Se buscará utilizar bases de conocimiento existentes o crear nuevas bases de conocimiento para resolver problemas específicos mediante programación lógica.
- Diseñar y desarrollar un sistema experto utilizando Prolog. Se explorarán los conceptos y técnicas necesarias para la construcción de sistemas expertos, como la representación del conocimiento, la inferencia basada en reglas y el razonamiento. Se buscará implementar un sistema que pueda realizar consultas y brindar respuestas y recomendaciones basadas en su base de conocimiento. Se aplicarán técnicas de adquisición de conocimiento y se evaluará la eficacia y eficiencia del sistema experto desarrollado.

Descripción de la solución

Cantidad de recursos que utiliza el sistema experto

Archivos del sistema

- conocimientos.pl: Este archivo contiene los datos relacionados con clientes, hoteles, departamentos, registros y empleados.
- reportes.pl: Este archivo contiene el código para la generación de reportes.

Requerimientos del sistema

- Sistema operativo: Windows 10, Mac OS o Linux
- Consola SWI-Prolog
- Memoria RAM de 2 GB
- Procesadores Intel Celeron o superior

Código de la solución

Menú principal

```
Welcome to the expert Hotel System

Type the selection criteria that you need:
  1. Search by budget
    2. Search by language
    3. Search by stars
    4. Search by weather
    5. Reports:
    >> Select an option
```

En esta sección el usuario podrá elegir los criterios en los cuales quiere buscar la mejor solución para hospedarse.

Sistema experto

El sistema experto de búsqueda de hoteles utilizará los siguientes criterios para realizar la búsqueda que mejor se adapte a los parámetros seleccionados.

- Por presupuesto

Para poder mostrar una mejor experiencia a los potenciales clientes de los hoteles, se dividirá el sistema en análisis por presupuesto caro o barato, como lo muestra el siguiente fragmento de código.

```
% ? ===== check budget =====
budget_separator(BUDGET):-(
    BUDGET =< 3000 -> cheap(BUDGET);
    BUDGET >= 6000 -> vip(BUDGET)
).
```

- Por idioma

```
%? ===== check language =====
language_separator(LANGUAGE):-(
    LANGUAGE == 'espanol' -> hotel_language_selection(LANGUAGE);
    LANGUAGE == 'katchikel' -> hotel_language_selection(LANGUAGE);
    LANGUAGE == 'ingles' -> hotel_language_selection(LANGUAGE)
).
```

- Por cantidad de estrellas

```
rating_separator(RATING):-(
    RATING == 1 -> hotel_rating_selection(RATING);
    RATING == 2 -> hotel_rating_selection(RATING);
    RATING == 3 -> hotel_rating_selection(RATING);
    RATING == 4 -> hotel_rating_selection(RATING);
    RATING == 5 -> hotel_rating_selection(RATING)
).
```

- Por clima

```
weather_separator(WEATHER):-(
    WEATHER == 'tropical' -> hotel_weather_selection(WEATHER);
    WEATHER == 'calor' -> hotel_weather_selection(WEATHER);
    WEATHER == 'frio' -> hotel_weather_selection(WEATHER);
    WEATHER == 'templado' -> hotel_weather_selection(WEATHER)
).
```

Una vez se elige en qué parámetro se desean generar las sugerencias, el usuario tiene que agregar datos adicionales referentes a la estadía de hotel, calificaciones del hotel, como en el código de abajo.

```
% ? ===== economic plan
cheap(BUDGET):-write('You are in the economic plan'), nl,
write('Type the maximun price that you are willing to pay'), nl,
read(TICKET),
write('Type your favorite weather'), nl,
read(WEATHER),
write('Type the opinion that you want to search'), nl,
read(OPINION),
write('What kind of room do you want? (simple / double)'), nl,
read(ROOMTYPE),
write('Type the maximun distance that are you willing to travel in your trip'),nl,
read(DISTANCE),
write('Type the maximun price that you are willing to pay in each meal'), nl,
read(FOOD),
write('Type the maximun price that you are willing to pay for one room'), nl,
read(ROOM),
write('How many days do you want to stay?'), nl,
read(NDAYS),
write('Type the minimun valuation of a hotel'), nl,
read(RATE),
write('Do you have a vehicle? y/n'), nl,
read(VEHICLE),
```

Una vez se definen los datos relacionados a la estadía del tipo de hotel deseado,

Reportes

Reporte 1

```
% 1. Nombre y nacionalidad de clientes nacidos en Europa con opiniones mayores de 5
report1:-
write(' ===== Report # 1 ===== '), nl,
  cliente(Id_customer, Customer_name, Customer_lastname, Customer_country, _, _, _),
  registro(_, Id_customer, _, _, _, Customer_Opinion),

  Customer_Opinion > 5,
  (
    Customer_country=='italia';
    Customer_country=='holanda';
    Customer_country=='italia';
    Customer_country=='espanol';
    Customer_country=='inglaterra';
    Customer_country=='espanola'
  ),

  show_rep1(Customer_name, Customer_lastname, Customer_country),
  menu_reports.
show_rep1(Customer_name, Customer_lastname, Customer_country):-
write(' >> Result:'), nl,
  format('
    Name: ~a
    Lastname: ~a
    Country: ~a
  ', [Customer_name, Customer_lastname, Customer_country]), nl, fail, true.
```

El objetivo del informe es mostrar información sobre clientes que cumplen con ciertas condiciones. El código comienza imprimiendo un encabezado para el informe y luego busca clientes que cumplan las siguientes condiciones: tener una opinión del cliente mayor a 5 y pertenecer a uno de los países específicos (Italia, Holanda, España o Inglaterra).

Una vez encontrados los clientes que cumplen con estas condiciones, se muestra un resultado en forma de un bloque de información que incluye el nombre, apellido y país del cliente. Luego se llama a `menu_reports`, que probablemente sea parte de un menú de opciones para generar otros informes o realizar otras acciones.

En la definición de `show_rep1`, se muestra el formato en el que se imprimirá la información de cada cliente que cumple con las condiciones. Se utiliza el predicado `format` para imprimir los datos del cliente en un formato específico. Luego, se utiliza `fail` para forzar el fallo de la regla y permitir que el programa continúe buscando más soluciones que cumplan con los criterios del informe. Finalmente, se utiliza `true` para indicar que se ha alcanzado un punto de éxito y permitir que el programa continúe ejecutando otras acciones.

Reporte 2

```
% 2. Nombre y estado civil de clientes con reservaciones en hoteles con más de 4 estrellas
report2:-
    write(' ===== Report # 2 ===== '), nl,
    cliente(Id_customer, Customer_name, Customer_lastname, _, _, Customer_status, _),
    registro(_, Id_customer, Id_hotel, _, _, _),
    hotel(Id_hotel, _, _, N_stars, _, _, _),
    N_stars > 4,
    show_rep2(Customer_name, Customer_lastname, Customer_status).

show_rep2(Customer_name, Customer_lastname, Customer_status):-
    write(' >> Result:'), nl,
    format('
        ~a
        ~a
        ~a
    ', [Customer_name, Customer_lastname, Customer_status]), nl, fail, true.
```

Este código en Prolog está diseñado para generar un informe que muestra el nombre y estado civil de los clientes que tienen reservaciones en hoteles con más de 4 estrellas. El informe se construye utilizando una serie de consultas y coincidencias en la base de conocimientos.

El predicado `report2` se encarga de iniciar el proceso del informe. Primero, utiliza el predicado `cliente` para obtener información sobre los clientes, como su identificador, nombre, apellido y estado civil. Luego, utiliza el predicado `registro` para obtener información sobre las reservaciones de los clientes, como el identificador del cliente y el identificador del hotel. A continuación, utiliza el predicado `hotel` para obtener información sobre los hoteles, como el número de estrellas. Luego, se aplica una condición de filtro donde se verifica si el número de estrellas del hotel es mayor a 4.

Después de aplicar el filtro, se llama al predicado `show_rep2` para mostrar los resultados obtenidos. El predicado `show_rep2` se encarga de imprimir en pantalla el nombre, apellido y

estado civil del cliente encontrado. Finalmente, se utiliza el fail para forzar el retroceso y buscar más resultados que cumplan con los criterios establecidos, hasta que no haya más resultados y el informe se complete.

Reporte 3

```
% 3. Nombre de Administradores en hoteles con valoraciones mayor o igual que 5
report3:-
    write(' ===== Report # 3 ===== '), nl,
    trabajador(_, Employee_name, Employee_occupation, Hotel_id),
    registro(_,_,Hotel_id,_,_,Hotel_opinion),

    Employee_occupation == 'Administrador',
    Hotel_opinion >= 5,

    show_rep3(Employee_name, Employee_occupation, Hotel_id).
show_rep3(Employee_name, Employee_occupation, Hotel_id):-
    write(' >> Result: '), nl,
    format('
        Name: ~a
        Occupation: ~a
        Hotelid: ~a
    ', [Employee_name, Employee_occupation, Hotel_id]),
    format('~\n~t~45|\n~n'),
    nl, fail, true.
```

Este código en Prolog tiene como objetivo generar un informe que muestra los nombres de los administradores de hoteles que tienen valoraciones igual o mayor a 5.

El predicado report inicia el informe y muestra un encabezado. Luego, utiliza el predicado trabajador para obtener información de los trabajadores, donde se busca el nombre, ocupación y ID del hotel. Luego, se utiliza el predicado registro/6 para obtener información de los registros de opiniones de los hoteles, donde se busca el ID del hotel y la opinión.

Después se aplican una serie de condiciones para filtrar los resultados. Primero, se verifica que la ocupación del empleado sea "Administrador" utilizando la igualdad ==. Luego, se verifica que la opinión del hotel sea mayor o igual a 5.

Una vez que se cumplan estas condiciones, se llama al predicado show_rep para mostrar los resultados. Este predicado muestra el nombre, ocupación y ID del hotel del administrador encontrado. Luego, se utiliza el predicado format/2 para imprimir los resultados de manera formateada.

Reporte 4

```

% 4. Departamentos y hotel con más reservaciones, en clima calor

% Reglas para contar las reservaciones de cada departamento en clima calor
count_reservations(Departamento, Count) :-
    departamento(Departamento, _, _, Calor, _),
    findall(Registro, registro(_, _, Hotel, _, _, _), Registros),
    count_reservations_helper(Departamento, Registros, Count).

count_reservations_helper(_, [], 0).
count_reservations_helper(Departamento, [registro(_, _, Hotel, _, _, _) | Registros], Count) :-
    hotel(Hotel, _, _, _, _, Departamento, _),
    count_reservations_helper(Departamento, Registros, Count1),
    Count is Count1 + 1.
count_reservations_helper(Departamento, [_ | Registros], Count) :-
    count_reservations_helper(Departamento, Registros, Count).

% Regla para obtener el departamento con más reservaciones en clima calor
departamento_mas_reservaciones(Departamento, Count) :-
    departamento(_, Departamento, _, _, Calor, _),
    findall(Count-Departamento, count_reservations(Departamento, Count), Pares),
    max_list(Pares, Count-Departamento),
    format('Departamento con más reservaciones en clima calor: ~w (Reservaciones: ~w)~n', [Departamento, Count]).

% Regla para obtener el hotel con más reservaciones en clima calor
hotel_mas_reservaciones(Hotel, Count) :-
    hotel(Hotel, _, _, _, _, Departamento, _),
    count_reservations(Departamento, Count),
    findall(Count-Hotel, (hotel(Hotel, _, _, _, _, D, _), departamento(D, _, _, _, calor, _)), Pares),
    max_list(Pares, Count-Hotel),
    format('Hotel con más reservaciones en clima calor: ~w (Reservaciones: ~w)~n', [Hotel, Count]).

% Departamentos y hotel con más reservaciones, en clima calor
report4 :-
    write(' ===== Report # 4 ===== '), nl,
    findall(Reservations, registro(_,_,_,_,Reservations), ReservationsList),
    max_list(ReservationsList, MaxReservations),
    registro(_,_,Id_hotel,_,_,_),
    hotel(Id_hotel,Hotel_name,_,_,_,_,_),
    departamento(Id_department,Department_name,_,_,DepWeather,_),
    DepWeather == 'calor',
    show_rep4(Hotel_name, Department_name).

show_rep4(Hotel_name, Department_name) :-
    write(' >> Result:'), nl,
    format('Hotel: ~a~nDepartment: ~a~n', [Hotel_name, Department_name]), nl, fail, true.

```

Este código en Prolog tiene como objetivo encontrar el departamento y el hotel con más reservaciones en clima calor. Para lograr esto, se definen varias reglas. La regla `count_reservations` cuenta las reservaciones para un departamento específico en clima calor. Utiliza la base de conocimiento que contiene información sobre los departamentos y los registros de reservaciones en hoteles. La regla `count_reservations_helper` es un

predicado auxiliar que recorre la lista de registros y verifica si pertenecen al departamento en cuestión, incrementando un contador.

La regla departamento_mas_reservaciones busca el departamento con más reservaciones en clima calor. Utiliza la regla count_reservations para obtener el conteo de reservaciones de cada departamento en clima calor y luego encuentra el máximo entre ellos utilizando max_list.

La regla hotel_mas_reservaciones busca el hotel con más reservaciones en clima calor. Utiliza la regla count_reservations para obtener el conteo de reservaciones del departamento al que pertenece el hotel y luego encuentra el máximo entre ellos.

La regla report es el punto de entrada principal del programa. Muestra un encabezado y luego busca el máximo número de reservaciones en la base de conocimiento de registros. A partir de esto, se obtiene el nombre del hotel y el nombre del departamento que tienen ese número máximo de reservaciones en clima calor.

Reporte 5

```
% 5. Nombre de clientes extranjeros hospedados en departamentos de habla español
report5 :-
    write(' ===== Report # 5 ===== '), nl,
    cliente(Id_customer, Customer_name, Customer_lastname, Costumer_country, _, _, _),
    departamento(_, DepName, _, Deplang, _, _),
    registro(_, Id_customer, HotelId, _, _, _),
    hotel(HotelId, _, _, _, _, _, _),

    Deplang == 'espanol',
    Costumer_country \= 'guatemala',

    show_rep5(Customer_name, Customer_lastname, DepName).
show_rep5(Customer_name, Customer_lastname, DepName):-
    write(' >> Result: '), nl,
    format('
        Name: ~a
        Lastname: ~a
    ', [Customer_name, Customer_lastname]), nl, fail, true.
```

Este código en Prolog implementa un reporte que busca mostrar los nombres de los clientes extranjeros que se encuentran hospedados en departamentos donde se habla español. El reporte recorre diferentes predicados y realiza verificaciones para obtener los resultados deseados.

El predicado principal "report5" inicia imprimiendo un encabezado para el reporte. Luego, utiliza el predicado "cliente" para obtener información sobre los clientes, como su identificación, nombre, apellido y país. También utiliza el predicado "departamento" para obtener información sobre los departamentos, como el nombre y el idioma que se habla en ellos. A través del predicado "registro", se obtiene información sobre los registros de hospedaje, como el identificador del cliente y del hotel. Finalmente, el predicado "hotel" proporciona información detallada sobre el hotel en el que se hospeda cada cliente.

Se aplican condiciones específicas en las variables obtenidas para filtrar los resultados. Se verifica que el idioma del departamento sea español y que el país del cliente no sea Guatemala. Si se cumplen estas condiciones, se llama al predicado "show_rep5" para mostrar el nombre y apellido del cliente, así como el nombre del departamento en el que se encuentra hospedado.

El predicado "show_rep5" simplemente imprime en pantalla los datos del cliente obtenidos anteriormente.

En resumen, este código busca generar un reporte que muestre los nombres de los clientes extranjeros hospedados en departamentos de habla española, excluyendo a los clientes de Guatemala.

Reporte 6

```
% 6. Nombre de Hotel, departamento, idioma y Nombre de clientes con opiniones
% mayores o igual que 7 y estadías mayores o igual a 3 días

report6 :-
    write(' ===== Report # 6 ===== '), nl,
    cliente(IdCustomer, CustomerName, CustomerLastName,_,_,_),
    departamento(DepId, DepName, _,DepLang, _,_),
    hotel(IdHotel, HotelName, _, NStars, _,_,DepId,_),
    registro(_, IdCustomer, IdHotel, _,NDays,Opinion),

    Opinion >= 7,
    NDays >= 3,

    show_rep6(HotelName, DepName, DepLang, CustomerName, CustomerLastName).
show_rep6(HotelName, DepName, DepLang, CustomerName, CustomerLastName):-
    write(' >> Result '), nl,
    format('
        HotelName: ~a
        DepName: ~a
        Language: ~a
        Name: ~a
        Lastname: ~a
    ', [HotelName, DepName, DepLang, CustomerName, CustomerLastName]), nl, fail, true.
```

Este código en Prolog representa un reporte que muestra información sobre hoteles, departamentos, clientes y opiniones. El objetivo del reporte es encontrar clientes que hayan dejado opiniones con calificaciones iguales o superiores a 7 y hayan tenido estadías de al menos 3 días en un hotel.

El código realiza consultas a diferentes hechos y utiliza reglas para obtener la información necesaria. Primero, se obtiene información de los clientes, departamentos, hoteles y registros de estadías. Luego, se aplican las condiciones de opinión y duración de estadía para filtrar los resultados. Finalmente, se muestra la información relevante, como el nombre del hotel, el departamento, el idioma, el nombre y el apellido del cliente.

El código utiliza la regla show_rep para imprimir los resultados obtenidos en un formato legible. Cada vez que se encuentra un resultado válido, se imprime en pantalla la información correspondiente. El proceso se repite hasta que se hayan mostrado todos los resultados válidos.

Reporte 7

```
% 7. Nombre País y Nombre de Hotel de clientes extranjeros hospedados en
% departamentos de habla inglés, con menos de 2 días de hospedaje
report7 :-
    write(' ===== Report # 7 ===== '), nl,
    cliente(IdCustomer, CustomerName, CustomerLastName, CustomerCountry, _, _, _),
    departamento(DepId, DepName, _, DepLang, _, _),
    hotel(IdHotel, HotelName, _, NStars, _, _, DepId, _),
    registro(_, IdCustomer, IdHotel, _, NDays, _),

    CustomerCountry \= 'guatemala',
    DepLang == 'ingles',
    NDays <= 2,

    show_rep7(CustomerName, CustomerCountry, HotelName, DepName).
show_rep7(CustomerName, CustomerCountry, HotelName, DepName) :-
    write(' >> Result '), nl,
    format('
        CustomerName: ~a
        CustomerCountry: ~a
        HotelName: ~a
        DepName: ~a
    ', [CustomerName, CustomerCountry, HotelName, DepName]), nl, fail, true.
```

Este código en Prolog implementa un informe o reporte específico, denominado Reporte #7, que busca mostrar información sobre clientes extranjeros hospedados en departamentos de habla inglesa, con menos de 2 días de estadía. El informe muestra el nombre del cliente, su país de origen, el nombre del hotel donde se encuentra hospedado y el nombre del departamento donde se ubica el hotel.

El código utiliza consultas y reglas para obtener la información necesaria. Primero, recorre la base de conocimiento para encontrar clientes (representados por el predicado cliente/7), departamentos (predicado departamento/6) y hoteles (predicado hotel/9). Luego, busca registros de hospedaje (predicado registro/5) que cumplan con las condiciones establecidas: el cliente no debe ser de Guatemala, el departamento debe ser de habla inglesa y la estadía del cliente debe ser de menos de 2 días.

Cuando se encuentra un registro que cumple con estas condiciones, se llama al predicado `show_rep7/4` para mostrar los detalles del cliente, incluyendo su nombre, país de origen, nombre del hotel y nombre del departamento. El resultado se imprime en el formato especificado.

Reporte 8

```
% 8. nacionalidad de clientes que reservaron en peten
report8:-
    write(' ===== Report # 8 ===== '), nl,
    hotel(IdHotel, HotelName, _, NStars, _,_,_,DepId,_),
    cliente(IdCustomer, CustomerName, CustomerLastName, CustomerCountry, _,_,_),
    departamento(DepId, DepName, _, DepLang, _,_),
    registro(_, IdCustomer, IdHotel, _, NDays, _),

    IdCustomer == IdCustomer,
    DepId==DepId,

    DepName == 'peten',

    show_rep8(CustomerCountry).
show_rep8(CustomerCountry):-
    write(' >> Result '), nl,
    format('
        CustomerCountry: ~a
    ', [CustomerCountry]), nl, fail, true.
```

Este código en Prolog muestra un reporte que busca obtener la nacionalidad de los clientes que han realizado reservas en hoteles ubicados en el departamento de Petén.

En primer lugar, se muestra el encabezado del reporte. Luego, se realizan consultas a la base de conocimiento para obtener información sobre los hoteles, los clientes, los departamentos y los registros de reservas. Se vinculan las variables correspondientes para realizar las consultas adecuadas.

Se establecen las condiciones para que los registros de reserva cumplan con ciertas características: el departamento debe ser "peten". Luego, se llama a un predicado auxiliar llamado "show_rep8" que mostrará la nacionalidad del cliente obtenida en la consulta.

El predicado "show_rep8" imprime la nacionalidad del cliente en el formato deseado. Se utiliza el predicado "format" para mostrar el resultado de manera legible. Después de imprimir la información, se realiza un "fail" para forzar la retrotracción y buscar más soluciones posibles. Finalmente, se finaliza la consulta con "true".

Reporte 9

```
% 9. Nombre de Hotel y dirección de hoteles que recibieron a clientes casados, que
% tengan opiniones mayores de 6 por personas con mínimo de 3 días de estadía
report9:-
    write(' ===== Report # 9 ===== '), nl,
    hotel(IdHotel, HotelName, HotelAddress, NStars, _,_,_,DepId,_),
    cliente(IdCustomer,_,_,_,Civilstate,_),
    departamento(DepId, DepName, _,DepLang, _,_),
    registro(_, IdCustomer, IdHotel, _,NDays,Opinion),

    Civilstate=='casado',
    NDays >= 3,
    Opinion > 6,

    show_rep9(HotelName, HotelAddress).

show_rep9(HotelName, HotelAddress) :-
    write(' >> Result '), nl,
    format('
        HotelName: ~a
        HotelAddress: ~a
    ', [HotelName, HotelAddress]), nl, fail, true.
```

You, 16 hours ago • report

Opinión de la solución

El desarrollo de un sistema experto de búsqueda de hoteles utilizando Prolog es una solución prometedora que cumple con una serie de objetivos importantes en el campo de la programación lógica y la inteligencia artificial.

En primer lugar, la comprensión de los fundamentos teóricos de la programación lógica, como la lógica de primer orden, la resolución de predicados y la unificación, es esencial para construir un sistema experto eficaz. Estos conceptos teóricos permiten modelar de manera adecuada el conocimiento y las relaciones entre los diferentes elementos involucrados en la búsqueda de hoteles.

Además, adquirir habilidades prácticas en la programación en Prolog es crucial para desarrollar programas complejos y funcionales. La capacidad de definir predicados, manipular listas y utilizar la recursión en Prolog proporciona las herramientas necesarias para implementar algoritmos de búsqueda eficientes y lógica de inferencia sólida.

El concepto de una base de conocimiento en Prolog resulta fundamental para representar la información y los conocimientos necesarios en un sistema experto de búsqueda de hoteles. La capacidad de crear, modificar y consultar bases de conocimiento permite almacenar hechos y reglas que describen las características de los hoteles, las preferencias del usuario y las relaciones entre ellos. Esto permite una personalización y adaptación adecuada del sistema a las necesidades individuales de los usuarios.

En cuanto al diseño y desarrollo del sistema experto en Prolog, es esencial comprender y aplicar técnicas de representación del conocimiento, inferencia basada en reglas y razonamiento. Estos conceptos permiten al sistema realizar consultas y brindar respuestas y recomendaciones basadas en la base de conocimiento. La capacidad de adquirir conocimiento y evaluar la eficacia y eficiencia del sistema experto garantiza un desarrollo sólido y la mejora continua del sistema a lo largo del tiempo.

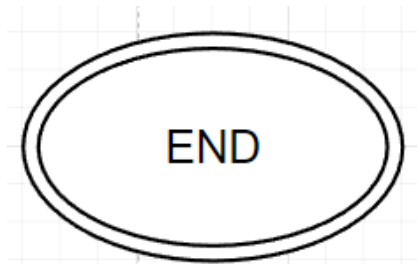
Diagrama de encadenamiento

Simbología Utilizada

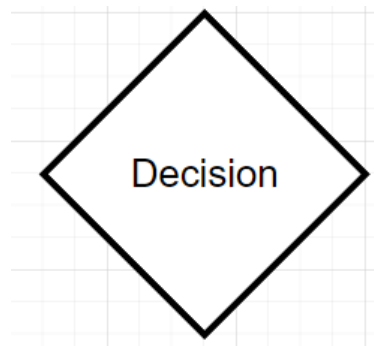
1. Componente de inicio



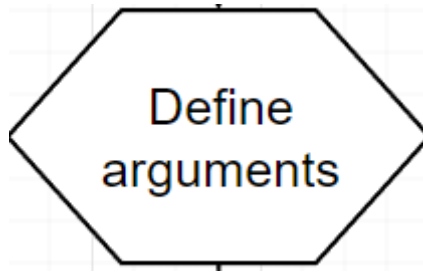
2. Componente de final



3. Componente de decisión (menú)




4. Componente de definición de argumentos



5. Componente de proceso

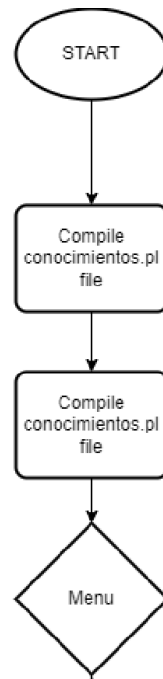


Enlace al diagrama de encadenamiento completo:

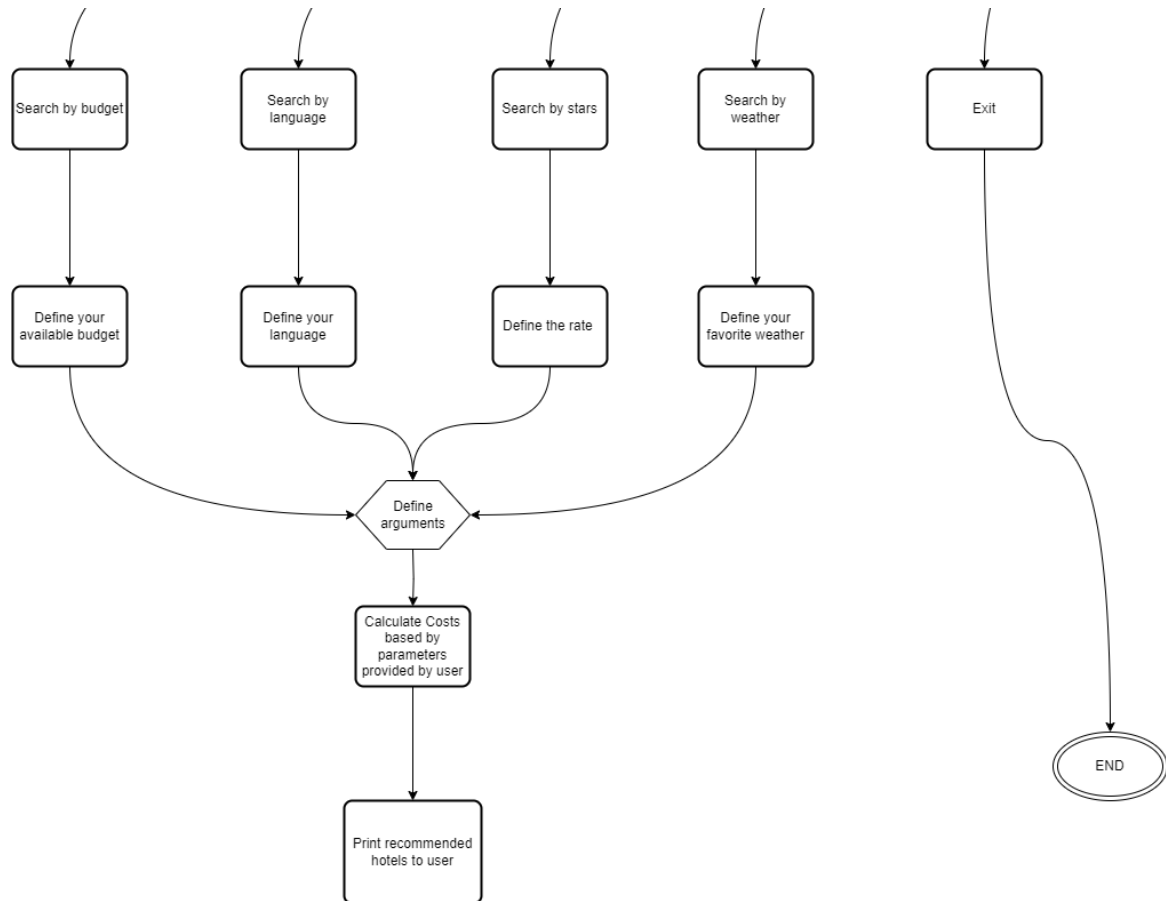
 DiagramaEncadenamiento.png

Explicación de diagrama de encadenamiento

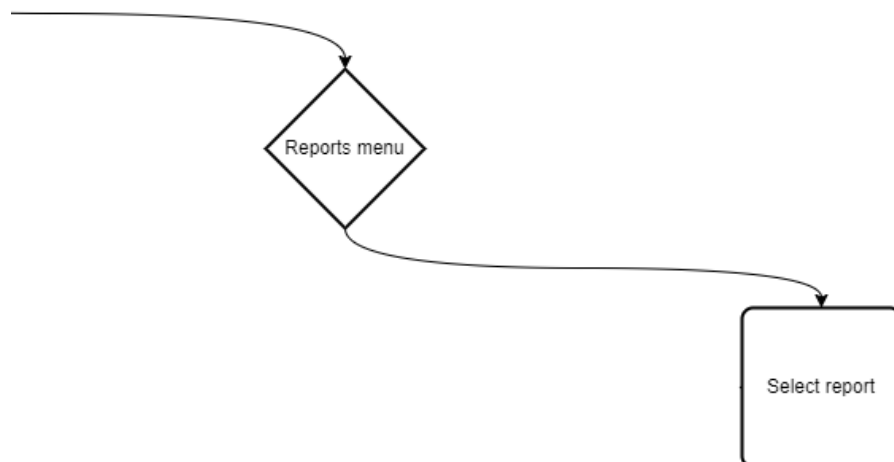
- Flujo inicial del programa: en este flujo, se definen los datos que conformarán la base de conocimientos con los que cuenta la aplicación de prolog y posteriormente se harán las consultas y reportes.

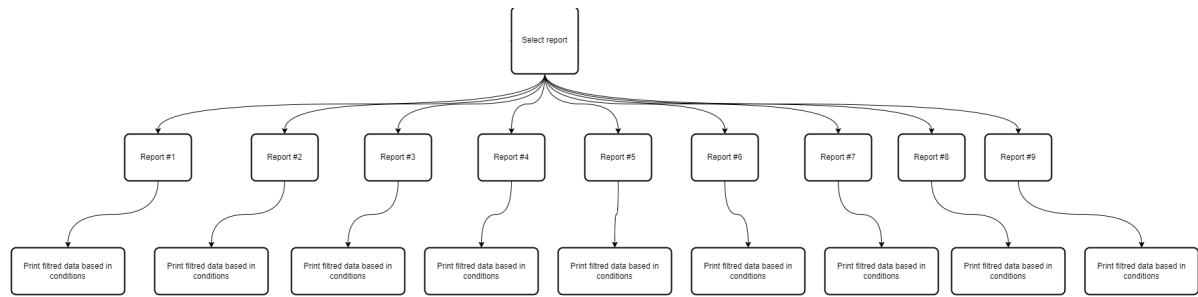


- Opciones para generar recomendaciones: El usuario por medio del menú puede generar sugerencias de hoteles basándose en los siguientes parámetros:
 - Presupuesto
 - Lenguaje
 - Cantidad de estrellas del hotel
 - Clima



- Menú de reportes: en esta sección el usuario podrá generar distintos tipos de reportes utilizando los datos de la base de conocimientos





Conclusiones

- Al lograr comprender los fundamentos teóricos de la programación lógica, como la lógica de primer orden, la resolución de predicados y la unificación, se adquiere una base sólida para el desarrollo de sistemas expertos en Prolog. Además, explorar conceptos como las reglas, los hechos y las consultas en Prolog permite comprender cómo se estructura y opera en este lenguaje. Esta comprensión proporciona las bases teóricas necesarias para desarrollar un sistema experto de búsqueda de hoteles eficiente y funcional.
- La adquisición de habilidades prácticas en la programación en Prolog es esencial para poder crear programas más complejos y funcionales. Al trabajar en la definición de predicados, la manipulación de listas y la recursión, se desarrolla la capacidad de implementar algoritmos y soluciones eficientes en Prolog. Resolver problemas prácticos utilizando Prolog, como búsquedas en árboles o manipulación de bases de conocimiento, brinda la oportunidad de aplicar estos conocimientos en escenarios reales y fortalecer las habilidades de programación en este lenguaje.
- La comprensión del concepto de una base de conocimiento en Prolog es fundamental para representar información y conocimientos en un sistema experto de búsqueda de hoteles. Al explorar diferentes técnicas para la creación, modificación y consulta de bases de conocimiento, se adquiere la capacidad de utilizar eficientemente esta herramienta. Aprender a definir hechos y reglas que representen información y relaciones entre entidades permite construir una base de conocimiento sólida y adaptada a las necesidades específicas del sistema experto. Tanto el uso de bases de conocimiento existentes como la creación de nuevas bases de conocimiento amplían las posibilidades de resolver problemas específicos mediante programación lógica.
- Al explorar los conceptos y técnicas necesarias para la construcción de sistemas expertos, como la representación del conocimiento, la inferencia basada en reglas y el razonamiento, se sientan las bases para el diseño y desarrollo de un sistema experto de búsqueda de hoteles en Prolog. La implementación de un sistema capaz de realizar consultas, brindar respuestas y recomendaciones basadas en su base de conocimiento demuestra la aplicación práctica de los conocimientos adquiridos. La aplicación de técnicas de adquisición de conocimiento y la evaluación de la eficacia y eficiencia del sistema experto permiten mejorar y perfeccionar continuamente su desempeño.