Desarrollo de un SBC - Recomendador de recetas

Entrega 2 - Práctica 6

Asignatura: Ingeniería del Conocimiento
Curso académico: 2024/2025
Grupo de Prácticas: 1 (Viernes)
Email: mariacribilles@correo.ugr.es

María Cribillés Pérez
DNI: 75573410X

Índice

1. Resumen del funcionamiento del sistema

Hemos implementado un sistema basado en conocimiento que responde a la pregunta "¿Qué cocino hoy?". Está diseñado para recomendar recetas de forma personalizada según las preferencias y restricciones del usuario. El sistema se estructura de forma modular en cinco módulos principales: main, pedir, deducir, compatibles y proponer, cada uno de los cuales se encarga de una parte específica del proceso de recomendación.

El funcionamiento comienza con una propuesta inicial por defecto, basada en una receta genérica que puede ser de gusto de mucha gente. En todo momento, el sistema le pregunta al usuario si la receta enseñada es de su gusto, si no lo es, pregunta información para recomendar una receta más adecuada al usuario. Por tanto, si el usuario rechaza la receta por defecto, el sistema interactúa con el usuario para recopilar información relevante, como el tipo de plato que busca (por ejemplo, postre o plato principal) y posibles propiedades especiales (como vegana, sin gluten o sin lactosa).

Una vez recogidas las preferencias, se deducen internamente las restricciones aplicables, y se filtran las recetas compatibles con dichas condiciones. Las recetas candidatas son evaluadas también desde el punto de vista nutricional y de salud, y se asigna un factor de certeza a cada una según su conveniencia saludable.

El sistema recomienda la receta más adecuada y permite al usuario decidir si desea más información sobre la receta. A continuación, el sistema pregunta al usuario si le gusta la receta. En caso de rechazo, se pregunta por el motivo (por ejemplo, duración excesiva), se ajustan los filtros en consecuencia, y se realiza una nueva búsqueda iterativa hasta encontrar una receta satisfactoria.

Además de mostrar ingredientes y propiedades especiales, el sistema justifica cada recomendación basándose en las preferencias indicadas por el usuario y en las características de la receta propuesta, fomentando así la trazabilidad y la transparencia de la recomendación final.

2. Procedimiento seguido

2.1. Base de conocimiento

La base de conocimiento se ha ido incrementando poco a poco desde la entrega 1. Es una base de conocimiento bien estructurada, fácil de entender y muy fácilmente modificable ya que en cualquier momento se puede aumentar o disminuir la información y el sistema seguiría funcionando correctamente.

Por una parte tenemos el fichero *recetas.txt* generado por todos los alumnos y yo misma añadiendo dos recetas. Lo he filtrado un poco ya que había algunos errores sintácticos en algunas recetas y tenía que seguir la plantilla que tenemos declarada en clips. Por otra parte, tenemos información sobre alimentos que se han utilizado en las recetas. Esta base se realizó en varias fases, siguiendo una estructura declarativa clara y modular en CLIPS. Está compuesta por plantillas de hechos (deftemplate) y hechos iniciales (deffacts) que modelan tanto la estructura de las recetas como el conocimiento nutricional y alimentario relevante. Se encuentran en el módulo main.

En primer lugar, se definieron las plantillas necesarias para representar los distintos elementos del dominio. Entre ellas destacan:

- La plantilla receta, que incluye atributos como ingredientes, dificultad, duración, coste, tipo de cocción, valores nutricionales, entre otros.
- Plantillas auxiliares para la gestión del flujo del sistema, como modo, preferencia-plato, preferencia-propiedad, receta-candidata, receta-seleccionada o confirmacion_receta.
- Plantillas específicas para representar propiedades alimentarias (propiedad_receta), factores de certeza (factorcerteza) o justificaciones (justificacion).
- Una vez establecida la estructura, se procedió a definir el conocimiento explícito del sistema mediante deffacts. Este conocimiento se dividió en bloques temáticos:
- Piramide alimentaria: se asignó a cada grupo de alimentos un nivel dentro de la pirámide nutricional (por ejemplo, frutas en nivel 2, dulces en el nivel 9). Esta información es útil para determinar si una receta puede considerarse saludable.
- Jerarquías y clasificaciones: se usaron hechos del tipo (es_un_tipo_de ...) para establecer relaciones jerárquicas entre alimentos. Por ejemplo, (es_un_tipo_de pollo carne_blanca) y (es_un_tipo_de carne_blanca carne)

permiten razonar de forma transitiva sobre ingredientes. Esto facilita la deducción de propiedades como *es vegetariana* o *contiene gluten*.

- Condimentos y bebidas: se catalogaron ingredientes que no son alimentos principales pero que pueden aparecer en recetas, como el vino o las especias, para no considerarlos erróneamente como fuentes de proteínas o grasas.
- Conocimiento ampliado: se incluyó una extensa lista de ingredientes comunes, organizados por grupos (carnes, pescados, frutas, verduras, cereales, etc.), y etiquetados según su rol alimentario. Esto amplía el alcance del sistema y mejora su capacidad para razonar sobre cualquier receta del dominio.

Todo este conocimiento se diseñó con el objetivo de ser reutilizable, ampliable y orientado a la inferencia. Gracias a esta base estructurada, el sistema puede analizar los ingredientes de una receta, deducir propiedades como si es vegana o sin gluten, y evaluar su adecuación según las preferencias del usuario.

2.2. Validación y verificación del sistema

Para garantizar la calidad y la fiabilidad del sistema experto, se ha llevado a cabo un procedimiento exhaustivo de verificación y validación durante su desarrollo. Este proceso ha tenido como objetivos principales asegurar el correcto funcionamiento técnico del sistema, comprobar la lógica de inferencia, y evaluar si la solución propuesta satisface de forma adecuada las necesidades del usuario. Para este procedimiento nos hemos pautado por la actividad desarollada en teoría de diseñar el proceso de validación para el SBC de la unidad de prácticas 6. En resumen, queremos garantizar que el sistema basado en el conocimiento desarrollado en CLIPS cumple:

- Correctamente su funcionalidad, es decir, proponer recetas adecuadas.
- Los requisitos de calidad del software, como puede ser veracidad, completitud, consistencia, explicabilidad...
- Responde con fiabilidad ante casos con incertidumbre.

2.2.1. Verificación del sistema

La verificación se centró en asegurar que el sistema estaba construido correctamente desde el punto de vista técnico. Se siguieron los siguientes pasos:

1. Verificación de sintaxis y carga:
    - Se comprobó que todos los archivos .clp del sistema (main, pedir, deducir, compatibles, proponer) se cargaban sin errores.
    - Se utilizó watch rules y watch facts para observar la ejecución de reglas y hechos en tiempo real, verificando que el sistema podía resetearse y ejecutarse correctamente.
2. Verificación de consistencia estructural y lógica:
    - Se revisaron manualmente las reglas del sistema para detectar:
        - Reglas no disparables o inalcanzables.
        - Posibles ciclos entre reglas o redundancias lógicas.
        - Contradicciones entre reglas (por ejemplo, inferir una propiedad y su contraria).
    - Se activaron módulos individualmente para comprobar la alcanzabilidad de todas las reglas.
3. Verificación de completitud:
    - Se realizaron pruebas con distintos tipos de entrada:
        - Casos sin restricciones.
        - Casos con múltiples restricciones (ej. vegana, rápida, baja en calorías).
    - Se comprobó que el sistema siempre proponía una receta adecuada o notificaba correctamente si no había ninguna compatible.
4. Pruebas de módulos por separado:
    - Se probaron de forma individual todos los módulos, activándolos mediante hechos de control del tipo (modulo <nombre>). De hecho, esto se realizó en la entrega 1 ya que fuimos creando módulo a módulo y viendo que funcionaban correctamente todos antes de acoplarlos.
    - Se verificó el correcto paso de control entre módulos y su activación secuencial adecuada.

2.2.2. Validación del sistema

La validación se centró en evaluar el comportamiento del sistema en condiciones reales de uso, teniendo en cuenta la experiencia del usuario y la calidad de las recomendaciones. Se aplicaron las siguientes metodologías:

1. Validación funcional con casos de prueba:
    - Se diseñaron perfiles de usuario con distintas combinaciones de preferencias y restricciones.
    - Para cada caso, se ejecutó el sistema y se registraron la receta recomendada y su justificación.
    - Se validó que:
        - La receta cumplía las preferencias.
        - La justificación era coherente.
        - El sistema solo preguntaba lo necesario.
2. Validación del razonamiento con incertidumbre:
    - Se incorporaron factores de certeza en algunas reglas para manejar información ambigua o incompleta.
    - Se verificó que el sistema ofrecía recomendaciones razonables incluso en ausencia de información explícita.
3. Validación interpretativa:
    - Se analizó la explicación generada por el sistema al recomendar una receta. Son explicaciones bastante claras y contundentes para que el usuario pueda entender por qué le ha recomendado esa receta.
    - Se comprobó que la justificación era comprensible y relevante.
4. Análisis de sensibilidad: se modificaron ligeramente algunas restricciones o preferencias para comprobar que el sistema no cambiaba radicalmente de recomendación sin motivo justificado, asegurando así la estabilidad ante pequeñas variaciones.

2.2.3. Casos de prueba

Vamos a comprobar los casos de prueba que especificamos en la actividad para ver si se cumplen.

Entrada del usuario	**Preferencias / Restricciones**	**Resultado esperado**	**Resultado real**
Usuario sin restricciones	Ninguna	Cualquier receta adecuada	Elige una receta aleatoriamente entre todas
Usuario vegano	Preferencia vegana	Receta con ingredientes compatibles	Receta vegana
Usuario rechaza la primera receta	Interacción con rechazo	Nueva receta propuesta	Nueva receta propuesta
Usuario pide postre fácil	tipo_plato = postre, dificultad = baja	Receta que cumpla ambas condiciones	Postre fácil
Usuario pide explicación	Solicita justificación	Sistema explica por qué cumple requisitos	Siempre se le explica el por qué

Cuadro 1: Casos de prueba utilizados en la validación del sistema

3. Descripción del sistema

En esta sección vamos a profundizar más en nuestro sistema y en su explicación. Como ya sabemos, ha sido desarrollado en CLIPS con una arquitectura modular y extensible, orientada a la recomendación de recetas en función de las preferencias del usuario. Esta organización favorece tanto la comprensión del sistema como su mantenimiento y ampliación por parte de futuros desarrolladores.

3.1. Variables de entrada

Las variables de entrada del sistema representan la información que el usuario proporciona explícitamente durante la interacción, así como las posibles restricciones o preferencias que condicionan la recomendación de recetas. Estas variables se modelan como hechos en CLIPS, y se utilizan para activar reglas y filtrar el conocimiento relevante. A continuación se describen las principales variables de entrada:

- **Tipo de plato:** Representa la categoría de receta que desea el usuario (por ejemplo, postre, plato principal, desayuno/merienda...).

- Representación: `(preferencia-plato (tipo postre))`
- Valores posibles: `entrante`, `primer_plato`, `plato_principal`, `postre`, `desayuno_merienda`, `acompanamiento`

- **Propiedades especiales:** Son restricciones opcionales relacionadas con el estilo de vida o necesidades alimentarias del usuario, como si busca una receta vegana, sin gluten, de dieta...
  - Representación: `(preferencia-propiedad (tipo es_vegana))`
  - Valores posibles: `es_vegana`, `es_vegetariana`, `es_sin_gluten`, `es_picante`, `es_sin_lactosa`, `es_de_dieta`

- **Motivo de rechazo:** Si el usuario rechaza la receta propuesta, el sistema solicita que indique el aspecto que no le ha gustado, para refinar la búsqueda. Las posibles causas de rechazo son:
  - **Ingredientes:** Si el usuario no quiere que se repitan ciertos ingredientes en la siguiente propuesta.
    - Representación: `(detalle-rechazo (tipo ingrediente) (valor cebolla))`
    - Efecto: Se filtran todas las recetas que contengan ese ingrediente.
  - **Duración:** Si la receta anterior tarda demasiado.
    - Representación: `(detalle-rechazo (tipo duracion) (valor 20))`
    - Efecto: Solo se considerarán recetas con una duración igual o menor al valor especificado (en minutos).
  - **Dificultad:** Si la receta era demasiado difícil.
    - Representación: `(detalle-rechazo (tipo dificultad) (valor media))`
    - Efecto: Se excluyen las recetas con nivel de dificultad superior al indicado.

  Esta información se recoge primero mediante un hecho del tipo:

  ```
  (motivo-rechazo (tipo duracion))
  ```

  y posteriormente mediante un hecho más específico que incluye el valor concreto:

  ```
  (detalle-rechazo (tipo duracion) (valor 20))
  ```

- **Deseo de información adicional:** El sistema permite al usuario solicitar más detalles sobre una receta propuesta.
  - Representación: `(desea-info (valor si))`
- **Confirmación de aceptación de la receta:** El usuario puede confirmar si le gusta o no la receta recomendada.
  - Representación: `(confirmacion_receta (valor si))` o `(valor no)`

3.2. Variables de salida

Las variables de salida del sistema representan las conclusiones o resultados generados durante la ejecución, en función de las entradas del usuario y del conocimiento almacenado. Estas salidas incluyen tanto la receta recomendada como sus propiedades, la explicación generada y el razonamiento con incertidumbre asociado.

Las variables de salida se expresan mediante hechos CLIPS creados por reglas de deducción o selección. A continuación se detallan las principales:

- **Receta recomendada:** Es la salida principal del sistema, que contiene el nombre de la receta seleccionada.
  - Representación: `(recomendacion (receta arroz_a_la_cubana))`
- **Recetas candidatas:** Conjunto de recetas compatibles con las restricciones del usuario, entre las que se selecciona la final.
  - Representación: `(receta-candidata (nombre jugo_de_pinia_y_papaya))`
- **Receta seleccionada:** Hecho auxiliar para mostrar los detalles completos de la receta elegida.
  - Representación: `(receta-seleccionada (nombre flan_de_dulce_de_leche))`
- **Propiedades deducidas de la receta:** Indica si la receta propuesta cumple ciertas propiedades especiales (vegana, sin gluten, etc.), ya sea de forma directa o inferida.
  - Representación: `(propiedad_receta (tipo es_sin_gluten) (receta flan_de_dulce_de_leche))`

- **Justificación de la recomendación:** Texto explicativo generado por el sistema para argumentar por qué se considera saludable o adecuada la receta.
    - Representación: `(justificacion (propiedad es_de_dieta) (receta jugo_de_pinia_y_papaya) (texto ‘‘Tiene mucha fibra, eso es saludable.’’))`
- **Razonamiento con incertidumbre:** Factor de certeza asociado a la valoración saludable de una receta, en un rango entre -1.0 (nada saludable) y 1.0 (muy saludable).
    - Representación: `(factor-certeza (propiedad es_saludable) (receta flan_de_dulce_de_leche) (valor -0.706))`

3.3. Conocimiento global del sistema

El conocimiento global del sistema representa la base común y compartida que está disponible desde el inicio de la ejecución. Se carga mediante `deffacts` en el módulo `MAIN` y proporciona al sistema los datos necesarios para razonar sobre ingredientes, clasificaciones, salud nutricional y jerarquías alimentarias. Este conocimiento no depende de la interacción con el usuario. Se ha ido introduciendo desde la prática 2 y ha sido desarrollado también en pate por el profesor y aumentado por mí parte. El conocimiento global incluye:

- **Niveles de la pirámide alimentaria:** Clasificación de grupos de alimentos según su frecuencia recomendada de consumo. Cuanto más bajo es el nivel, más saludable se considera el grupo.
    - Ejemplo: `(nivel_piramide_alimentaria verdura 1)`
- **Relaciones jerárquicas entre alimentos:** Conocimiento de tipo `es_un_tipo_de` que permite agrupar ingredientes bajo categorías nutricionales o culinarias. Estas relaciones permiten la inferencia transitiva sobre propiedades.
    - Ejemplo: `(es_un_tipo_de pollo carne_blanca)`, `(es_un_tipo_de carne_blanca carne)`
- **Clasificación de ingredientes especiales:** Grupos como *condimentos*, *especias*, *bebidas*, *gluten*, etc., que permiten tratar adecuadamente ingredientes que no aportan nutrientes clave pero afectan a propiedades de la receta (por ejemplo, si es sin gluten).

  - Ejemplo: `(es_un_tipo_de harina gluten)`, `(es_un_tipo_de vino bebida)`

- **Conocimiento ampliado de ingredientes:** Amplia base de hechos que agrupan ingredientes habituales (carnes, pescados, frutas, verduras, legumbres, cereales, frutos secos, dulces, etc.) para aumentar la cobertura del sistema frente a nuevas recetas.

  - Ejemplo: `(es_un_tipo_de calabacin verdura)`, `(es_un_tipo_de quinoa cereales)`, `(es_un_tipo_de miel dulces)`

Este conocimiento inicial es esencial para el razonamiento automático del sistema. Gracias a él, las reglas pueden deducir propiedades de las recetas, identificar ingredientes problemáticos, o estimar el nivel de salud de un plato. Además, al estar expresado en forma de hechos independientes, es fácilmente ampliable por cualquier persona sin necesidad de modificar las reglas del sistema.

3.4. Especificación de los módulos

El sistema se ha diseñado siguiendo una estructura modular clara y jerárquica, lo que permite una separación funcional de responsabilidades, facilita la comprensión del sistema y mejora su mantenibilidad. Cada módulo corresponde a una fase del proceso de recomendación y se activa mediante un hecho de control del tipo `(modulo <nombre>)`. A continuación se describe la estructura y el contenido de cada módulo.

3.4.1. Estructura en módulos

El sistema consta de los siguientes módulos:

- **main.clp** – Módulo principal: define las plantillas, los hechos iniciales del conocimiento global y controla el flujo general del sistema.
- **pedir.clp** – Módulo de interacción inicial: solicita al usuario sus preferencias (tipo de plato y propiedades especiales).
- **deducir.clp** – Módulo de inferencia: deduce propiedades de las recetas en base a sus ingredientes, utilizando reglas de conocimiento.
- **compatibles.clp** – Módulo de filtrado: identifica las recetas que son compatibles con las preferencias del usuario.

- **proponer.clp** – Módulo de recomendación: selecciona una receta candidata, presenta sus detalles, justifica su elección y maneja la interacción con el usuario (rechazo, petición de información, etc.).

3.4.2. Descripción de cada módulo

1. `main.clp` – Módulo principal

- Objetivo: Define la base de conocimiento y plantillas necesarias para todo el sistema. Carga los hechos iniciales y establece la estructura modular.
- Conocimiento que utiliza: Relaciones jerárquicas de ingredientes, pirámide alimentaria, clasificaciones alimentarias, conocimiento ampliado.
- Conocimiento que deduce: No deduce hechos directamente, pero provee la infraestructura sobre la que trabajan el resto de módulos.

2. `pedir.clp` – Módulo de interacción con el usuario

- Objetivo: Recoger las preferencias del usuario de forma conversacional, preguntando por tipo de plato y propiedades especiales deseadas.
- Hechos que utiliza: `(modulo pedir-informacion)`
- Hechos que deduce:
  - `(preferencia-plato (tipo ...))`
  - `(preferencia-propiedad (tipo ...))`
- Reglas clave: Pregunta al usuario sólo por la información necesaria, evitando repeticiones. Si el usuario introduce una opción no válida, se asume que no hay restricción para esa variable.

3. `deducir.clp` – Módulo de deducción de propiedades

- Objetivo: Analizar cada receta e inferir sus propiedades especiales (como vegana, sin gluten, sin lactosa, etc.), a partir de sus ingredientes.
- Hechos que utiliza:
  - `(receta ...)`
  - `(es_un_tipo_de ...)`

- Hechos que deduce:
  - `(propiedad_receta (tipo ...) (receta ...) (ingrediente ...))`
- Reglas clave: Aplicación de inferencia transitiva para determinar si una receta contiene ingredientes de origen animal, con gluten, etc. Utiliza listas de ingredientes para comprobar si se viola alguna propiedad especial.

4. `compatibles.clp` – Módulo de filtrado de recetas

- Objetivo: Determinar qué recetas de la base de datos son compatibles con las preferencias y restricciones del usuario.
- Hechos que utiliza:
  - `(receta ...)`
  - `(preferencia-plato ...)`
  - `(preferencia-propiedad ...)`
  - `(propiedad_receta ...)`
- Hechos que deduce:
  - `(receta-candidata (nombre ...))`
- Reglas clave: Una receta es candidata si cumple el tipo de plato requerido y no incumple ninguna de las restricciones activas.

5. `proponer.clp` – Módulo de recomendación y explicación

- Objetivo: Seleccionar una receta candidata y presentarla al usuario. Gestionar interacciones posteriores (rechazo, petición de más información) y justificar la elección.
- Hechos que utiliza:
  - `(receta-candidata ...)`
  - `(preferencia-plato ...)`
  - `(preferencia-propiedad ...)`
  - `(factor-certeza ...)`
- Hechos que deduce:

- `(recomendacion ...)`
- `(receta-seleccionada ...)`
- `(justificacion ...)`

- Reglas clave: Justifica por qué una receta ha sido elegida, ofrece explicación nutricional basada en la pirámide alimentaria, y permite ajustes tras rechazo (por ejemplo, filtrar por duración máxima).

Gracias a esta estructura modular, el sistema es fácilmente ampliable: se pueden añadir nuevas propiedades, reglas o tipos de receta sin necesidad de alterar el flujo general. Además, el uso explícito de hechos y plantillas favorece la comprensión por parte de otros desarrolladores o usuarios.

4. Razonamiento con incertidumbre

Para esta práctica hemos añadido dos tipos de razonamiento con incertidumbre. El añadirlo lo hace un sistema más real ya que en entornos reales, los usuarios no siempre proporcionan toda la información necesaria, y las decisiones pueden depender de factores subjetivos o incompletos. Por ello, un sistema experto práctico debe ser capaz de manejar la incertidumbre, evaluar alternativas parcialmente adecuadas y justificar sus decisiones con grados de confianza. En este sistema, se han introducido mecanismos de razonamiento con incertidumbre para abordar estas situaciones de forma flexible y realista. Hemos implementado dos razonamientos con incertidumbre:

- Factores de certeza: permiten asignar un valor numérico al nivel de confianza de ciertas inferencias, en nuestro caso sobre la salud de una receta.
- Razonamiento por defecto: permite al sistema continuar la ejecución y aceptar valores por defecto en caso de que el usuario no proporcione información explícita.

4.1. Factores de certeza

Los factores de certeza permiten representar conocimiento incierto de forma cuantitativa. En nuestro caso, se utilizan para evaluar si una receta puede considerarse saludable, incluso aunque contenga algunos elementos desfavorables. Cada razonamiento aporta un valor en el rango `[-1.0, 1.0]`, donde:

- Valores positivos indican si es saludable,
- Valores negativos indican riesgo o inconvenientes,
- Valores cercanos a cero reflejan duda o neutralidad. Es con lo que se inicializa cada receta.

El sistema utiliza un conjunto de reglas que asignan valores positivos o negativos al factor de certeza de una receta en función de sus ingredientes. Estas reglas están basadas en criterios nutricionales comunes:

- Aumentan el factor de certeza (receta más saludable):
  - Verduras saludables (ej. lechuga, tomate, calabacín, brócoli...): `+0.5`

- Tipo de cocción al horno: `+0.4`
- Tipo de cocción al vapor: `+0.5`
- Alta en fibra (más de 5g): `+0.3`
- Baja en calorías (menos de 200): `+0.4`
- Proteínas moderadas (entre 10 y 30g): `+0.2`
- Presencia de integrales (pan integral, arroz integral, avena): `+0.3`

- Reducen el factor de certeza (receta menos saludable):
  - Tipo de cocción frita: `-0.6`
  - Grasa alta (más de 50g): `-0.4`
  - Colesterol alto (más de 200mg): `-0.3`
  - Productos procesados (salchichas, mayonesa): `-0.4`
  - Contiene azúcar: `-0.3`

Cada vez que se activa una regla de evaluación saludable, el sistema genera no solo un `factor-certeza`, sino también un hecho explicativo en lenguaje natural. Este hecho sigue el formato:

```
(justificacion (propiedad saludable) (receta flan_de_dulce_de_leche)
(texto "Lleva azúcar, no deberías de tomar."))
```

Estos textos sirven para justificar ante el usuario por qué una receta ha sido valorada como saludable o no. En tiempo de ejecución, se recopilan todos los textos generados para cada receta y se muestran al usuario como explicación:

```
>>> Justificación de salud para la receta
Flan de dulce de leche:
- Tiene mucha grasa, no se debería de comer mucha.
- Lleva azúcar, no deberías de tomar.
```

4.2. Razonamiento por defecto

El razonamiento por defecto permite continuar la inferencia cuando no se ha recibido una respuesta clara del usuario. En sistemas interactivos, es común que el usuario:

- Omita información.

- Introduzca valores no válidos o ambiguos.
- No tenga preferencias claras sobre ciertos criterios.

Sin esta capacidad, el sistema quedaría bloqueado o requeriría preguntar de nuevo constantemente, lo que afectaría negativamente a la experiencia de uso.

Cuando el usuario no indica una propiedad especial válida, el sistema infiere por defecto que no hay restricciones, mediante un mensaje como:

```
No se especificó ninguna propiedad especial. Se aceptan
todas las recetas del tipo indicado.
```

Esto se gestiona mediante reglas que detectan errores de entrada o valores vacíos y generan un hecho neutral:

```
(preferencia-propiedad (tipo sin_restriccion))
```

Gracias a este mecanismo, el sistema no necesita forzar al usuario a responder correctamente y puede continuar razonando de forma coherente, aunque con menor personalización.

Además, antes de empezar a preguntar propiedades, el sistema recomienda una receta por defecto. Esta receta es neutra y que suele gustar a todo el mundo.

En conjunto, ambos mecanismos permiten que el sistema mantenga un comportamiento flexible, tolerante a errores y capaz de explicar sus decisiones en situaciones de información incompleta o ambigua.

5. Manual del sistema

Ejecución SBC

```
(clear) ;por si se han realizado ejecuciones anteriores
(load main.clp) ; aqui se cargan los otros modulos
(reset)
(run)
```

Este manual describe el funcionamiento básico del sistema experto desde el punto de vista del usuario. El sistema está diseñado para ejecutarse en el entorno CLIPS y proporciona recomendaciones de recetas personalizadas a partir de las preferencias alimentarias del usuario.

Inicio del sistema

Al ejecutar el sistema, se carga automáticamente una receta por defecto que puede servir como sugerencia inicial. El usuario puede aceptar esta receta o rechazarla para iniciar una recomendación personalizada.

```
¿Te gusta esta receta? (si / no): no
```

Si respondiese que sí, el sistema se despide y termina la ejecución.

Paso 1: Introducción de preferencias

Una vez rechazada la receta por defecto, el sistema solicita al usuario que introduzca sus preferencias sobre el tipo de plato y posibles restricciones alimentarias (propiedades especiales). Si se escribe algo que no sea una opción, en el tipo de plato se cogen todos los platos y en la propiedad especial se asumirá que no hay ninguna restricción.

- **Tipo de plato:** puede ser `entrante`, `primer_plato`, `plato_principal`, `postre`, `desayuno_merienda`, o `acompanamiento`.
- **Propiedades especiales:** el usuario puede indicar si desea una receta `es_vegana`, `es_vegetariana`, `es_sin_gluten`, `es_picante`, `es_sin_lactosa` o `es_de_dieta`. También puede escribir `ninguna` si no tiene restricciones.

```
¿Qué tipo de plato buscas?: postre
¿Buscas alguna propiedad especial?: ninguna
```

Paso 2: Recomendación de una receta

El sistema busca entre las recetas disponibles aquellas que se ajusten a las preferencias del usuario y selecciona una de ellas como propuesta. Muestra un resumen con:

- Tipo de plato e ingredientes
- Propiedades deducidas (por ejemplo, si es sin gluten)
- Factor de certeza sobre si se considera saludable
- Explicación de por qué se cree que es saludable o no (alimentos o cocción que le ha hecho deducirlo)

```
Te recomendamos la receta: Flan de dulce de leche
```

Paso 3: Solicitud de información adicional

El usuario puede solicitar más información sobre la receta recomendada:

```
¿Quieres más información sobre esta receta? (si / no):
si
```

El sistema mostrará entonces todos los detalles: autor, dificultad, duración, ingredientes completos, tipo de cocción, valores nutricionales, y el enlace original.

Paso 4: Rechazo y refinamiento de búsqueda

Si la receta propuesta no es satisfactoria, el sistema permite rechazarla e indicar el motivo:

- **Ingredientes**: el usuario no desea que se incluyan ciertos ingredientes.
- **Dificultad**: el usuario desea una receta más sencilla.
- **Duración**: el usuario desea una receta más rápida.

Ejemplo:

```
¿Qué aspecto no te convence de la receta?: duracion
¿Cuál es el tiempo máximo de preparación en minutos?: 20
```

Para la dificultada y duración, como es una lista cerrada de opciones, si no se introduce una que está en la lista se vuelve a lanzar la pregunta hasta que se introduzca una opción correcta. El sistema filtra las recetas y realiza una nueva recomendación que se ajusta al límite introducido.

Paso 5: Aceptación de la receta

Una vez el usuario está satisfecho con la receta propuesta, puede confirmarlo:

```
¿Te gusta esta receta? (si / no): si
```

El sistema justifica por qué la receta cumple con las preferencias expresadas (tipo de plato, ingredientes y propiedades) y finaliza la interacción.

Cierre

El sistema termina la ejecución con un mensaje de despedida:

```
Gracias por usar el recomendador de recetas. ¡Buen provecho!
```

Resumen del flujo de uso

1. Se muestra una receta por defecto.
2. El usuario puede rechazarla y expresar sus preferencias. Si la acepta, termina el programa.
3. Se genera una receta personalizada con justificación.
4. El usuario puede pedir más información o rechazarla indicando por qué.
5. Se realiza una nueva búsqueda refinada si es necesario.
6. Una vez aceptada, el sistema explica la elección y finaliza.

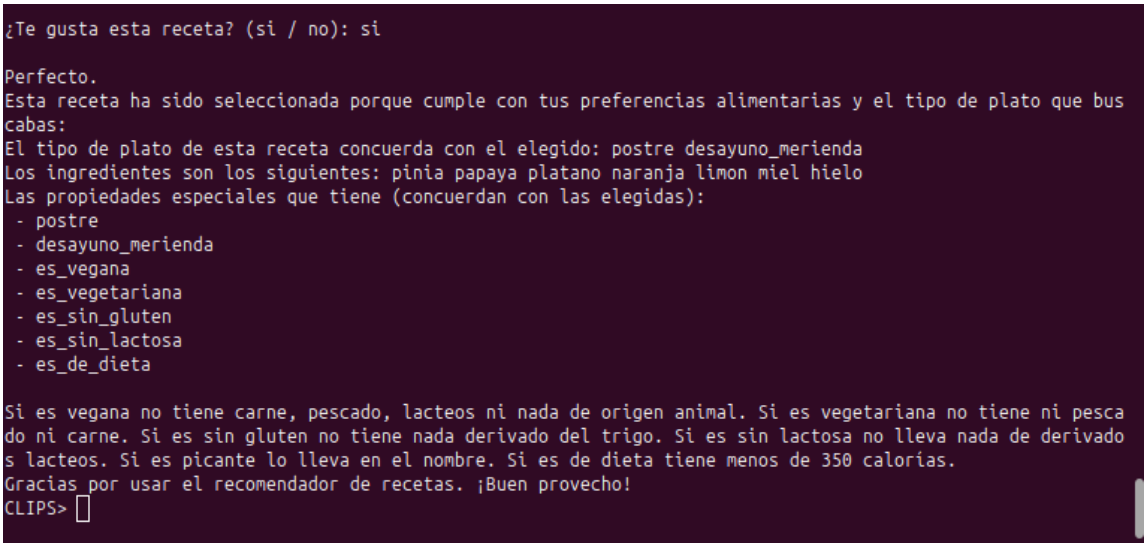
6. Ejemplo de funcionamiento

Este es un ejemplo de una ejecución de nuestro sistema:

```
Te recomendamos la receta por defecto:
- Tipo de plato: plato_principal
- Ingredientes: arroz tomate cebolla aceite_de_oliva huevo
- Personas: 2
- Dificultad: baja
- Duración: 30m
- Coste: bajo
- Tipo de cocción: cocido
- Cocina: mediterranea
- Temporada: todo_el_anio
- Calorías: 53
- Proteínas: 13
- Grasas: 20
- Carbohidratos: 66
- Fibra: 2
- Colesterol: 10
- Enlace: https://www.recetasgratis.net/receta-de-arroz-a-la-cubana-facil-y-rapido-72089.html

¿Te gusta esta receta? (si / no): no

Hola. Soy un recomendador de platos según tus necesidades.
¿Qué tipo de plato buscas? (entrante, primer_plato, plato_principal, postre, desayuno_merienda, acompanamie
nto): postre
¿Buscas alguna propiedad especial? (es_vegana, es_vegetariana, es_sin_gluten, es_picante, es_sin_lactosa, e
s_de_dieta) o escribe 'ninguna': ninguna
No se aplicará ninguna restricción por propiedad especial.

Resumen de tus preferencias:
  Tipo(s) de plato: postre
  Propiedad especial: ninguna

No se especificó ninguna propiedad especial. Se aceptan todas las recetas del tipo indicado.
Esta receta se considera saludable con certeza -0.706

Te recomendamos la receta: *** Flan de dulce de leche ***
Tipo de plato: postre
Ingredientes: leche crema_de_leche maicena huevo yema azucar dulce_de_leche
Propiedades especiales:
 - no_es_vegana
 - es_vegetariana
 - es_sin_gluten
 - es_con_lactosa


>>> Justificación de salud para la receta *** Flan de dulce de leche ***
 - Tiene mucha grasa, no se deberia de comer mucha.
 - Lleva azucar, no deberias de tomar.
→ Esta receta se considera saludable con un factor de certeza de -0.706

¿Quieres más información sobre esta receta? (si / no): si
```

Figura 1: Ejemplo

```
¿Quieres más información sobre esta receta? (si / no): si

*** Información detallada de la receta: ***
Autor: Pedro Luis Abril Ortega
Para 10 personas
Dificultad: baja, Duración: 90m
Tipo(s) de plato: postre
Ingredientes: leche crema_de_leche maicena huevo yema azucar dulce_de_leche
Tipo de cocción: crudo, Tipo cocina:
Temporada: nil, Coste: nil
Valores nutricionales - Cal: 2967, Prot: 83, Grasa: 171, Carb: 276, Fibra: 0, Colesterol: 2371
Enlace: https://www.recetasgratis.net/receta-de-flan-de-dulce-de-leche-76973.html

¿Te gusta esta receta? (si / no): no
¿Qué aspecto no te convence de la receta? (ingredientes, dificultad, duracion): duracion
¿Cuál es el tiempo máximo de preparación en minutos (solo número, ejemplo: 45)?: 20
Filtrado por duración completado. Máximo permitido: 20 minutos.
Esta receta se considera saludable con certeza 0.3

Te recomendamos la receta: *** Jugo de pinia y papaya ***
Tipo de plato: postre desayuno_merienda
Ingredientes: pinia papaya platano naranja limon miel hielo
Propiedades especiales:
 - postre
 - desayuno_merienda
 - es_vegana
 - es_vegetariana
 - es_sin_gluten
 - es_sin_lactosa
 - es_de_dieta


>>> Justificación de salud para la receta *** Jugo de pinia y papaya ***
 - Tiene mucha fibra, eso es saludable.
→ Esta receta se considera saludable con un factor de certeza de 0.3

¿Quieres más información sobre esta receta? (si / no): si

*** Información detallada de la receta: ***
Autor: Juan Manuel Garzon Ferrer
Para 1 personas
Dificultad: muy_baja, Duración: 10m
Tipo(s) de plato: postre desayuno_merienda
Ingredientes: pinia papaya platano naranja limon miel hielo
Tipo de cocción: crudo, Tipo cocina:
Temporada: nil, Coste: nil
Valores nutricionales - Cal: 283, Prot: 5, Grasa: 1, Carb: 64, Fibra: 8, Colesterol: 0
Enlace: https://www.recetasgratis.net/receta-de-jugo-de-pina-y-papaya-77229.html

¿Te gusta esta receta? (si / no): si
```


¿Te gusta esta receta? (si / no): si
Perfecto.
Esta receta ha sido seleccionada porque cumple con tus preferencias alimentarias y el tipo de plato que bus
cabas:
El tipo de plato de esta receta concuerda con el elegido: postre desayuno_merienda
Los ingredientes son los siguientes: pinia papaya platano naranja limon miel hielo
Las propiedades especiales que tiene (concuerdan con las elegidas):
- postre
- desayuno_merienda
- es_vegana
- es_vegetariana
- es_sin_gluten
- es_sin_lactosa
- es_de_dieta
Si es vegana no tiene carne, pescado, lacteos ni nada de origen animal. Si es vegetariana no tiene ni pesca
do ni carne. Si es sin gluten no tiene nada derivado del trigo. Si es sin lactosa no lleva nada de derivado
s lacteos. Si es picante lo lleva en el nombre. Si es de dieta tiene menos de 350 calorías.
Gracias por usar el recomendador de recetas. ¡Buen provecho!
CLIPS>