Ingeniería De Sistemas Basados En Conocimientos

Práctica 2. Uso de ontologías en un sistema CBR

Miguel Alexander Maldonado Lenis

Facultad de Informática

Índice

Estructura del proyecto…….*3*

Funcionamiento……8

Restricciones…….……..12

Función de similitud…….……..13

Función de sustitución…….……..14

Estructura del proyecto

En esta sección se describirá la estructura de paquetes del proyecto.

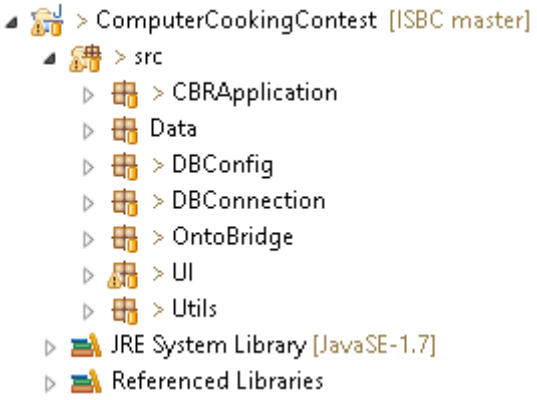

Ilustración 1

Paquete CBRApplication

El paquete ***CBRApplication*** contiene las clases ***SandwichDescription.java***, ***SandwichRecommender.java***, ***SandwichSolution.java*** y ***JointSandwichDescription.java***

Ilustración 2

SandwichDescription.java contiene la representación de los casos.

```
private String caseID;
private String ingredient1;
private String ingredient2;
private String ingredient3;
private String ingredient4;
private String ingredient5;
```

Ilustración 3

SandwichRecommender.java representa la clase principal de la aplicación y contiene los métodos que se encargar de realizar el ciclo CBR (Configure, PreCycle, Cycle y PostCycle) y el ciclo CBR para el uso de la ontología (`cycleWithOntology`).

SandwichSolution.java contiene la representación de la solución (casos recuperados).

```
private String caseID;
private String ingredient1;
private String ingredient2;
private String ingredient3;
private String ingredient4;
private String ingredient5;
```

Ilustración 4

***JointSandwichDescription*.java** contiene la representación de los casos para el modo de uso con la ontología.

```
private String caseID;
private ArrayList<String> ingredients;
```

Ilustración 5

Paquete Data

El paquete ***Data*** contiene un conjunto de clases que representan de manera lógica la siguiente jerarquía de alimentos:

Ilustración 6

Ilustración 7

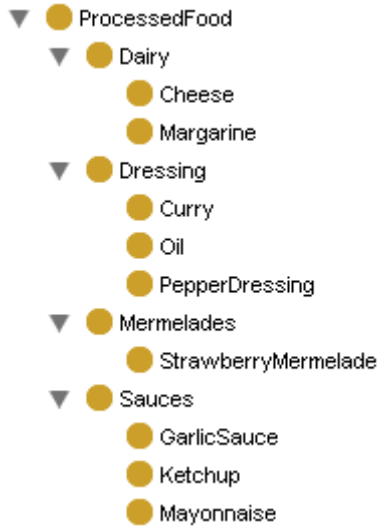

Ilustración 8

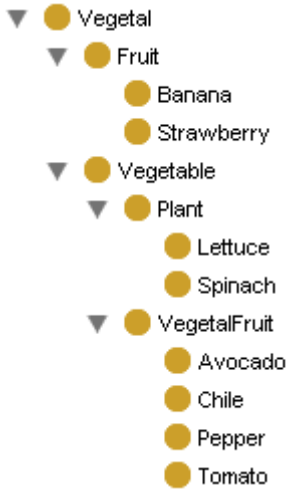

Ilustración 9

Cabe destacar el método ***getLastChildsInHierarchy,*** implementado por todas las clases, el cual permite acceder a las hojas de cualquier rama de la jerarquía pudiendo, de esta forma, acceder a los ingredientes "finales".

<u>Por ejemplo</u>:

Partiendo de la categoría **Vegetable**, el método devuelve las hojas: **Lettuce**, **Spinach**, **Avocado**, **Chile**, **Pepper** y **Tomato**.

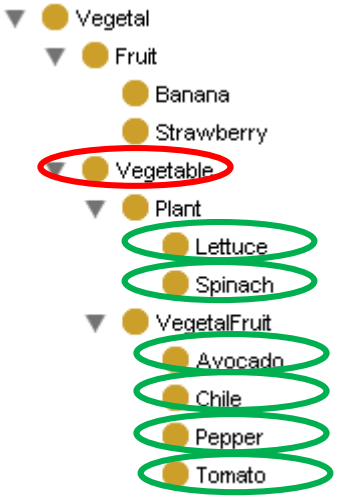

Ilustración 10

Paquete DBConfig

El paquete ***DBConfig*** contiene un conjunto de clases necesarias para la configuración de la BBDD, así como la configuración del mapeo entre los atributos de los casos y las columnas de la tabla en la BBDD, la configuración de ***Hibernate*** (librería encargada de leer y escribir de la BBDD)

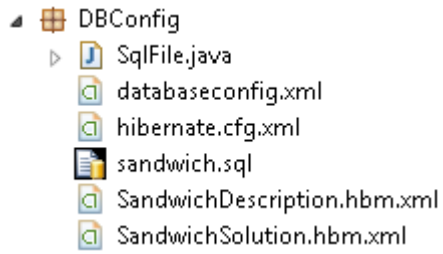

Ilustración 11

Paquete DBConnection

El paquete ***DBConnection*** contiene la clase ***HSQLDBServer.java***, la cual se encarga de crear la BBDD, configurar su conexión y lanzar el servidor.

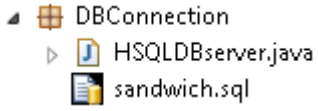

Ilustración 12

Paquete OntoBridge

El paquete ***OntoBridge*** contiene la implementación de la funcionalidad mediante el uso de la ontología así como el archivo .OWL que contiene la ontología.

En la clase ***OntologySimilarityFunction.java*** se encuentra definida la función de evaluación que se emplea para el filtrado de casos.

La clase ***OntologyUsefulFunctions.java*** contiene una serie de métodos que facilitan el recorrido del árbol de la ontología.

Se ha modificado la clase ***PnlConceptsTree.java*** para poder registrar los elementos que se seleccionan en el árbol.

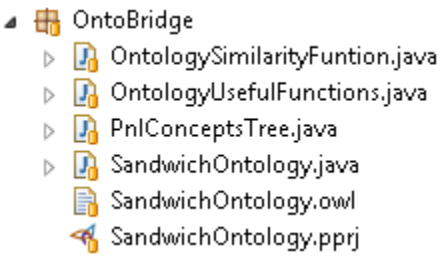

Ilustración 13

Por último, en la clase ***SandwichOntology.java*** se ha implementado la carga de la ontología así como un control de selección para los elementos del árbol y métodos que facilitan su visualización en la interfaz.

Paquete UI

El paquete ***UI*** contiene las clases es las que se define la interfaz de usuario.

Ilustración 14

Paquete Utils

El paquete ***Utils*** contiene la clase ***LevelChildHierarchy.java***, define una estructura que permite comprobar si cualquier ingrediente es un ingrediente “final” (si es una hoja en la jerarquía de alimentos). Y las clases ***ArrayUtils.java***, la cual contiene un método para la concatenación de ArrayList’s y ***StringEvaluartor.java*** contiene dos métodos para poder extraer de una cadena con un formato específico los elementos que se encuentran en esa cadena.

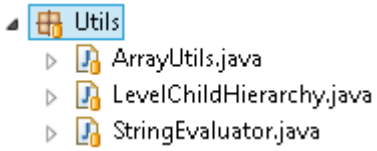

Ilustración 15

Funcionamiento

En esta sección se describirá la interfaz de usuario y el funcionamiento de la aplicación.

Interfaz

En la interfaz nos encontramos con dos modos de uso o selección. El primero, Listas, tiene la misma funcionalidad que en la práctica 1. El segundo, Árbol, permite la selección de elementos en el árbol de la ontología y habilita el uso de la ontología para el ciclo CBR.

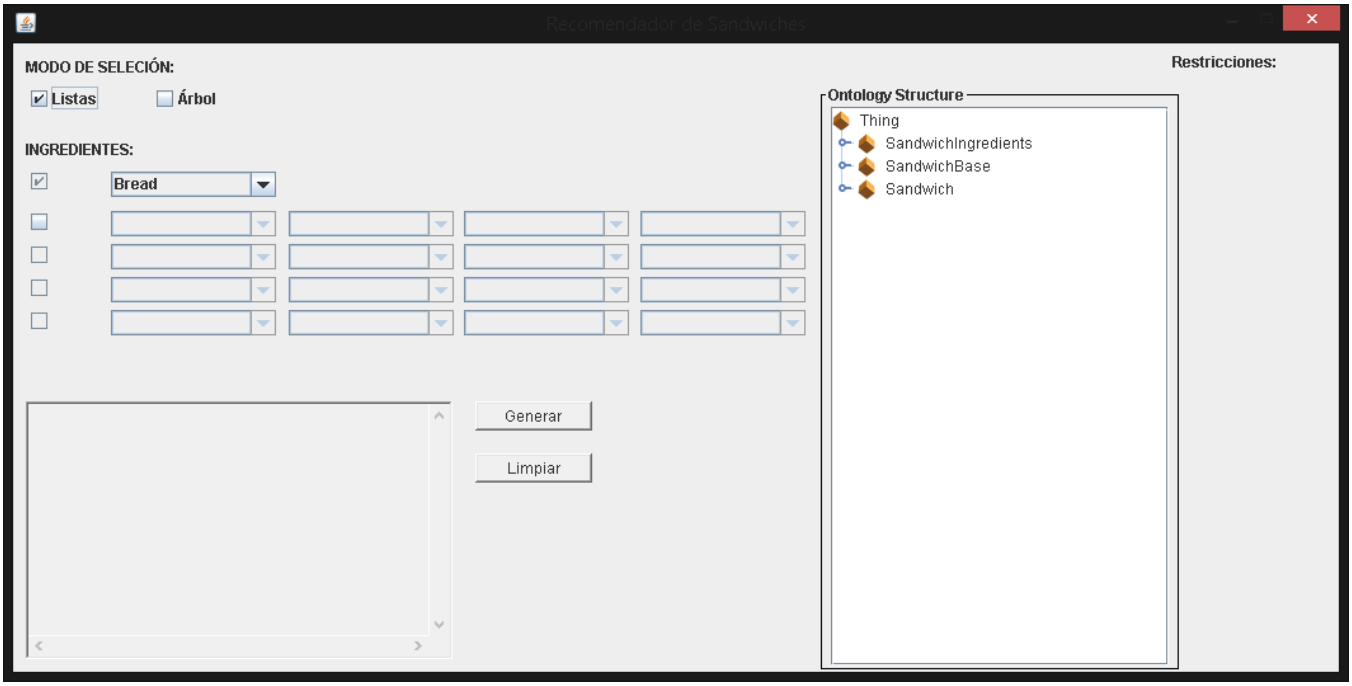

Ilustración 16

1. Selección de elementos (restricciones)

i) Click sobre el botón derecho del ratón en cualquier elemento

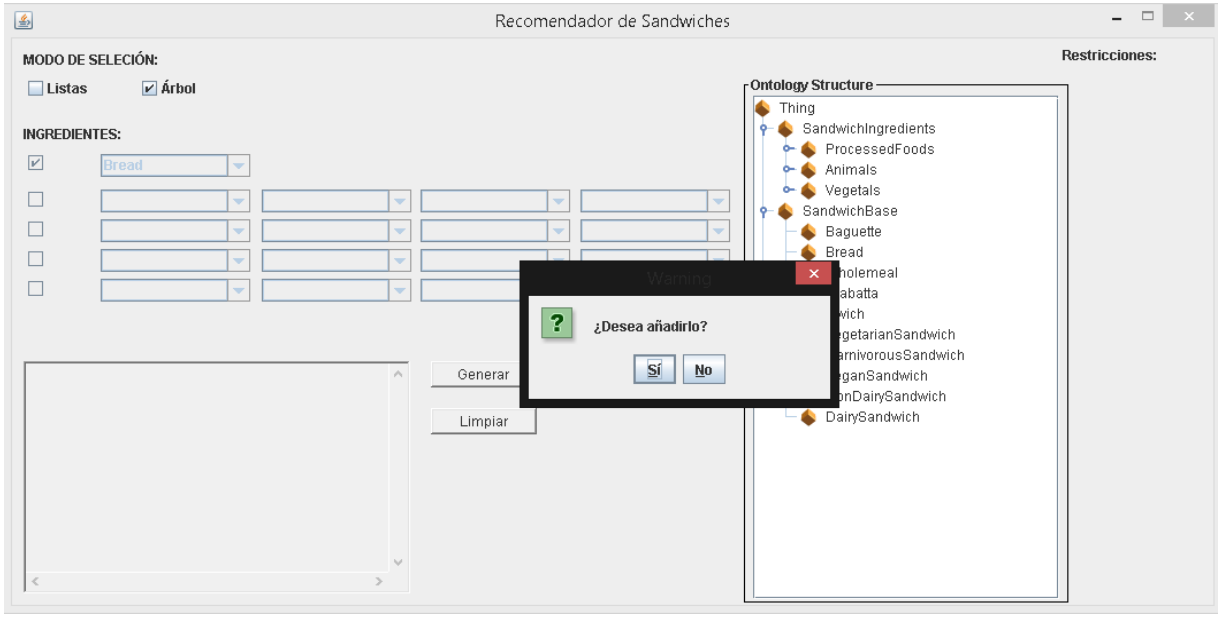

Ilustración 17

ii) "Sí"

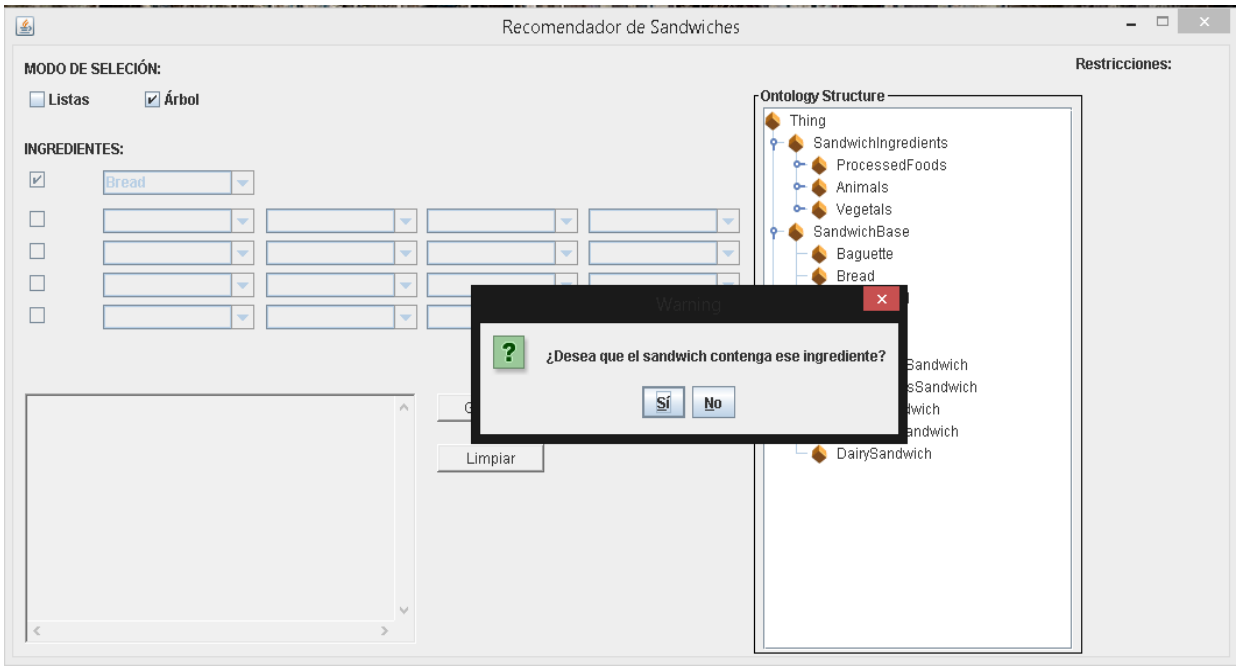

Ilustración 18

iii) “Sí”, se añade ese elemento como restricción “positiva” (Ingrediente que se quiere que contenga el sándwich)

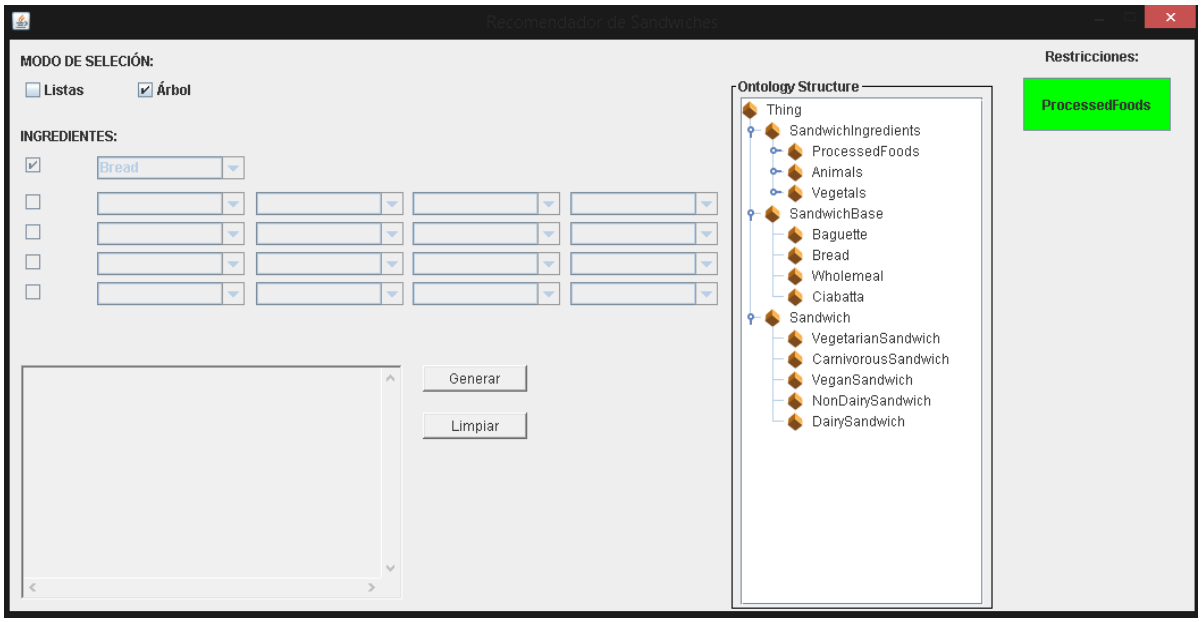

Ilustración 19

iv) “No”, se añade ese elemento como restricción “negativa” (Ingrediente que no se quiere que contenga el sándwich)

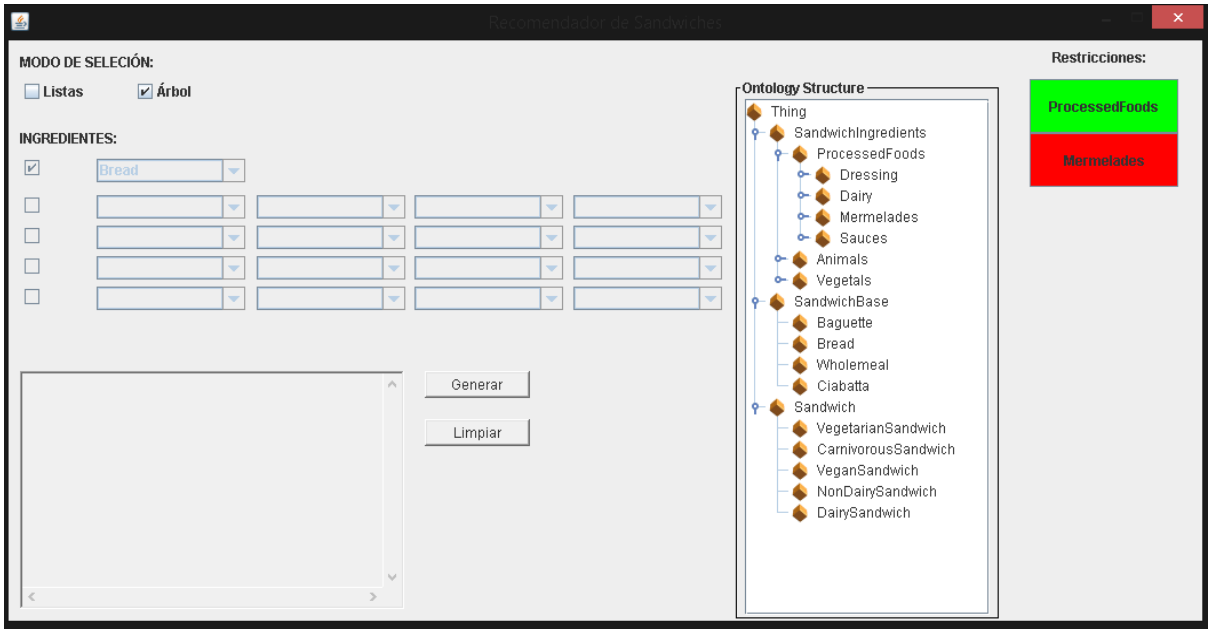

Ilustración 20

v) Posibilidad de borrar una restricción (positiva o negativa) al clickar sobre ella.

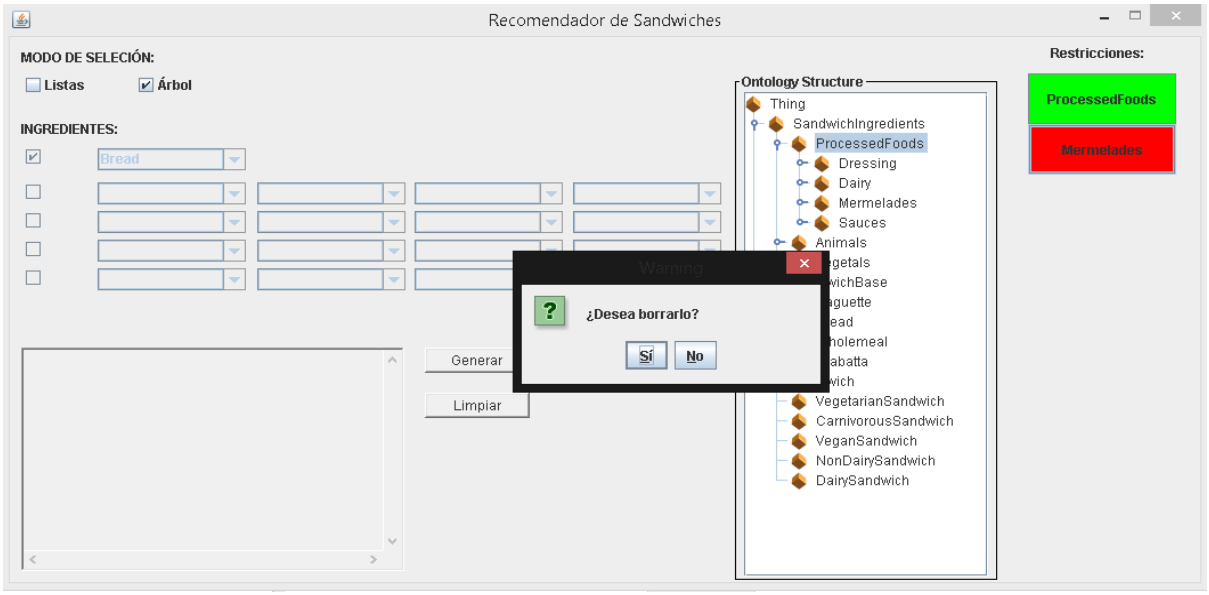

Ilustración 21

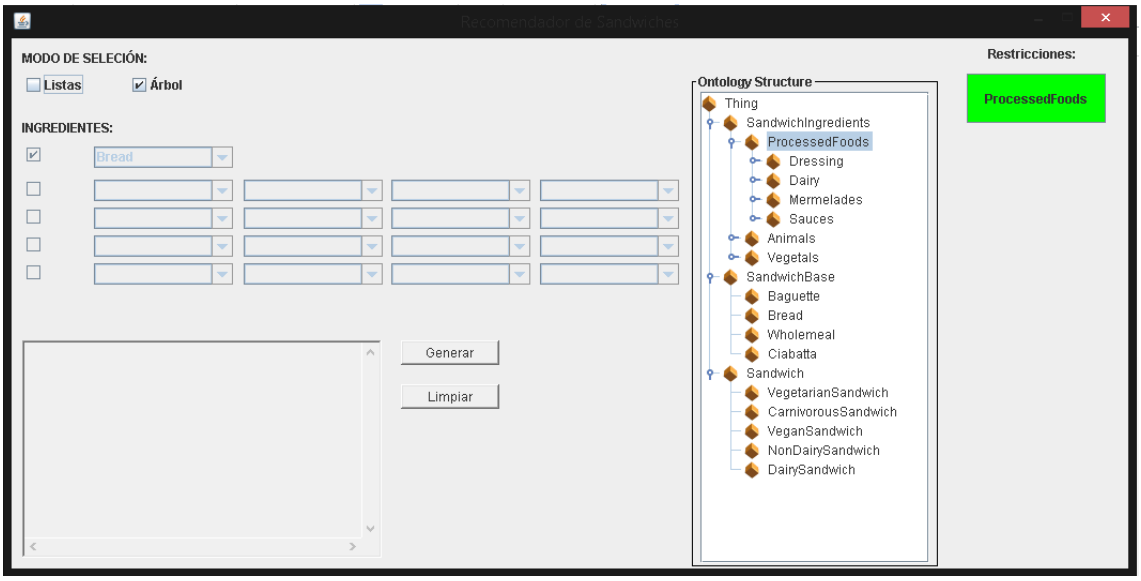

Ilustración 22

Por otro lado, tenemos el botón **‘Generar’**, que pone en funcionamiento el ciclo CBR tomando como consulta los ingredientes seleccionados y el botón **‘Limpiar’**, el cual borra el contenido del ***textArea*** donde se muestran las soluciones.

Ilustración 23

2. Generación de consultas

Atendiendo al siguiente caso (*Ilustración 20*)

Resultado:

Ilustración 22

Restricciones: (Modo de uso, Árbol)

1. Máximo 2 restricciones negativas.
2. No se puede elegir un tipo de sándwich como restricción negativa.
3. Máximo 2 restricciones positivas.
4. Si se elige un tipo de sándwich como restricción positiva, no se pueden añadir más restricciones positivas.
5. Sólo se puede añadir un tipo de pan.
6. No se puede añadir un ingrediente como restricción positiva y negativa al mismo tiempo.
7. Función de similitud: Implementada en la clase ***OntologySimilarityFuntion.java.***

Función de similitud

Restricciones positivas (Ingredientes que se quieren en el sándwich)

```
OntologySimilarityFuntion.java
private void searchIngredients(String ingredientsCase,
ArrayList<String> restrictionsQuery)
```

El valor que represente la similitud se calcula incrementando o decrementando en la evaluación de cada ingrediente el siguiente coste; (1 / número de restricciones), por lo tanto, si el valor es 1 el caso encaja totalmente con la consulta.

Pasos:

i) Si es un tipo de sándwich, por ejemplo, VegetarianSandwich:

No puede contener ningún ingrediente que sea subtipo de la "familia" Animals, por lo tanto, se comprueba que ningún ingrediente del caso pertenezca a la familia Animals, y si es así se asigna directamente una similitud de 1.

ii) Sino, se comprueba que cada ingrediente de la consulta está en el caso.

a) Si el ingrediente actual está en el caso se pasa al siguiente ingrediente de la consulta y se incrementa el valor de la similitud.

b) Si el ingrediente actual de la consulta es una hoja y no está en el caso, se mira si algún hermano está en el caso.

c) Si existe algún hermano en la consulta, este se sustituye por el ingrediente de la consulta y se incrementa el valor de la similitud.

d) Si no existe, se mira si el caso contiene algún hijo de los hermanos de la superclase del ingrediente actual. Si contiene alguno se sustituye y se incrementa el valor de la similitud y sino, se decrementa el valor de la similitud.

e) Si el ingrediente actual de la consulta no es una hoja y no está en el caso, se mira si alguno de sus hijos está en el caso.

f) Si existe algún hijo en la consulta se incrementa el valor de la similitud.

g) Si no existe, se mira si el caso contiene algún hijo de alguno de sus hermanos. Si contiene alguno se sustituye y se incrementa el valor de la similitud y sino, se decrementa el valor de la similitud.

iii) Por último, si el valor de la similitud es 1, se registra ese caso y se devuelve.

Conclusión, todos los casos devueltos van a tener siempre una similitud de 1, los que no la tienen son directamente excluidos, es decir, nunca van a ser devueltos.

Función de sustitución

Restricciones negativas (Ingredientes que se quieren en el sándwich)

```
SandwichRecommender.js
private String negativeRestrictionsFilter(String sandwich)
```

Si hay restricciones negativas en la consulta, se utiliza esta función para filtrar los casos. Funciona de la siguiente forma; se van evaluando una a una las restricciones, si en el caso hay un ingrediente que pertenece a la restricción, este es sustituido por un hermano si la restricción evaluada es una hoja, por otro lado, si la restricción evaluada no es un hoja, se mira si el caso contiene algún hijo “final” (hojas de la rama) de esta restricción, sí es así se sustituye por un hijo “final” de algún hermano de la restricción.