Universidad Complutense de Madrid

Facultad de Informática

Práctica 10
Memoria

Trabajo realizado por

Grupo 1
Ana María Martínez Gómez
Víctor Adolfo Gallego Alcalá

2015

Inteligencia Artificial

Índice general

Descripción de la ontología

En primer lugar, el conocimiento del cliente lo englobamos en dos clases, de forma similar al prototipo anterior de Jess: *Usuario* y *ViajeDeseado*. En la primera contamos con algunos datos del usuario (edad, nacionalidad, número de pasajeros y sexo). Además, cuenta con el multislot viajes_recomendados, donde se almacenarán los viajes recomendados al usuario. En la segunda clase, guardamos la información sobre las preferencias del usuario en cuanto al viaje (clima, duración, origen, el tipo o tipos de viaje que prefiere (culinario, de descanso, urbano o deportivo), su presupuesto máximo y el régimen de comida que desea).

En cuanto al conocimiento sobre los viajes, tenemos una clase *Cadena* con datos sobres las empresas propietarias de los alojamientos: número de empleados (se utiliza en las regla) y un slot múltiple con los alojamientos que posee (no se utiliza en las reglas pero hace que la información en el sistema sea más completa y facilitaría posibles consultas). Contamos con una clase *Alojamiento* que tiene un identificador único para cada establecimiento (es importante que sea único porque se usa en las reglas para evitar los ciclos como se explica aquí), la localización (que es una instancia de *Destino*), el precio por día y persona, la cadena propietaria, los regímenes de comidas que ofrece (desayuno, media pensión, pensión completa y/o todo incluído), el tipo de alojamiento (hotel, albergue, barco, pensión y parador) y los viajes en que se ha ofertado tal alojamiento (al igual que en *Cadena* no se utiliza pero nos pareció coherente incluirlo).

A continuación, la representación geográfica cuenta con tres clases. El *País*, del que conocemos su número de habitantes. La *Región*, con el país al que pertenece y los idiomas hablados (los idiomas no se utilizan en ninguna regla pero nos pareció razonable incluirlo para establecer alguna diferencia entre distintas regiones y el propio *País*). El *Destino* representa una zona dentro de una región que se utiliza como destino y origen de los viajes. Del destino almacenamos su clima (frío o cálido), la región a la que pertenece, el tipo de destino que es (playa, relax, cultura, compras, deporte, aventura, alta cocina o experiencia gastronómica) y los viajes que le tienen como destino (se incluye por la misma razón que los de *Cadena* y *Alojamiento*).

En cuanto a la clase *Transporte*, cuenta con dos subclases: *TransporteTurista* y *TransporteVIP*, que establecen una clasificación que se usa en las reglas para recomendar un tipo de transporte en función del nivel económico del cliente (clientes ricos y pobres). Los transportes contienen la clase (turista en *TransporteTurista* y bussiness o primera en *TransporteVIP*), el origen y destino del transporte (ambas serán instancias de *Destino*), la hora de salida y duración (no se utilizan en las reglas

pero nos proporcionan una información importante acerca del transporte y por tanto nos pareció necesario incluirlo), un identificador único del transporte (con el mismo objetivo que el del *Alojamiento*), y el precio por persona.

Por último, la clase *Viaje* contendrá las instancias de los viajes que se recomienden a los usuarios y que por tanto alberga información sobre el alojamiento, origen, destino, duración y los transportes utilizados tanto para la ida como para la vuelta adecuados para el usuario. También almacena el número de viajeros, el precio total del viaje y el usuario a quién se le ha recomendado el viaje. Este campo no será el inverso del campo en Usuario porque no es necesario, dado que se mantendrá mediante el programa en Jess, tal y como se explica en la siguiente sección.

Además, en las clases contamos con un slot con un nombre para denotarlas. En los slots se han incluido restricciones necesarias para garantizar la coherencia del sistema. Por ejemplo, que los precios y el presupuesto sean mayores o iguales que cero, que la edad sea al menos dieciocho, o que la hora del transporte esté comprendida entre 0 y 2359 (ya que se representa como un Integer). Se añadieron algunos valores por defecto, en su mayoría para agilizar la tarea de crear instancias.

Programa Jess

En primer lugar establecimos la correspondencia entre las clases de Protégé y los templates de Jess mediante *mapclass*.

Conservamos la estructura que teníamos en la práctica anterior de Jess en cuanto a la clasificación de clientes (reglas RtipoClienteJoven, RtipoClienteMayor, RtipoClientePobre, RtipoClienteRico). Es decir, a los usuarios menores de 30 años los clasificamos como jóvenes y a los que tienen al menos 30 como mayores. Por otra parte, los que tienen un presupuesto menor o igual que 5000€ por persona y día los clasificamos como pobres y a los que tienen un presupuesto mayor que dicha cantidad como ricos. Nótese que en la práctica anterior algunos clientes no eran ricos ni pobres, cosa que modificamos para poder recomendar un tipo de transporte diferente en función de si el usuario es rico o pobre como se explicará más adelante.

También conservamos las reglas de clasificación de tipo de viaje (reglas RcategoriaX, con X de 1 a 8) que se le va a recomendar al cliente en función del tipo de viaje que este prefiere y si es joven o mayor. Aunque las reglas son las mismas, hay una clara diferencia con respecto a lo que se hacía en la práctica anterior, pues siguiendo la recomendación de la profesora ahora permitimos desde Protégé que el

cliente tenga más de una preferencia lo que hace que con las reglas se obtenga más de un tipo de viaje.

El núcleo del programa en Jess lo forman las reglas Rrecomendacion y Rrecomendacion2, dado que es donde realmente se recomienda el viaje. Estas reglas no pudimos basarlas en lo que teníamos hecho en Jess debido a las características de la ontología, ya descritas en la primera sección. Para cada usuario se crean y se recomiendan viajes compuestos por: Un alojamiento, en el que esté disponible el régimen de comidas preferido por el usuario, y que se encuentre en un destino cuyo clima y tipo concuerden con los adecuados para el cliente. Un transporte de ida, cuyo origen sea el origen especificado por el cliente y destino el ya mencionado. Un transporte de vuelta, cuyo origen es el destino del transporte de ida y cuyo destino es el origen del usuario. Además, se calcula el precio del viaje (teniendo en cuenta que el precio en el alojamiento es por persona y día y en los transportes por persona) y se comprueba que no exceda el presupuesto. Para generar el nombre del viaje hicimos uso de la función str-cat para concatenar el nombre del usuario, el nombre del destino, el nombre del alojamiento y los identificadores de los transportes. La razón por la que se decidió elegir estos campos es porque resultaba más sencillo reconocer los *Viajes*.

Con los datos mencionados en el párrafo anterior se crea una instancia de Viaje en cuyo campo recomendado_a se guarda el usuario. Añadimos también los viajes en el multislot viajes_recomendados de Usuario ordenados por precio descendente. Esta ordenación se realizó mediante la función calcula, que recibe un precio (el del viaje a insertar) y un multislot con los viajes (ya insertados y ordenados por precio) y devuelve el índice donde se debe insertar el viaje, que será aquel que ahora ocupe el primer viaje cuyo precio sea superior. Se explicará más adelante el motivo de esta ordenación por precio. Por otra parte, en estas reglas fue necesario incluir un assert con el usuario (su referencia, no su nombre, ver último párrafo de esta sección) y los identificadores únicos tanto de alojamiento como de transportes para poder comprobar en la parte izquierda si dicho viaje ya había sido almacenado en el usuario dado que si esto no se hace cada vez que se modifique al usuario se vuelve a acceder a la regla y se entra en un bucle infinito. Nótese que la única diferencia entre las dos reglas es que en la primera se comprueba que el usuario sea pobre y solo se le recomiendan transportes tanto de ida como de vuelta de la subclase *TransporteTurista* y en la segunda se comprueba que sea rico y se recomiendan transportes de la subclase *TransporteVIP*.

Con el objetivo de hacer más interesantes las recomendaciones añadimos dos reglas más. La primera, RlimitacionHabitantesPaís, impide (borrando instancias de viajes ya recomendados) que un usuario, viaje a un destino en un país con población

superior a tres veces la población de su propio país (el de su nacionalidad). Hacer esta regla fue particularmente interesante porque descubrimos que en Protégé los números se desbordan para cifras que no habíamos considerado lo suficientemente grandes como para desbordarse. Por ejemplo multiplicar por tres la población china da un número negativo. Esto lo descubrimos porque en un primer momento habíamos escrito:

```
(test ( (* 3 ?num) ?num2))
```

Esto provocaba que los ciudadanos chinos, a los que en principio no debería afectar esta regla, no pudiesen viajar a ningún destino, incluido su propio país. La solución es sencilla, basta con dividir entre tres en lugar de multiplicar en el otro término, trabajando así con cantidades más pequeñas:

```
(test ( ?num (/ ?num2 3)))
```

En cualquier caso, esto fue interesante porque nos advirtió de los problemas que nos podíamos encontrar en Protégé al trabajar con números grandes.

La segunda regla que añadimos es RalojamientoIntimo, y se hizo con el objetivo de usar más datos, por ejemplo los almacenados en la clase *Cadena*. Pensamos que las mujeres podrían preferir alojamientos más íntimos, por lo que cuando el usuario sea mujer se eliminan todas las instancias de viajes que contengan alojamientos de tipo hotel cuya cadena propietaria tenga un número de empleados mayor a 1000.

Como ya hemos comentado el multislot viajes_recomendados de Usuario se mantiene ordenado por precio decrecientemente. Aunque dicha ordenación resulta de utilidad a la hora de consultar los viajes, la verdadera razón por la que se ordenan es para poder borrar los viajes de menor precio fácilmente cuando se obtienen muchas recomendaciones. Esto se realiza en la regla Rborrado, donde se borran todos los viajes excepto los 4 primeros de cada usuario. Cabe destacar que ha sido necesario establecer una prioridad de -1 para que la aplicación de esta regla no interfiera con las anteriores. Nótese que solo se borran los viajes en los usuarios, pero hemos decidido mantenerlos en viajes para poder realizar comprobaciones. En cualquier caso borrar los viajes que no aparecen en el multislot de su usuario solo implicaría la realización de una sencilla regla más, que se podría implementar por tanto fácilmente si fuese necesario.

Por último, comentar que en muchos de los asserts se utiliza la referencia a un *Usuario* en lugar de su nombre. Esto ocasiona que los assert sean dificiles de comprender pero a cambio podemos tener varios usuarios con el mismo nombre pero que

se consideran diferentes y por tanto se les recomiendan viajes diferentes. Además, el hecho de que los asserts sean difíciles de leer no es algo problemático dado que solo se utilizan internamente.

Objetivos de la integración Jess-Protégé

Ya hemos comentado a lo largo de toda la memoria la forma en la que hemos ido integrando las dos prácticas que teníamos en Jess y en Protégé. La ventaja de esta integración con respecto a usar solo Jess es la facilidad para el usuario de introducir y consultar los datos a través de la interfaz de Protégé mediante formularios, en lugar de tener que utilizar el propio programa en Jess o un programa en Java como hacíamos en prácticas anteriores. También Jess aporta ventajas a Protégé, pues además de las recomendaciones de viajes en sí, aporta coherencia al sistema al facilitar la posibilidad de crear reglas que calculen los precios totales de los viajes. Esto antes debía hacerse sumando a mano los precios de alojamiento y transporte, pudiendo introducir errores. También se puede usar para procesar automáticamente las instancias, por ejemplo con reglas que borren algunas de ellas con respecto a algunos criterios, tal y como hacemos en las reglas RlimitacionHabitantesPais y RalojamientoIntimo. Además, se gestiona mediante reglas en Jess la relación de *inverso de* entre el slot recomendado_a de *Viaje* y viajes_recomendados de *Usuario*, permitiendo así que se inserten los viajes en viajes_recomendados ordenados por precio.

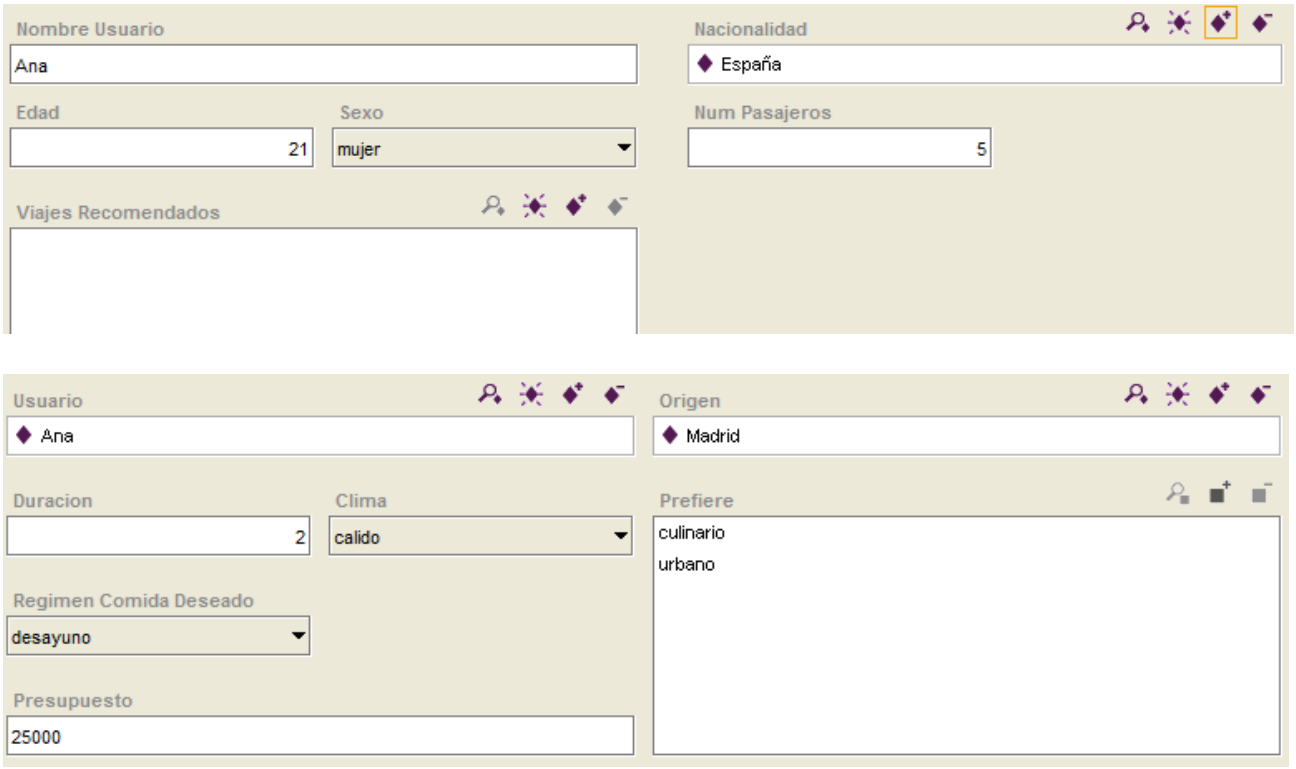
Algo que no hemos comentado y que está directamente relacionado con esta integración es que tuvimos que eliminar muchas de las subclases que teníamos en Protégé, debido a que al usar una superclase en una regla de Jess no se accede a las subclases, sino que es necesario acceder una a una. Esto es de hecho lo que se hace en las reglas Rrecomendacion y Rrecomendacion2, donde consideramos que sí tenía sentido mantener las dos subclases de transportes tal y como se explica más arriba.

Resultados obtenidos

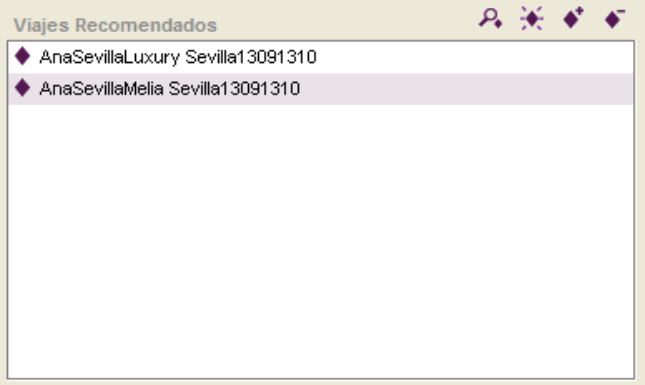
Para probar el prototipo final, contamos con 3 usuarios diferentes para que se vea el efecto de las reglas sobre ello. A continuación se muestran para cada uno sus datos de entrada, y los viajes que les han sido recomendados:

El primer caso es el de Ana:

DATOS DE ENTRADA



VIAJES RECOMENDADOS

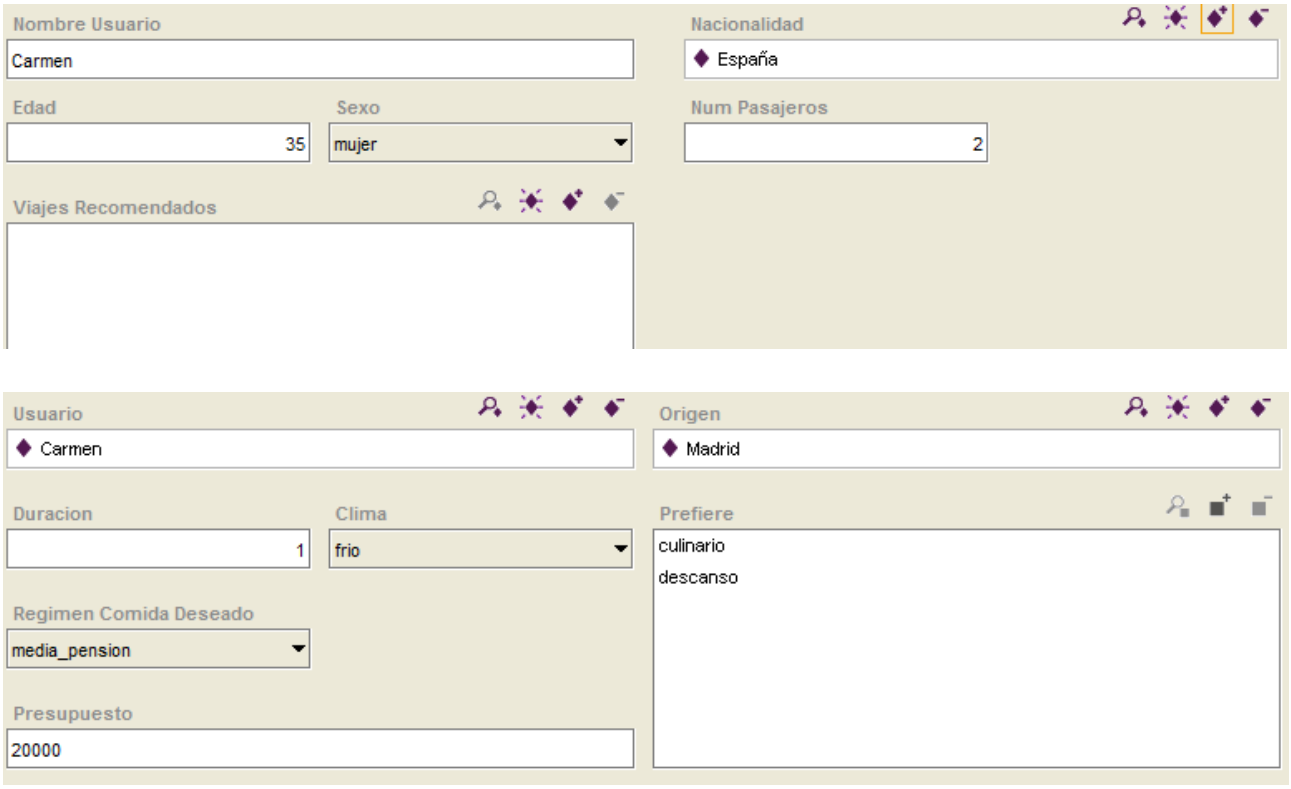


Dadas sus preferencias se la recomienda un tipo de viaje de compras o de experiencia gastronómica (ya que es joven). Por tanto, dado que el clima de Barcelona es cálido, este es un destino idóneo para este usuario. Sin embargo, ninguno de los alojamientos de Barcelona se le recomiendan, pues o bien no cuentan con desayuno (Romantic Hotel Barcelona), o pertenecen a la cadena NH (NHBarcelona) y no se recomiendan por ser Ana mujer y tener NH más de 1000 empleados. Por otra parte, los dos hoteles de Sevilla si que cumplen todas las restricciones y como además disponemos de transportes de ida y vuelta en clase turista es posible recomendar dos viajes.

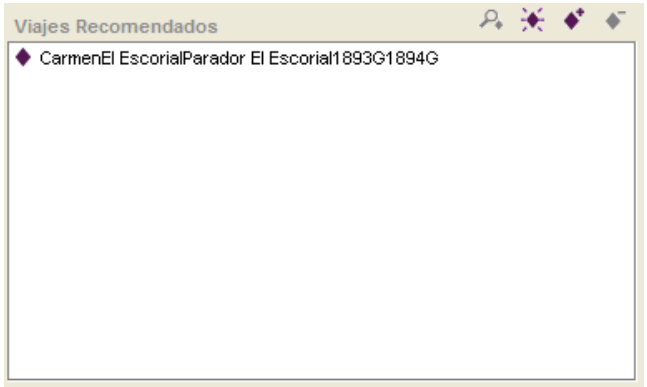
A modo de ejemplo, mostramos una instancia de la clase Viaje concreta:

El segundo caso es el Carmen:

DATOS DE ENTRADA



VIAJES RECOMENDADOS



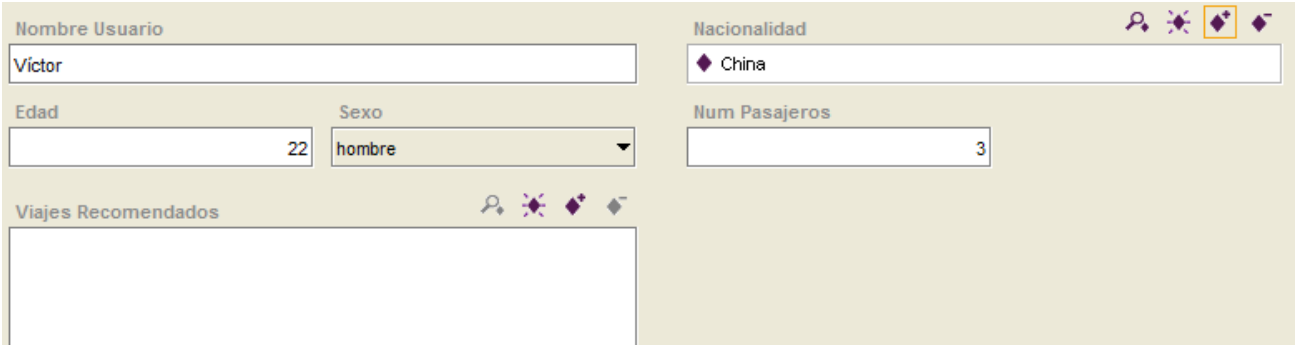
A diferencia del caso anterior, el sistema determinó que Carmen es mayor y rica.

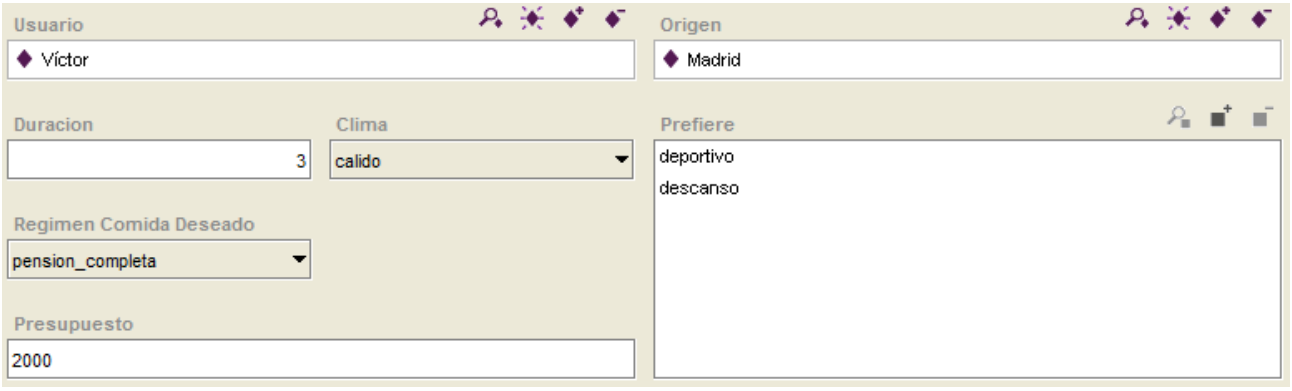
Por tanto, como necesita un transporte VIP, y un destino frío, el único destino que se le recomendó fue el que aparece en la última captura.

Observamos además que en este caso y el anterior, de ninguna manera hubieran podido obtener un destino en China, ya que las dos son españolas (véase la regla RlimitacionHabitantesPais).

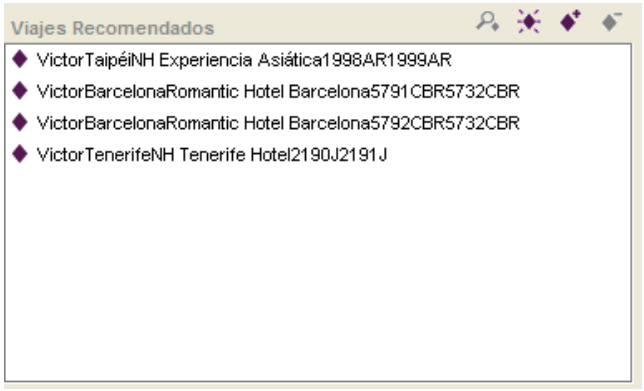
El tercer caso es el de Víctor:

DATOS DE ENTRADA



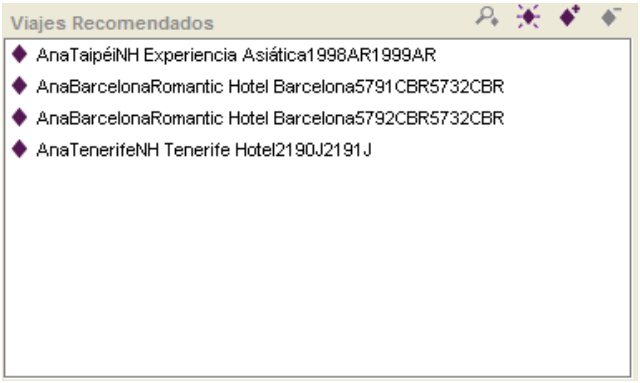


VIAJES RECOMENDADOS



Ahora observamos que se han alcanzado el máximo de viajes que se pueden recomendar (4), de hecho si no existiese esta limitación se habrían recomendado 6 viajes. Dado el tipo de usuario que es Víctor (joven y pobre), y sus preferencias, podría haber obtenido además destinos como Taipéi, sin restricciones como la citada en el caso anterior, ya que Víctor es de nacionalidad china.

Por último, hemos querido ilustrar que se pueden tener usuarios con el mismo nombre, tal y como se explicaba al final de la segunda sección. Hemos mantenido en el sistema solo a Ana y Víctor, y le hemos cambiado el nombre a este último obteniendo dos usuarios con el nombre Ana. Se muestran a continuación como los resultados obtenidos son los mismos:


Viajes Recomendados
AnaTaipéiNH Experiencia Asiática1998AR1999AR
AnaBarcelonaRomantic Hotel Barcelona5791CBR5732CBR
AnaBarcelonaRomantic Hotel Barcelona5792CBR5732CBR
AnaTenerifeNH Tenerife Hotel2190J2191J