Universitat Politècnica de Catalunya

ECSDI

Entrega final

Pau Canosa

Anna Llanza

Laia Ondoño

Q2 2019-2020 *01/06/2020*

ÍNDICE

1. Diseño detallado

En este apartado justificamos y explicamos los protocolos y acciones que usan los agentes para que el sistema funcione.

1.1. Ag. BuscadorProductos

Este agente se encarga de buscar productos en el catálogo con ciertas restricciones que indica el agente asistente. El agente asistente y el agente buscador de productos siguen un protocolo *FIPA Request.* Primero, el agente asistente manda una petición al agente buscador para que se realice una búsqueda de productos. Este realiza la búsqueda aplicando las restricciones y retorna el resultado. Como esta funcionalidad es simple, ningún agente necesita interaccionar con otras entidades, simplemente el agente buscador accederá a la base de datos donde tenemos los productos del catálogo y los elegirá.

1.2. Ag. GestorCompras

El agente gestor de compras es la primera entidad que recibe la información de una nueva compra, este la gestiona y delega parte del trabajo a otros agentes. Primeramente, el agente asistente informa al gestor de compras de una nueva compra utilizando el protocolo FIPA *Request*. Una vez el gestor de compras recibe la información de una nueva compra a realizar, envía una respuesta con la factura del pedido, confirmando, de tal manera, que la compra es viable y se realizará. Más tarde, el agente gestor de compra le enviará al agente asistente toda la información del pedido realizado, incluyendo la fecha de entrega final y el transportista encargado de hacer llegar el pedido.

Sin embargo, antes de enviar la información completa al agente asistente, el agente gestor de compras deberá traspasar la información del pedido al agente centro logístico. Este agente se encargará de enviar el/los productos que forman parte del pedido. Esta comunicación entre agente gestor de compras y centro logístico se realizará vía el protocolo FIPA request: El agente gestor realiza la petición *enviarpedido* hacia el otro agente, y este responde con los datos de la entrega. En ese momento el envío se considera iniciado. Con la información recibida podrá comunicar al agente asistente cuál es su transportista y cuándo llegará el pedido.

Además, una vez se haya cobrado la compra, se encargará de informar al AgProcesadorOpiniones mediante el protocolo FIPA-request, de que el usuario ha realizado la compra de un producto. De esta manera, el AgProcesadorOpiniones ya podrá pedir al usuario que valore el producto.

1.3. Ag. CentroLogístico

El agente CentroLogístico se encarga de gestionar la entrega correspondiente al pedido. Para realizar este proceso, primero recibe mediante el protocolo FIPA Request iniciado por el Agente Gestor de Compra, los datos de la compra que quiere realizar el Agente Asistente. A partir de aquí, mediante el protocolo FIPA Contract-Net, el centro logístico negocia el precio de envío con los respectivos transportistas.

Una vez el precio es acordado, el centro logístico envía un mensaje de confirmación al transportista elegido y posteriormente, envía un mensaje al agente gestor de compra con la nueva información del pedido.

1.4. Ag. GestorDevoluciones

El agente GestorDevoluciones se encarga de la solicitud de devolución de un producto comprado. Para realizar este proceso, primero recibe mediante el protocolo FIPA Request iniciado por el AgenteAsistente, la solicitud de devolución del producto que quiere devolver. Es entonces cuando el AgGestorDevoluciones se encargará de ver si es posible realizar la devolución de ese producto. Una vez haya obtenido una resolución, el AgGestorDevoluciones le enviará al Asistente una respuesta con la información necesaria en el caso de que se acepte la devolución, o con el rechazo de dicha solicitud.

En caso de realizar la devolución, el agente AgGestorDevoluciones se encargará de eliminar del registro de pedidos finalizados el producto a devolver de la compra correspondiente y contactará con el AgServicioPago mediante el protocolo FIPA Request, para que realice una transferencia al agente asistente devolviéndole el precio del producto. Si el producto es de una empresa externa, también le indicará al servicio de pago que realice la transferencia correspondiente.

1.5. Ag. ProcesadorOpiniones

El agente AgProcesadorOpiniones se encarga, por una parte, de gestionar la valoración de un producto comprado por el usuario vía el AgAsistente (pasado cierto tiempo). Cuando un usuario ha realizado una compra el ag procesador de opiniones es avisado de esto por el agente gestor compra. Una vez notificado, este le envía al agente asistente un mensaje en el que se pide que el usuario valore el producto.

El procesador de opiniones, más tarde, recibe la información de usuario, producto, y valoración de este. Una vez recibe estos datos los registra y actualiza el producto que se haya valorado.

Por otra parte, este agente también se encarga de mostrar al AgAsistente posibles productos que le puedan interesar según su historial de compra y de búsqueda. Cada vez que un usuario hace una búsqueda, si el usuario no confirma la compra de los productos encontrados el agente asistente le envía la información al procesador de opiniones. Este lo registra en el historial de búsquedas. Adicionalmente, cuando se realiza una compra, si hay productos que el usuario no ha querido comprar (de los encontrados en la previa búsqueda) se envían al AgProcesadorOpiniones como información para más tarde calcular las recomendaciones.

El AgProcesadorOpiniones envía las recomendaciones al AgAsistente una vez a la semana, para todos los usuarios que hayan hecho como mínimo una búsqueda/compra.

1.6. Ag. GestorProductos

El agente AgGestorProductos es el encargado de añadir los productos que ofrece un vendedor externo al catálogo de productos que ofrece la tienda. Para llevar a cabo esta acción, el AgVendedorExterno le enviará mediante el protocolo FIPA Request, el producto que quiere que sea añadido con sus respectivos datos.

1.7. Justificaciones de diseño y comentarios

Creemos que las decisiones que se han tomado durante el proceso de diseño se deberían justificar, así como citar las posibilidades que hemos barajado detrás de cada decisión. Referente a la distribución de agentes, hemos creído conveniente separar los agentes no solo por sus funcionalidades y con quién se comunican, sino que también teniendo en cuenta distribución geográfica. Es por esto que el agente centro logístico está separado del gestor de compra. En un principio queríamos que hubiera un solo agente que se encargará de la compra, pero nos dimos cuenta que los centros logísticos están distribuidos geográficamente y es por esto que deberíamos separar el proceso de compra.

Otra decisión que nos ha generado dudas ha sido separar o no el buscador de productos y gestor de productos. Primeramente, creíamos que sería mejor tenerlo junto, pero nos dimos cuenta que podíamos separarlo en dos agentes que operaran sobre los mismos datos pero que uno los modificara y otro los leyera. Pensando de

forma global, creímos que así se podría evitar más fácilmente que hubiera cualquier pérdida información, sobre escrituras, etc.

El agente gestor de devoluciones lo hemos tenido claro des del principio, creemos que su comportamiento es muy diferente a todas las otras secciones del sistema y debería ser independiente.

Finalmente, con el agente procesador de opiniones también hemos tenido que discutir un poco. Tal y como hemos dicho anteriormente, este agente se encarga de procesar las valoraciones de los usuarios y recomendar productos a estos utilizando su historial de compras y búsquedas. Hemos decidido juntar estas dos funcionalidades porque creemos que están relacionadas. En ambas funciones el usuario toma parte de protagonismo con su feedback, en un lado dando su opinión, y en el otro utilizando sus intereses para mostrarle productos. Hemos pensado que está suficientemente ligado como para que solo un agente se encargue de estas funcionalidades.

2. Ontología

Este apartado definiremos cada una de las clases de la ontología. Durante todo este apartado, las justificaciones del uso de atributos y relaciones estarán justificados en el mismo apartado..

2.1. Acción

La clase Acción representa las acciones que deberán realizar los agentes del sistema para poder llevar a cabo todos los escenarios. Es por eso que como subclases encontramos primeramente **BuscarProductos**. Esta acción es realizada por el agente BuscadorProductos y se ejecutará siempre que el sistema reciba un conjunto de restricciones por parte del agente asistente, de las cuales hablaremos más adelante. En cuanto haya recibido dichas restricciones, el agente hará una búsqueda entre los productos (clase documentada a continuación) ofrecidos en el catálogo que las cumplan y enviará la información de los productos resultantes al agente asistente que ha realizado la búsqueda.

Respecto al escenario de comprar productos, se especifican dos acciones. La primera, **HacerPedido**, engloba la creación de la compra de los productos del pedido recibido por el agente asistente, además del envío de la factura (**EnviarFactura**). Como resultado, tenemos una instancia de la compra generada. Esta acción la ejecuta el agente AgGestorCompra.

Paralelamente al envío de la factura al agente asistente, mediante la acción **ProcesarEnvio**, el AgGestorCompra asignará la compra al centro logístico más cercano a la ciudad del cliente y le encargará al agente AgCentroLogistico el envío de los productos de la compra a la dirección indicada por el AgenteAsistente. Para esta entrega, el centro logístico elegido será el más cercano al cliente, siempre que sea posible. El centro logístico asignado creará un lote que contendrá los productos de la compra. Una vez se haya generado el lote, se realizará la acción **PedirPreciosEnvio**, que seguirá el protocolo de FIPA Contract-Net, en la cual el agente AgCentroLogistico se pondrá en contacto con varios transportistas indicando el lote que desea enviar para así obtener, como respuesta, las ofertas de los precios de envio correspondientes al lote generado por la compra. Será entonces cuando el centro logístico realice una contraoferta mediante la acción **PedirContraofertasPreciosEnvio** con un precio ligeramente inferior a la oferta mínima recibida por los transportistas. Los transportistas replicarán rechazándola o aceptándola proponiendo otra inferior a su precio inicial. El centro logístico elegirá al transportista que haya propuesta el precio más barato o, en caso de que ningún transportista haya aceptado la contraoferta, elegirá al transportista que inicialmente ofreció el precio más barato. Con la acción **EnviarPaquete**, el AgCentroLogistico

confirmará al transportista para que éste se encargue de entregar el paquete. Además, el mismo agente le enviará al AgGestorCompra la información completa de la compra con los datos que se han conocido recientemente (fecha de entrega y nombre del transportista). Si hay productos en la compra que son ofrecidos por vendedores externos, entonces el AgCentroLogistico enviará un mensaje a cada vendedor para informar de que su(s) producto(s) ya están en camino (**AvisarEnvio**).

Para finalizar la compra, el agente AgGestorCompra realizará la acción **CobrarCompra** en la cual se encargará de informar al AgServicioPago de una transferencia que debe realizar con la cual el agente asistente pagará la compra a la tienda. Si la compra contiene productos de un vendedor externo, contactará con el AgVendedorExterno para obtener la cuenta bancaria a la que debe realizar la transferencia, mediante la acción **PagarVendedorExterno**. A continuación, informará al AgServicioPago de que realice la transferencia de la tienda a la empresa externa correspondiente al valor de los precios de los productos.

Cuando el agente asistente indique que desea realizar la devolución de un producto, le enviará a la tienda una solicitud de devolución. Esto inicia la acción **DevolverProducto**, con la cual el agente AgGestorDevoluciones posteriormente se dispondrá a procesarla. Una vez haya obtenido una resolución, contactará con el agente asistente. En el caso de que el AgGestorDevoluciones no acepte la devolución del producto, se informará al AgAsistente de que no se le devolverá el dinero. En caso contrario, el sistema le enviará al agente asistente la información necesaria para que devuelva el producto. El AgAsistente realizará la acción **FinalizarDevolucion**, en la que indicará que ya ha entregado el producto a devolver a la empresa de mensajería encargada de devolverlo al centro logístico. Cuando el AgGestorDevoluciones reciba esta confirmación, procederá a eliminar del registro de pedidos finalizados el producto a devolver de la compra correspondiente y empezará la acción **DevolverDinero** y contactará con el AgServicioPago para que realice una transferencia al agente asistente devolviéndole el precio del producto. Si el producto es de una empresa externa, también le indicará al servicio de pago de que realice la transferencia correspondiente, después de haber realizado la acción **CobrarVendedorExterno** en la que pedirá al vendedor externo que le indique a qué cuenta bancaria debe realizar la transferencia.

En cuanto haya pasado cierto tiempo después de la entrega del producto, el agente AgProcesadorOpiniones iniciará la acción **ValorarProducto** en la que le pedirá al agente asistente que realice una valoración de un producto comprado. Si éste desea valorar el producto, el sistema añadirá la valoración de dicho producto.

Periódicamente, el agente AgProcesadorOpiniones iniciará la acción **RecomendarProductos**, en la que seleccionará productos del catálogo que

considere que pueden interesar al agente asistente (basándose en su historial de compra y de búsqueda) y se los enviará.

En la acción **AñadirProductoExterno**, el agente AgGestorProductos recibe la información de un nuevo producto ofrecido por un vendedor externo. Es entonces cuando lo añade al catálogo de la tienda.

La acción **ActualizarHistorial** podrá iniciarse con dos acciones distintas. Por una parte, cuando el usuario haya realizado una búsqueda de productos pero decida volver a la página inicial, el agente asistente le enviará al AgProcesadorOpiniones el usuario que ha realizado la búsqueda y que productos se han encontrado según sus restricciones. Por otra parte, si el usuario decide comprar algún producto, la información que se le envíe al AgProcesadorOpiniones consistirá en los productos de la búsqueda que ha decidido no comprar.

2.2. Producto

La clase Producto es una de las más importantes. Define las características de cada producto del catálogo y contiene relaciones con otras entidades del sistema. Para diferenciar cada producto de otro, la clase producto tiene el atributo Identificador de tipo string. Además, tiene una Marca (string), un Nombre (string), un PrecioProducto (float), una Valoración (float), una CantidadValoraciones (int), una Categoria (string) y un Peso (float).

Producto está relacionada con la clase Compra mediante la relación llamada ProductosCompra, la cual indica los productos que incluye cada compra. De la misma forma existe la relación ProductosFactura. También se relaciona Producto con la acción AñadirProductExterno, que identifica el producto que se está añadiendo. Adicionalmente, un lote contiene productos, y se especifica con la relación Contiene. Un producto puede estar ofrecido por una empresa externa, en este caso se instancia con la relación Ofrece. Cuando se quiere devolver un producto, se instancia con la relación ProductoADevolver. Ésta relaciona la acción FinalizarDevolucion con DevolverProducto. Cuando se tiene que actualizar el historial de búsquedas, se pasan unos productos, estos forman parte de la relación ProductosHistorial, en la que interviene la acción ActualizarHistorial, y el producto en si. La relación ProductoRecomendado la cual relaciona esta clase con la subclase de Acción RecomendarProductos, sirve para indicar al agente asistente los productos recomendados. Por último, ProductosPedido que relaciona Producto y la clase HacerPedido para indicar los productos que tiene cada pedido.

Esta clase será accedida por los agentes BuscadorProductos y GestorCompras. El primero cuando deba realizar la búsqueda de productos con las restricciones del

agente asistente, y el segundo para cuando se necesita el precio para realizar la factura.

2.3. Compra

La clase Compra, representa la compra de ciertos productos que hace el agente asistente a la Tienda. Para distinguir cada Compra, usamos el atributo Identificador (string). También disponemos del atributo Ciudad (string), la cual indica la ciudad donde se ha de hacer la entrega del pedido. El atributo Fecha (dateTime), que indica la fecha de entrega del pedido. El atributo PrecioTotal (float), que indica el precio de la compra, es decir, la suma del precio de todos los PreciosProductos más el precio del envío. El atributo PrioridadEntrega (int) indica el nivel de la prioridad de su entrega, el cual va de un rango del 1 al 3, siendo el 1 el de mayor prioridad y el 3 el de menor. El atributo TarjetaCredito (int) indica el número de la tarjeta de crédito del agente asistente con el que se realizará el cobro de la compra.Finalmente el atributo NombreCL indica el nombre del centro logístico que se encargará de gestionar el envío y desde donde saldrá el producto.

Compra está relacionada con 4 clases diferentes. Una de sus relaciones es HechaPor vinculada con la clase Usuario, que sirve para identificar el usuario que ha realizado cada una de las compras. También tiene Envia que relaciona la clase Compra con la subclase de Acción de ProcesarEnvio, la cual sirve para indicar la compra que se está procesando en el envío. Compra se relaciona, también, con la acción finalizar devolución y devolver producto. En esta relación llamada CompraDevolucion se referencia la compra sobre la cual se están haciendo dichas acciones. Por último, tiene la relación ProductosCompra que relaciona la Compra con el Producto. Está sirve para indicar los productos que incluye cada compra.

Esta clase será accedida por los agentes GestorCompras y CentroLogistico. El primero cuando crea una instancia de Compra y le envía los datos de la compra al CentroLogistico, el cual necesita saber los datos de la compra para poder asignar el centro logístico que realizará la entrega de los productos de la compra.

2.4. Factura

La clase Factura representa la factura de una compra, la cual será enviada al agente asistente una vez se haya creado la compra en el sistema. Tendrá el atributo Identificador, un string con el cual se identificará cada instancia de esta clase. De forma similar a la compra, también tendrá los atributos ciudad , numero productos, precio total, prioridad entrega y tarjeta de credito. Como relaciones, tiene *FacturaEnviada* que la relaciona con la accion EnviarFactura, y también tiene la relación *ProductosFactura* que relacionará los productos de la factura con esta.Esta

clase sólo será accedida por el agente GestorCompras, pues es el único que necesita esta información para enviarle al agente asistente la factura correspondiente al pedido.

2.5. Centro Logístico

Esta clase representa los centros logísticos que almacenan los productos ofrecidos por la tienda. Es la encargada de realizar todas las acciones delegadas a cada centro logístico. Para poder representarla correctamente, dicha clase contendrá los siguientes atributos. El atributo Identificador (string) es el atributo que identificará cada centro logístico del sistema, además de un string Ciudad que contendrá la ciudad en la que se encuentre el centro logístico. Este atributo se usará para determinar si el centro logístico es el más cercano a la ciudad destino de un pedido a la hora de asignar un centro logístico a la entrega de éste.

2.6. Lote

Esta clase representa los lotes de productos que se distribuyen entre transportistas para una entrega. Como atributos simples, Lote tiene una ciudad, en forma de string, que es la ciudad destino donde se debe enviar el lote. También tiene un identificador, para una mejor gestión de todos los lotes. Además, en todo lote consta una prioridad de entrega. Finalmente tiene un atributo llamado peso, que sirve como información para contactar con ciertos transportistas en el momento de negociar el envío, y el atributo NombreCL que indica la ciudad del centro logístico donde se va a enviar. Sin embargo, existen dos *Object Property* que relacionan los lotes con otras entidades. La primera relaciona cada lote con todos los productos que contiene. Esta relación se llama *Contiene* , y los dos integrantes son Lote y Compra. Hay que matizar que el objeto Compra, puede ser dos cosas: Puede ser una compra completa, es decir, un pedido que se puede enviar con un mismo transportista y la puede distribuir un solo centro logístico. También puede ser una porción de una compra más grande. En algunas ocasiones, un pedido involucra más de un producto que solo está disponible en diferentes centros logísticos. Cuando pasa esto, la compra se divide en partes que se envían individualmente, estas partes también las denominamos compras, y son las que procesan los centros logísticos.

Otra relación que tenemos es *Lote* que relaciona la acción PedirPreciosEnvio con el lote que se tiene que enviar. Una vez el Lote se ha entregado, necesitamos usar la relación LoteEntregado con la clase CobrarCompra.Adicionalmente, tenemos la relación LoteContraOfertas, que relaciona la acción pedir contra ofertas precios envio con un Lote. Finalmente la relación LoteFinal relaciona la acción EnviarPaquete con el Lote.

Esta clase será accedida por el agente CentroLogistico cuando éste deba agrupar las compras en lotes y posteriormente cuando el agente elija qué lotes enviar al servicio de transporte para la entrega de sus productos.

2.7. Transportista

Esta clase representa los transportistas que se encargan de entregar los lotes que se les asignan. Como atributos simples, tiene un identificador, para poder gestionar cada uno de los transportistas existentes, también tiene un nombre, que sirve como información general y que se utiliza para generar el informe de pedido completo para el agente asistente.Dispone de dos atributos más, PrecioKg y PrecioKm que sirven para calcular el precio del envio.

Esta clase es accedida por el agente CentroLogistico, ya que se encargará de asignar un transportista para la entrega de cada lote.

2.8. Restricciones

Esta clase identifica las restricciones que el agente asistente indica para poder realizar la búsquedas de productos. Esta clase tiene un *Object Property* llamado *RestringidaPor* con la acción BuscarProductos. Básicamente indica que restricciones tener en cuenta para una búsqueda específica. Hemos identificado 4 diferentes restricciones que se pueden instanciar:

- RestriccionMarca: Tiene un atributo String, que identifica la marca.
- RestriccionNombre: Tiene un atributo String, que identifica el nombre de producto con el que se quiere filtrar la búsqueda.
- RestriccionPrecio: Tiene dos atributos int para indicar el precio mínimo y el precio máximo que queremos que tenga lo que estamos buscando.
- RestriccionValoracion: Tiene un atributo int, que representa la valoración que tiene que tener el producto que se busque. Puede ir del 1 al 5.

Esta clase la accederá el agente buscador de productos, que instanciará las restricciones que le pase el agente asistente y las utilizará para obtener un resultado de productos.

2.9 Usuario

Esta clase representa el usuario de un usuario del sistema. El usuario es, en la mayoría de funcionalidades, quién desencadenará una acción en el sistema. Para identificar los usuarios, cada uno se identificará por el DNI. La relación HechaPor relacionará al usuario con cada compra que haga. La relación HistorialDe une la acción ActualizarHistorial con esta clase para indicar el usuario del historial que se

debe actualizar. Finalmente, la relación DevueltoPor relaciona el usuario que quiere devolver un producto con la acción finalizar devolución.

2.10 Vendedor externo

Esta clase representa un vendedor externo. El sistema tendrá un identificador además del nombre de la empresa externa y su cuenta bancaria. Como en algunos casos se necesitará, también se guardará qué productos que ofrece son de un vendedor externo con la Object Property Ofrece.

3. Descripción de la implementación

Nuestro sistema permite a un agente asistente realizar un conjunto de acciones. Para ello, el usuario deberá introducir el DNI, que le identificará como usuario dentro del sistema. Una vez autentificado, el usuario va a poder indicar unas restricciones para buscar productos que posteriormente podrá comprar (si está interesado). Además podrá valorarlos una vez los haya recibido y devolverlos en caso de que éste no los quiera, indicando, en ese caso, el motivo. El sistema también se encargará de recomendar un seguido de productos que considera que le podrían gustar al usuario, partiendo de las búsquedas que ha realizado y de su historial de compra. Nuestro sistema también permite que ciertas empresas externas añaden productos al catálogo para poder ser comprados.

De todo el conjunto de código que utilizamos para que el sistema funcione, creemos que es importante destacar ciertos recursos que utilizamos. Primeramente, utilizamos las funciones *Build Message* y *Send Message* que nos ayudan a crear mensajes con diferentes performativas FIPA-ACL para que los agentes se comuniquen. Estas funciones también nos permiten adjuntar un grafo con el contenido respectivo. Otra herramienta que hemos utilizado bastante es *Namespace* de la librería *rdflib*. Esta herramienta nos ha servido para definir los conceptos de la ontologia. La librería *Multiprocessing* nos ha sido de gran ayuda para poder generar procesos que se ejecuten concurrentemente y así paralelizar las tareas que tiene cada uno de los agente. Ésto ha sido útil, especialmente, cuando necesitábamos que un agente realizará otra tarea antes de finalizar con lo que estaba haciendo. También hemos hecho uso de *Flask*, un framework que ha sido vital para el funcionamiento del proyecto. Con él, hemos especificado todas las rutas de cada uno de los agentes implementados. Además, ha sido útil para redirigir páginas siempre que ha estado necesario.Finalmente, hemos utilizado *Graph*, también de la libreria *RDFLIB* que nos permite crear grafos e insertar contenido en ellos. Los grafos son el medio de comunicación principal.

Aunque cuando hicimos el diseño detallado del sistema vimos claramente que agentes éste debía de incorporar, no tuvimos en cuenta los agentes que tomaban parte como actores externos. Estos actores externos son entidades que interactúan con el sistema enviándose información para informar o obtener resultados que desean. Para poder tener un proyecto completo, encontramos necesario implementar los siguientes actores externos como agentes.

El agente externo más importante, y que vimos que era esencial implementarlo, es el agente asistente. Esto es porque el sistema, que simula una tienda, está centrado en sus usuarios y, consecuentemente, consideramos que de alguna manera

debíamos simular la interacción entre usuario y sistema. Este agente facilita la comunicación entre dichos sujetos en todas y cada una de las funcionalidades que el sistema ofrece.

También forma parte del sistema, como agente externo, el AgVendedorExterno, el cual es necesario ya que representa a un vendedor externo que forma parte de una empresa externa en nuestro sistema que quiere añadir a nuestro catálogo un producto para que este se pueda tener en cuenta en la venta de productos.

El AgServicioPago, es otro agente externo con el que el sistema interactúa. Este es necesario para realizar todas las transferencias de pago o cobro de productos en las compras y devoluciones de productos.

Por último, como agente externo considerado en este proyecto tenemos al AgTransportista. Éste está involucrado en el proceso de envío, el cual se encarga de calcular la oferta del precio según los datos que recibe del AgCentroLogistico.

A continuación, explicaremos con más detalle la implementación de cada una de estas funcionalidades, mencionando, entre otros, los procesos que realizan cada agente y los mensajes intercambiados (con su contenido) entre éstos. Las justificaciones y comentarios sobre dicha implementación se encontrarán al final de cada subapartado.

3.1. Buscar productos

Esta acción la inicia el usuario cuando desea buscar productos en el catálogo que cumplan con ciertas restricciones que él mismo indique. Para hacerlo, tendrá disponible un formulario en /search_products en el que podrá rellenar la información de la búsqueda con los campos que servirán para indicarle al AgBuscadorProductos las restricciones de búsqueda y limitar los resultados. Estos campos están formados por el nombre del producto, un rango para el precio (precio mínimo y precio máximo), una marca y una valoración. El usuario no tiene porque rellenarlos todos.

Una vez haya enviado el formulario, el agente AgAsistente procederá a generar el grafo con las restricciones indicadas para enviarlo al AgBuscadorProductos. Este grafo, que contendrá la acción BuscarProductos, estará relacionado con cada uno de los nodos que representan las restricciones indicadas anteriormente por el usuario mediante la relación RestringidaPor. Cuando se le envíe el grafo al AgBuscadorProductos, éste pasará las restricciones del grafo a un diccionario para simplificar su contenido y lectura. Será entonces cuando este agente buscará, realizando una SPARQL query aplicando las restricciones, dentro de los 2 ficheros donde está la información de los productos del catálogo (Data/Productos y

Data/ProductosExternos). A partir del grafo resultante de la query, creará otro grafo que contendrá los productos con toda su información (identificador, nombre, marca, categoría, precio, peso, valoración y sujeto) para que el AgAsistente obtenga su respuesta.

Una vez el AgAsistente haya recibido los productos resultantes de la búsqueda, transformará el grafo en un vector de productos donde cada posición será un diccionario que contendrá la información de cada producto. Este vector se pasará como parámetro en la redirección a /hacer_pedido, haciendo uso de Flask. El usuario podrá visualizar la lista de productos con su nombre, marca y precio dándole la opción de que pueda marcar los que desee y comprarlos. Si no desea comprar ningún producto, podrá volver al inicio, y será entonces cuando el AgAsistente envíe al AgProcesadorOpiniones los productos resultantes de la búsqueda (relación ProductosHistorial) además del nombre de usuario (relación HistorialDe) incluidos en un grafo con la acción ActualizarHistorial para que este agente, una vez lo haya recibido, añada la información a Data/Historial.

Las decisiones más importantes que se han tomado en la implementación de este agente han estado, primeramente, como introducir las restricciones. Hemos optado por un form en un html porque creemos que es la opción más sencilla y comprensible para cualquier usuario.

Por otra parte, una vez se muestran los productos de la búsqueda, hemos creído conveniente que el usuario pueda escoger los productos. Hemos tomado esta decisión porque creemos que es más sensato escoger que determinar si la compra se hace o no solo con un botón confirmar. Para nosotros tiene más sentido que el usuario confirme la compra señalando lo que quiere. En el caso que solo pudiera confirmar o rechazar, el proceso para comprar productos sería mucho más lento y pesado.

Por último, y referente a la información enviada al agente procesador de opiniones, hemos creido que tenía valor que si había productos que el usuario no escogiera de una compra, se añadieran como información a procesar, porque aunque no los haya escogido, son productos que ha buscado y han salido como resultado.

3.2. Comprar productos

Esta acción solo se podrá iniciar una vez el AgAsistente haya realizado una búsqueda de productos y se hayan encontrado productos que, como hemos comentado en el apartado anterior, se verán listados en /hacer_pedido. El usuario tendrá la opción de marcar los productos que quiera comprar y deberá rellenar un formulario con 3 campos obligatorios: una ciudad de entrega, una prioridad de

entrega y una tarjeta de crédito con la cual, al recibir los productos, se pagará la compra. La prioridad de entrega podrá tener los siguientes valores: 1 para la más alta (entrega en un día), 2 (entrega entre 3 y 5 días) o 3 para la más baja (entrega entre 5 y 10 días).

Cuando envíe el formulario correctamente, el AgAsistente enviará dos mensajes paralelamente. En el primer mensaje, enviará al AgProcesadorOpiniones un grafo de la acción ActualizarHistorial con los productos de la búsqueda que el usuario no ha decidido comprar (relación ProductosHistorial) y el nombre de usuario (relación HistorialDe) para que este agente los añada a historial del usuario correspondiente. Cuando el AgProcesadorOpiniones lo reciba, añadirá esta información al fichero Data/Historial junto a los productos buscados previamente por el mismo usuario. En el segundo mensaje, el AgAsistente generará un grafo con la información necesaria para la compra, incluyendo, a parte de la acción que el AgGestorCompra debe realizar (HacerPedido), la ciudad y la prioridad de entrega, la tarjeta de crédito, el nombre de usuario (relación HechaPor) y la lista de productos a comprar (ProductosPedido). Este grafo será enviado al AgGestorCompra, quien, una vez lo haya recibido, irá leyendo los datos y, paralelamente, rellenará la factura en otro grafo que se distinguirá del anterior, sobretodo, porque contendrá el precio total de la compra. Antes de enviar la factura al AgAsistente, creará y empezará un proceso (haciendo uso de la clase Process) donde se dará paso a procesar la compra. En este punto, la ejecución se divide en dos procesos paralelos que explicaremos a continuación.

Por una parte, el AgAsistente recibirá la factura y procederá a mostrar sus datos. Inicialmente, solo se mostrará la lista de productos comprados, la ciudad y prioridad de entrega, la tarjeta de crédito y el precio total de la compra. El usuario tendrá disponible un botón con el que podrá consultar si el AgAsistente dispone de la factura completa, es decir, la factura que incorpora la fecha final de entrega y el nombre del transportista. En caso de que aún no se conozca esta información, se le mostrará un mensaje informándole de ello. Si el AgAsistente ya ha recibido estos datos del AgGestorCompra, se procederá a recargar la página y mostrar la factura completa con los datos conocidos recientemente. Una vez pueda ver la factura final, únicamente podrá volver a la página inicial para realizar otras acciones.

Por otra parte, se estará realizando el proceso que tiene como objetivo el envío de los productos comprados al usuario. Primero de todo, el agente AgGestorcompra elegirá el centro logístico al que le pasará la compra para que gestione su transporte. Para esto, calculará la distancia de la ciudad de entrega a cada uno de los 3 centros logísticos (Barcelona, Pekín y Nueva York) para así asignar el que esté más cerca. Tras elegirlo, procederá a generar un grafo correspondiente a la acción ProcesarEnvio que estará relacionada con una compra mediante la relación Envia.

Esta compra tendrá todos sus atributos (Identificador, Ciudad, PrecioTotal, PrioridadEntrega, TarjetaCredito y NombreCL), además de la relación ProductosCompra con los productos del pedido. Una vez haya generado la compra, se la enviará al AgCentroLogistico para que la procese.

Cuando el AgCentroLogístico recibe la compra que el AgGestorCompra le ha enviado, este prosigue a asignar un transportista a la entrega. Para ello, crea un grafo en el que añade el lote (con la información necesaria, como el nombre del centro logístico) que contiene todos los productos de la compra (especificado con la relación Contiene), su peso total y la información de entrega (ciudad y prioridad) y se lo envía al AgTransportista (acción PedirPreciosEnvio) para que le responda con una serie de ofertas de diferentes transportistas de ese centro logístico. Haciéndolo de esta manera, a parte de evitar tener varios agentes transportistas, conseguimos que un único agente transportista pueda gestionar toda la negociación de los transportistas con el centro logístico indicado.

El AgTransportista generará un grafo en el que se añadirán los nodos que contengan la fecha prevista de entrega y el precio de envío ofrecido por cada transportista, además de su nombre e identificador. Para calcular la fecha de entrega, mirará la prioridad y, en función del valor que tenga, se calculará la fecha prevista de llegada del pedido. Para el precio de envío, se usarán 3 vectores (uno para cada centro logístico) en los que se tendrán guardados todos los datos de cada uno de los transportistas de ese centro logístico, incluyendo el identificador, el nombre, el precio/kg y el precio/km. El AgTransportista calculará el precio en función de estos datos y de los recibidos en la request y lo añadirá al grafo que enviará como respuesta al AgCentroLogistico.

Cuando el AgCentroLogistico reciba las ofertas, calculará una contraoferta (atributo PrecioTransporte) que será un poco más barata a la oferta mínima que ha recibido y la añadirá al grafo, además del lote correspondiente, y enviará de vuelta al AgTransportista a través de la acción PedirContraofertasPrecioEnvio. El AgTransportista creará un grafo en el que indicará la respuesta a esta contraoferta para cada transportista. Si éste decide aceptarla, el grafo contendrá un nodo donde estará el nuevo precio de envío (calculado rebajando ligeramente el precio que ofreció inicialmente). En caso contrario, el grafo no tendrá este nodo y sólo contendrá la respuesta (nodo con la acción) que supondrá que rechaza la contraoferta. El AgCentroLogistico se quedará con la contraoferta más barata y, si ningún transportista la acepta, con la más barata de las ofertas iniciales.

Para finalizar, el AgCentroLogistico generará un grafo que representará la acción EnviarPaquete, la cual tendrá un atributo Identificador (del transportista elegido) y una relación con el lote final, y el AgTransportista le responderá conforme ha

recibido esta información. Si la compra contiene productos de vendedores externos, una vez el AgCentroLogistico haya enviado el mensaje EnviarPaquete, deberá avisar, paralelamente, a los correspondientes vendedores externos para avisarles de que uno o varios productos suyos han sido enviados. Para hacerlo, creará un proceso (Process) en el que generará un grafo de la acción AvisarEnvio y añadirá una relación con los productos de cada vendedor que se hayan enviado (relación ProductoExternoEnviado). Este grafo lo enviará al AgVendedorExterno, quien responderá una vez lo haya recibido con un grafo vacío.

Tras recibir la confirmación del AgTransportista conforme ha recibido el encargo, el AgCentroLogistico sumará el precio del envío al precio total, añadirá la fecha de entrega, el nombre del transportista y el lote al grafo recibido por el AgGestorCompra y se lo volverá a enviar. Será entonces cuando se añadirá la compra al fichero Data/RegistroPedidos y el AgGestorCompra añadirá la fecha de entrega y el nombre del transportista a un grafo que contendrá el resto de datos de la factura y se lo enviará al agente AgAsistente.

El AgTransportista creará un proceso paralelo en el que contará el transcurso del tiempo hasta que el pedido haya sido entregado al usuario. Cuando esta cuenta atrás acabe, le enviará un mensaje con la acción CobrarCompra y el lote que ha entregado (relación LoteEntregado) al AgCentroLogistico, quien lo reenviará al AgGestorCompra. Al recibir el grafo con la acción CobrarCompra, generará un grafo que contenga: la tarjeta de crédito que deberá pagar la compra, la cuenta de la tienda a la que realizar la transferencia, el importe de dicha transferencia, el identificador de la compra y el nombre que identifica al usuario (DNI). Este grafo será enviado al AgServicioPago, quién realizará la transferencia (escribiendo los datos en el fichero Data/RegistroEconomico) y responderá al AgGestorCompra una vez lo haya hecho.

Si hay productos que son ofrecidos por vendedores externos, se sumará para cada vendedor los precios de sus productos comprados y se procederá a pagar lo que corresponda a todos los vendedores. Para cada vendedor externo se realizará lo siguiente: el AgGestorCompra realizará la acción PagarVendedorExterno donde pedirá al AgVendedorExterno la cuenta bancaria del vendedor externo al que se le debe cobrar el importe. Lo hará enviándole un grafo con la acción PagarVendedorExterno y el nombre del vendedor externo del cual necesita esta información. El AgVendedorExterno le enviará un grafo como respuesta con la misma acción y un nodo que contenga la cuenta bancaria correspondiente. Tras conocer esta información, se generará un nuevo grafo con los nuevos datos y se le enviará al AgServicioPago. Este le responderá con un grafo vacío para indicarle que ya ha realizado el pago y, consecuentemente, se ha registrado en Data/RegistroEconomico.

Creemos que en este apartado no hay mucho que justificar, pues lo hemos detallado todo lo máximo que hemos podido. Sin embargo, creemos que podemos justificar el porqué hemos decidido añadir el precio de envío al precio total, pues se podría pensar que es más lógico que el precio de envío tenga valores predeterminados dependiendo de la prioridad. Nosotros, hemos diseñado un sistema que no se centra en un tipo de producto y de zona geográfica, es por esto que hemos creído conveniente que el precio del envio no sea prefijado. Si hubiéramos tenido que diseñar un sistema que solo operase en la Península Ibérica y sólo enviara móviles, si que hubiera sido sensato prefijar unos precios.

Referente a los centros logísticos y los transportistas, hemos decidido generar solo un agente que se comporte como distintas instancias de estos, para no tener demasiados agentes activos a la vez (uno por instancia). En un sistema “real” si que se debería separar, y sería muy fácil de modificar.

3.3. Valorar productos

Cuando se haya pagado una compra y el AgGestorCompra haya recibido la confirmación por parte del AgServicioPago, le enviará un mensaje al AgProcesadorOpiniones avisándole de que ya se han entregado los productos (incluidos en el grafo del mensaje) de esa compra. Tras un período de tiempo breve, el AgProcesadorOpiniones le enviará una FIPA-ACL request al AgAsistente para que éste dé la opción de valorar los productos comprados al usuario. Cuando el AgAsistente reciba la petición, le responderá avisándole de que la ha recibido correctamente. Si el usuario quiere valorar un producto, entonces irá a ver la lista de productos comprados y escribirá en un formulario la puntuación que desea darle (mediante un entero del 1 (menor puntuación) al 5 (mayor puntuación)) y lo enviará. El AgAsistente generará un grafo con la acción ValorarProducto que tendrá un atributo llamado Valoración con la puntuación recibida por parte del usuario y una relación con el producto (ProductoValorado). Cuando el AgProcesadorOpiniones reciba esta respuesta, actualizará el fichero donde está registrado el producto (Data/Productos o Data/ProductosExternos en función de si es de un producto externo o no), escribirá en Data/Valoraciones la valoración, el usuario y el producto al que pertenece y responderá con un grafo vacío para indicarle de que lo ha recibido y procesado. Solo se permitirá valorar una vez cada producto, aunque éste haya sido comprado diversas veces.

Hemos decidido que la valoración de productos espere unos instantes hasta hacer llegar al agente asistente, para que “simule” que haya llegado el pedido. Esto lo hacemos porque para la demo es mucho más fácil si pasan unos segundos, en caso contrario sería imposible.

Adicionalmente, hemos limitado la valoración hasta 5 puntos, ya que múltiples empresas y tiendas lo implementan de esta forma y nos ha parecido correcto.

3.4. Devolver productos

Esta acción la desencadenará el usuario cuando le indique al AgAsistente que desea devolver un producto comprado. Para ello, el usuario podrá acceder a una pantalla donde se visualizarán las compras que ha realizado con los productos de cada una. Para cada producto que desee devolver, deberá clicar en un botón que iniciará la acción de devolver producto. Primero de todo, indicará la razón en un formulario, que podrá ser por producto defectuoso, equivocado o que no satisfaga las expectativas del cliente. Será entonces cuando el agente AgAsistente generará un grafo con la acción DevolverProducto, relacionada con el producto que desea devolver (ProductoADevolver), con la compra a la que éste pertenece (CompraDevolucion) y tendrá como atributo el motivo de la devolución (MotivoDevolucion).

Cuando le envíe la solicitud de devolución al AgGestorDevoluciones, éste decidirá si la acepta o no. Para llegar a una decisión, mirará el motivo de la devolución. Si es porque no satisface las expectativas del cliente, mirará el tiempo que ha pasado desde que el producto fue entregado. Si este tiempo es mayor a 15 días, la devolución será rechazada y se responderá al AgAsistente con un grafo con la acción DevolverProducto. Si el motivo es por producto equivocado o defectuoso o porque no satisface las expectativas y los días que han pasado son 15 o menos, entonces se procederá a informar al AgAsistente de los datos necesarios para que el usuario devuelva el producto a la tienda. Esta información consistirá en la empresa de mensajería (suponemos que nuestra tienda solo tiene una única empresa de mensajería para simplificar: Correos) y la dirección de envío, que será el nombre de la ciudad en la que se encuentra el centro logístico más cercano, y estará almacenada en el grafo de respuesta de la acción DevolverProducto.

Cuando el AgAsistente reciba este grafo, dará la opción al usuario de que indique si ya ha enviado el producto de vuelta. Cuando el usuario lo indique, el AgAsistente generará un grafo con la acción FinalizarDevolución y la relacionará con el nombre del usuario, el identificador del producto devuelto y la compra a la que pertenecía y lo enviará al AgGestorDevoluciones. Nuestro sistema supondrá que cuando el usuario indique que lo ha enviado, el centro logístico lo recibirá instantáneamente y procederá a actualizar el registro de pedidos finalizados y a devolverle el dinero.

Para eliminar el producto devuelto de la compra, modificará el fichero donde se tienen registradas las compras realizadas y quitará el producto de la compra a la que pertenecía. Para devolverle el dinero al usuario, contactará con el

AgServicioPago para que realice una transferencia al agente asistente devolviéndole el precio del producto y lo registre en Data/RegistroEconomico. El grafo que le envíe al servicio de pago contendrá el nombre del usuario, la tarjeta de crédito que se usó para la compra (se devolverá el dinero a la misma tarjeta de crédito que pagó por el producto), la cuenta de la tienda (que será quién realice la transferencia), el precio del producto (que será el importe de la transferencia) y el identificador de la compra a la que el producto pertenecía.

En caso de que el producto pertenezca a un vendedor externo (distinguido por el prefijo en el identificador de producto), también se le pedirá al AgServicioPago que realice otra transferencia en la que el vendedor externo pague a la tienda el precio del producto que ésta ha pagado al usuario. En este caso, los datos variarán mínimamente, dado que quien hace la transferencia será la tienda y quien la reciba será el vendedor externo. Para saber a qué cuenta debe realizarla, el AgGestorDevoluciones realizará la acción CobrarVendedorExterno donde pedirá al AgVendedorExterno la cuenta bancaria del vendedor externo al que se le debe cobrar el importe. Lo hará enviándole un grafo con la acción CobrarVendedorExterno y el nombre del vendedor externo del cual necesita esta información. El AgVendedorExterno le enviará un grafo como respuesta con la misma acción y un nodo que contenga la cuenta bancaria correspondiente. Tras conocer esta información, en el grafo generado para devolver dinero se modificarán los campos de la cuenta que reciba y de la que paga. Este grafo se enviará al AgServicioPago para que realice esta nueva transferencia y éste escribirá en Data/RegistroEconomico y responderá una vez haya finalizado.

Para esta funcionalidad, hemos tomado ciertas decisiones de diseño que han afectado posteriormente a la implementación. Para empezar, nuestro sistema permite que un usuario compre un mismo producto tantas veces como lo desee. Por ello necesitamos que el AgAsistente no sólo indique el identificador del producto sino que también la compra a la que éste pertenece. Además, hacemos dos suposiciones para facilitar y agilizar la ejecución. Por una parte, suponemos que una vez el usuario indique que ha entregado un producto a la empresa de mensajería que se encargará de llevarlo al centro logístico correspondiente, el centro logístico lo recibe inmediatamente. De esta manera, puede seguir con el proceso de devolución. También suponemos que la tarjeta de crédito a la que se debe ingresar el importe de la devolución es la misma que la que compró dicho producto. Esta decisión, además de facilitar la ejecución, es lo que se hace frecuentemente entre tiendas electrónicas. Por último, al no disponer de los datos bancarios del vendedor externo propietario del producto, hemos visto necesario realizar una request a este agente para así poder obtenerlos y realizar la transferencia.

3.5. Recomendar productos

Cada semana, el sistema, mediante el agente AgProcesadorOpiniones, realizará una búsqueda de productos que considere que pueden interesar a sus usuarios. Esta búsqueda será personalizada basándose en su historial de búsqueda y de compra (se puede consultar en el fichero Data/Historial). Para hacerlo, ejecutará, para cada usuario que tenga historial de búsqueda, una función que devolverá los productos que recomendará en esa semana. La función irá leyendo los productos del fichero mencionado. Primero consultará a qué categoría pertenecen e incrementará en 1 el valor de la posición correspondiente a la categoría que sea de un vector que guardará la cantidad de veces que el usuario ha buscado productos de cada categoría. Con estos datos, el agente elegirá las categorías que tengan más peso en el vector y serán las elegidas para la búsqueda en el catálogo. Para restringir los productos por su precio basándose en el precio que ha interesado más al usuario, calculará la media de los productos del historial y calculará un rango, que variará en cada recomendación, que se sumará y restará al precio medio para encontrar el rango de precios que se usará para escoger los productos recomendados. De los productos que salgan, se descartaran los que haya comprado el usuario.

Tras haber calculado los filtros a aplicar, el AgProcesadorOpiniones leerá los ficheros donde están los productos de la tienda (Data/Productos para los internos de la tienda y Data/ProductosExternos para los externos) y, para cada producto, si cumple las condiciones, lo añadirá a un grafo de la acción RecomendarProductos que contendrá los productos recomendados con la relación ProductoRecomendado. Cuando haya recorrido todo el catálogo le enviará el grafo al AgAsistente. Éste lo guardará para que, cuando el usuario desee ver los productos que el sistema le haya recomendado, se lo pueda mostrar pueda visualizandolo en /hacer_pedido, ya que así podrá proseguir con la compra de los que desee adquirir.

En esta funcionalidad hay muchísimas opciones, nos hemos decantado por una que creemos que es bastante sensata y que realmente permite al usuario visualizar coses que le puedan interesar. Sería más fácil coger todos los productos que ha visto el usuario, quitar los que ha comprado y mostrarselos, pero creemos que nuestro sistema tiene más valor. Con nuestro sistema, podremos ver qué categorías le han gustado más al usuario, y al descartar los que haya comprado, podremos presentar al usuario un conjunto de productos que puedan ser complementarios a su compra, entre otros.

3.6. Añadir productos externos

Cuando un vendedor externo desee añadir un producto al catálogo de la tienda, el AgVendedorExterno rellenará un formulario con la información necesaria (Nombre del vendedor externo, Nombre, Marca, Categoría, Peso y PrecioProducto) y lo enviará. Este generará un grafo con la acción AñadirProductoExterno y, con la relación Añade, unirá el producto (con sus datos) que quiere añadir además de añadir el nombre del vendedor externo como atributo. Cuando el AgVendedorExterno envie el mensaje al AgGestorProductos, éste se encargará de escribir en el fichero Data/ProductosExternos el nuevo producto, inicializando los datos que faltan (Identificador, Categoría, Valoración y CantidadValoraciones y la relación Ofrece con el vendedor externo) y le responderá confirmando que se ha añadido correctamente.

El nombre de la empresa que quiera añadir un producto debe ser uno de los nombres que tenemos registrados. Esto simula las relaciones con empresas que tenga nuestro sistema. Solo puedes vender tus productos en nuestro portal si eres socio nuestro. Creemos que de esta forma le da un toque más real.

4. Tareas nivel de desarrollo avanzado

Para poder ampliar las funcionalidades y acciones que el sistema puede realizar con tal de hacer que sea más complejo y realista, hemos elegido los siguientes dos elementos.

La negociación compleja entre el centro logístico y los agentes de transporte. Esta negociación va más allá de la implementación básica, dado que una vez el centro logístico recibe una propuesta inicial, responde a los servicios de transporte con una contraoferta en la que reduce ligeramente el precio ofrecido. Estos transportistas pueden rechazar o aceptar la contraoferta, proponiendo otra inferior a su oferta inicial. El centro logístico elige la oferta más barata entre las contraofertas que haya recibido. Si todos los transportistas han rechazado la contraoferta, elegirá al transportista que ofreció el precio más bajo inicialmente.

Nuestro sistema también cuenta con una gestión del pago de compras. Esto incluye las transferencias que se deben hacer relacionadas con las compras (de los clientes a la tienda y de la tienda a los vendedores cuando se vende alguno de sus productos) y con las devoluciones (de la tienda a los clientes).

5. Tareas nota extra

Respecto a la nota extra hemos decidido realizar la tercera opción, es decir, hacer otra de las tareas de nivel avanzado. Esta ha sido la implementación de la petición de valoraciones al AgAsistente después de cierto tiempo de que ha realizado la compra de algún producto. Esta tarea, también incluye la implementación de las recomendaciones de productos que el AgProcesadorOpiniones le envía al AgAsistente a partir del historial de compra y de búsqueda, por si está interesado en obtener alguno de estos productos. Este proceso se describe de forma más detallada en el apartado 3.3 y 3.5 de esta documentación.

6. Evaluación de los resultados

Como equipo creemos que hemos desarrollado un sistema completo que da solución al problema planteado. El agente asistente que hemos creado es capaz de lidiar con cada una de las tareas planteadas, desde comprar productos, hasta valorar o devolver otros. Por otra parte, hemos desarrollado el sistema interno encargado de gestionar todas las peticiones que le lleguen del agente asistente.

Para empezar, hemos obtenido una búsqueda de productos que se comporta de manera correcta, capaz de buscar tanto productos externos como propios del sistema, y respetando cada una de las restricciones que se indiquen. Adicionalmente, hemos conseguido un proceso de compra correcto que permite al usuario comprar cualquier producto, indicando la respectiva información. Hemos sido capaces de crear distintos agentes para distribuir todo el proceso y subprocesos que implica la compra. Por una parte hemos diseñado la entidad del Centro Logístico, que es capaz de negociar, de forma compleja y realizando contraofertas, el envío de un lote de productos con los transportistas. Estos son capaces de proponer precios de envío diferentes basándose en el peso, prioridad, y distancia entre el centro logístico y la dirección de entrega.

Otro elemento que hemos desarrollado es el agente del servicio de pago. Este agente, nos permite efectuar cobros y transferencias a otros, con lo cual tenemos un registro económico que controla todas y cada una de las acciones económicas que intervienen en el sistema, desde la compra de un producto, hasta la devolución del importe de este.

La devolución y recomendación de productos también son presentes en nuestro sistema. Esta primera, permite al usuario devolver los productos por diferentes motivos. Para que la devolución se haga efectiva, se deben cumplir ciertos requisitos. En caso de que sea viable, nuestro sistema está preparado para que se pueda efectuar y el usuario pueda recuperar su dinero.

El agente procesador de opiniones, que gestiona tanto devoluciones como recomendaciones, es capaz de basar-se en el historial de búsqueda y compra de un usuario con tal de presentarle productos que sean de su agrado.

Como se puede apreciar, nuestro sistema abarca un gran conjunto de funcionalidades que dan solución al problema planteado, y que hacen que el objetivo de tener un agente asistente para el cliente, se cumpla.

Como trabajo futuro, se podría ampliar la cambiabilidad de las entidades con las que se relaciona el sistema. Es decir, se podría añadir la posibilidad de dejar de ser una empresa externa con un vínculo con nuestro sistema, y a la vez permitir que otras empresas nuevas puedan empezar a vender sus productos en nuestra plataforma. Lo mismo con los transportistas, se podrían cancelar y añadir distintas empresas de transporte para que hubiera más competitividad en ese mercado. Otra mejora para el futuro sería que un usuario además de dar una valoración pudiera dejar un comentario del producto, o que, por ejemplo, el gestor de pago te ofreciera otras modalidades a la hora de pagar un producto.

Para que la evaluación de los resultados sea completa, también creemos que debemos contemplar las limitaciones de nuestra solución, y las implicaciones de estas.
Una de las limitaciones más grandes que hemos tenido ha sido el desconocimiento de las librerías y las herramientas de estas. Durante la implementación lo hemos ido usando cuando lo hemos necesitado pero creemos que si hubiéramos tenido más experiencia con estas herramientas nuestra solución hubiera obtenido más calidad, y alomejor hubiéramos podido agrandar algunas secciones.

En el ámbito del diseño de la solución hemos tenido un par de limitaciones. La primera, el hecho de utilizar herramientas de diseño con las que cuesta mucho encontrar información y ejemplos. Es verdad que hemos podido contar con los vídeos de la asignatura, pero al ser programas que no se utilizan de forma *frecuente,* puede que el diseño haya perdido un poco de calidad por esa falta de referencias a otros usuarios de estas herramientas.

Por otra parte, el hecho de hacer la asignatura en el ámbito online nos ha perjudicado en reuniones de diseño. Nosotros tres, que ya hemos hecho más de un proyecto juntos, hemos notado como las sesiones de diseño han sido mucho más grávidas, por el hecho de tener que estar separados. Creemos que presencialmente es mucho más fácil y divertido de discutir todo tipo de dudas y proposiciones.

Por último, el tiempo es un factor que nos limita en términos de calidad y complejidad de la solución que proponemos. Está claro que con el doble o triple de tiempo podríamos desarrollar una solución que diese respuesta a más secciones del problema planteado, pero el curso tiene unas fechas programadas y tenemos que respetarlas.

7. Comentarios finales

Primeramente querríamos describir nuestra planificación del trabajo. En el ámbito de diseño del sistema y el desarrollo de la ontología, hemos realizado llamadas online todos los integrantes del grupo. En estas llamadas hemos discutido y tomado decisiones, a la vez que hemos buscado información en la documentación de la asignatura, y creado los ficheros necesarios para el desarrollo de la solución.

La implementación ha seguido una línea similar, pero con algunos cambios. Hemos mantenido el trabajo en línea. Esto implica que cada vez que hemos acordado una sesion de desarrollo de la implementación, nos hemos llamado y hemos estado trabajando juntos. Obviamente, esto no implica que hemos estado implementando los mismos ficheros. Al principio de cada reunión nos hemos distribuido pequeñas tareas para poder trabajar de forma independiente. Con este sistema, cualquier duda que haya podido tener cada integrante, la ha podido comentar con los otros dos justo en ese momento. Hemos adoptado esta forma de trabajar porque no tenemos experiencia en este tipo de proyectos y nos han surgido dudas en todo momento.

Como respuesta a qué partes de la implementación ha hecho cada persona, no lo podemos especificar pues todo integrante ha desarrollado porciones de cada parte, a veces de forma principal, a veces arreglando errores y a veces testeando la correcta ejecución de los agentes. Tal y como he dicho anteriormente, hemos querido ir paso a paso de forma conjunta para asegurar que obtenemos una solución buena, sólida y entendible.

Como comentario final, personalmente estamos muy contentos de cómo ha quedado el proyecto. Hemos sufrido un poco al principio con la implementación, pero gracias a la consultas y a ir aprendiendo paso a paso todas las partes de la asignatura hemos podido realizar un trabajo que consideramos de calidad.