



Ennvas

Sistema de tienda online multiagente basado en inteligencia artificial

Proyecto de prácticas
de Ingeniería del Software II

Desarrollado por:

Cbers

Raquel Pérez González de Ossuna
Olga Posada Iglesias
Nicolás Pardina Popp
Juan Francisco Carrión Molina
Melany Daniela Chicaiza Quezada

Profesor:
Eugenio Pablo Concepción Cuevas

Grupo B
Convocatoria de junio
Curso 2019/2020
Grado en Ingeniería Informática
Facultad de Informática

Índice

1. Introducción	2
1.1. Contexto	2
1.2. Elevator pitch	2
2. Descripción funcional	2
2.1. Capacidades planeadas	2
2.2. Mapa de historias de usuario	3
2.3. Índice de historias de usuario	4
3. Arquitectura de la aplicación	6
3.1. Diagrama de contexto	6
3.2. Diagrama de contenedor	7
3.3. Diagrama de componentes	7
4. Diseño detallado	7
4.1. Secuencia de una búsqueda normal	7
4.2. Comunicación mediante REST	7
4.3. Componente recomendador <code>recommender</code>	7
4.4. Componente de gestión de agentes <code>agent manager</code>	8
4.5. Componente orquestador <code>orchestrator</code>	8
4.6. Componente de interfaz <code>front end</code>	8
5. Instalación y prueba	9
5.1. Instalación	9
5.2. Despliegue simple	9
5.3. Pruebas	9
6. Conclusiones	10
7. Bibliografía	11

1. Introducción

Este documento es una memoria de la práctica de la asignatura de Ingeniería del Software 2. Recoge todo el desarrollo de la misma, realizado a partir de la especificación del proyecto y siguiendo las guías dadas.

La práctica ha consistido en el desarrollo de una aplicación ejecutable cuyo diseño hace uso de los patrones de diseño y estilos arquitectónicos estudiados en la asignatura.

Como metodología de desarrollo, se ha utilizado Scrum con *sprints* de 2 semanas de duración, con un incremento de valor del producto en desarrollo al finalizar cada uno.

Esta documentación recoge en líneas generales información relativa a la funcionalidad entregada, al diseño arquitectónico de la solución y al diseño detallado de los componentes más relevantes (incluyendo patrones de diseño), y un cierre incluyendo consideraciones finales.

El código fuente, junto a los medios de apoyo para la presentación y defensa del proyecto, se entrega a parte para su revisión y evaluación.

1.1. Contexto

En este proyecto se ha desarrollado un sistema de tienda online multiagente basado en inteligencia artificial (*AI multiagent shopping system*), cuyo nombre hemos decidido que sea **Ennvas**.

El sistema permite al usuario introducir los detalles de un producto en particular. A continuación, se guardan todos estos detalles y el sistema busca varios artículos que coincidan con la búsqueda.

Tras realizar la búsqueda, el sistema devuelve una lista con los artículos más adecuados para las necesidades del usuario. También se sugieren otros artículos que podrían gustar al usuario, es decir, que tienen alta probabilidad de ser comprados por este en base a sus requisitos previos.

El sistema atiende a varios usuarios a la vez y proporciona resultados precisos.

Los módulos y el resto de contenido de la especificación se desarrollan más adelante en este documento, como parte de la práctica.

1.2. Elevator pitch

En este *elevator pitch* analizamos, en base a un esquema, cómo presentaríamos el proyecto a un potencial cliente de forma concisa y en un breve espacio de tiempo, destacando qué hace el sistema en comparación con otros de competencia e identificando las principales ventajas y beneficiarios.

*Para cualquier tienda o comercio que quiera mejorar y agilizar la experiencia de compra de sus clientes el **sistema de tienda online multiagente basado en inteligencia artificial** (AI multiagent shopping system) es un sistema de tienda en línea que integra inteligencia artificial para sugerir productos a los clientes basados en sus intereses registrados (también ahorro en personal). A diferencia de cualquier otro sistema de compra convencional, este sistema recoge los detalles que interesan al usuario y le sugiere una lista con los productos más convenientes según esos detalles.*

2. Descripción funcional

Para organizar el desarrollo del sistema y comprender de forma mejor y más clara lo que se pide y lo que no, realizamos un análisis de capacidades funcionales. En un primer lugar, propusimos todas las funciones que imaginamos alrededor del sistema y las distribuimos en tres listas: *to do*, *not to do* y *unresolved*. Tras deliberación y aclaración con el cliente, en este caso, con el profesor y la hoja de especificación, la última lista quedó vacía y tuvimos claro qué desarrollar y qué no. Estas capacidades se especifican en el próximo subapartado.

A la hora de llevar el proyecto al código, se ha realizado la implementación de algunas partes de la funcionalidad. Este desarrollo se ha llevado a cabo siguiendo el principio de que cada avance añadiera funcionalidad al producto y centrándonos en el *núcleo* del proyecto, es decir, el algoritmo de búsqueda basado en inteligencia artificial. Esto se detalla en siguientes apartados.

2.1. Capacidades planeadas

2.a. Capacidades funcionales clasificadas para sí desarrollar (*to do*)

En este proyecto se planificaron para desarrollo las siguientes funcionalidades. En el momento de en-

trega de esta memoria, son funcionales las marcadas con la anotación **Func.**.

- Gestion de registro y sesión de usuario en el sistema.
- *Carrito de compra* para guardar artículos que el usuario pretende comprar.
- Un sistema de pago que soporte tarjetas bancarias, PayPal y criptomonedas.
- **Func.** Categorización y detallado de los productos.
- **Func.** Algoritmo de búsqueda basado en inteligencia artificial que reciba los detalles introducidos por el usuario y devuelva, de la base de datos, una lista de productos adecuados.
- **Func.** Base de datos en la que se almacenen los productos de forma adecuada para que el algoritmo pueda trabajar con los datos y ofrecer los resultados adecuados. Para este fin y otros, por ejemplo, estadísticos, también almacenará datos de compras, ubicaciones, etc.
- Un *web crawler* que rastree otras páginas que tengan productos similares para poder modificar los precios según convenga para mantenerse competitivos.
- **Func.** La posibilidad de soportar sin problemas una cantidad razonable de usuarios simultáneamente.
- Espacio de usuario con historial de pedidos, direcciones, facturación, etc.
- Espacios para anuncios que se puedan alquilar o usar para artículos propios de la tienda.
- **Func.** Interfaz web para escritorio.
- Interfaces web para teléfonos móviles y tabletas.
- Capacidad de trabajo con diferentes idiomas (internacionalización).
- **Func.** Valoración de productos.

2.b. Capacidades funcionales clasificadas para no desarrollar (*not to do*)

En este proyecto no se planificaron para desarrollo las siguientes funcionalidades.

- Reparto, seguimiento y gestión de paquetes.
- Atención al cliente y soporte.
- Decisión de compra (será el usuario final quien la tome).

- Lista de deseos.
- Predicción de la preferencia del usuario si los detalles especificados no son claros.
- Otros métodos de pago (contra reembolso, Bizum, Google Pay, etc.).
- Total corrección de resultados, es decir, garantía de que el usuario encuentre lo que buscaba.
- Otras interfaces como línea de comandos, de programación de aplicaciones (API), interacción con otros sistemas o aplicaciones nativas para sistemas operativos.

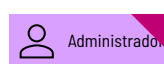
2.2. Mapa de historias de usuario

(Ver gráfico en el Anexo 1. Mapa de historias de usuario.)

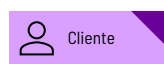
Un mapa de historias de usuario o *user stories map* es una ayuda en el desarrollo del sistema que nos permite definir y concretar el trabajo a realizar para conseguir la mejor experiencia de usuario final.

Nos permitió identificar todo lo que se debía desarrollar en el sistema y distinguirlo de los elementos en los que no debíamos trabajar. Es decir, el mapa de historias de usuario nos sirvió para finalizar y refinar el trabajo de la sección anterior en las listas *to do* y *not to do*.

En nuestro caso, construimos un mapa de historias de usuario con dos tipos de usuario:



Una persona física interna que gestiona y administra los productos y el contenido del sistema.



Una persona física externa que quiere adquirir un producto a través del sistema.

Presentamos el mapa en dos páginas debido al tamaño. Se muestran con menor opacidad las tarjetas cuya funcionalidad no se ha implementado en el momento de entregar esta memoria. Para dar más claridad, tenemos tres viajes de usuario completos:

- **Ciclo de pedido** En el que interactúan cliente (desde un frontend) y administrador (desde un backend) desde el momento en que el cliente hace la búsqueda hasta que recibe sus productos pasando por las opciones de compra, e incluyendo posibles devoluciones.

- **Gestión del sistema** El administrador podrá gestionar el sistema, productos, contenidos, usuarios, etc. en cualquier momento.
- **Gestión del perfil** El cliente podrá gestionar su perfil, su cuenta, sus pedidos, etc. en cualquier momento.

2.3. Índice de historias de usuario

Para describir cada una de las historias de usuario, hemos optado por *alinear* las tarjetas y colocar en frases las descripciones con el formato como **usuario a** quiero **b** para **c**. El título de cada elemento de la lista indica con la anotación **Func.** si la funcionalidad se ha implementado.

Las secciones 2.a., 2.b. y 2.c. hacen referencia a cada viaje completo de usuario. Dentro de cada viaje completo, se describen las historias de cada viaje parcial o más concreto. Para ayudar visualmente al seguimiento con el mapa de usuario, a la izquierda de los títulos colocamos una etiqueta con el usuario que corresponde.

Cada tarjeta de usuario lleva, además, la versión en la que sería implementada.

2.a. Ciclo de pedido

Cliente Registro de usuario

- **Introducir los datos del usuario** (v. 1): *Como nuevo usuario quiero introducir mis datos personales para identificarme y tener acceso al sistema.*
- **Enviar formulario y controlar errores** (v. 1): *Como nuevo usuario quiero que se controlen posibles errores al introducir mis datos para aportar fiabilidad a mi registro.*
- **Verificar dirección de correo electrónico** (v. 1): *Como nuevo usuario quiero recibir un correo para confirmar que el email que he introducido es correcto.*
- **Foto de perfil** (v. 2): *Como nuevo usuario quiero añadir una foto de perfil para facilitar mi identificación.*

Cliente Introducir detalles de búsqueda

- **Func. Proporcionar datos de búsqueda** (v. 1): *Como usuario quiero especificar mis*

necesidades para poder obtener como resultado una lista de productos correspondientes a estas.

- **Filtrar por categoría** (v. 1): *Como usuario quiero poner filtros por categoría de producto para encontrar más fácilmente el producto que deseo.*
- **Func. Filtrar por detalles específicos (color, tamaño, marca...)** (v. 2): *Como usuario quiero filtrar por detalles más concretos (color, tamaño, marca...) para acelerar la búsqueda del producto que deseo.*
- **Ofrecer resultados mientras se introducen los detalles** (v. 3): *Como usuario quiero obtener resultados una vez he introducido filtros para obtener mientras tanto posibles productos de compra.*

Cliente Resultados de búsqueda y recomendaciones

- **Func. Visualizar resultados** (v. 1): *Como usuario quiero visualizar los resultados de la búsqueda para poder elegir el que más se ajuste a mis necesidades.*
- **Ordenar resultados** (v. 1): *Como usuario quiero ordenar los resultados de mi búsqueda por popularidad, recomendaciones y precio ascendente o descendente para obtener un resultado más acorde con mi punto de vista.*
- **Sugerencias de búsqueda "quizás quisiste decir..."** (v. 2): *Como usuario quiero obtener correcciones a mis campos de filtrado ("quizás quisiste decir...") para corregir errores de búsqueda y obtener otros resultados.*
- **Paginar resultados** (v. 2): *Como usuario quiero que los resultados se dividan en páginas para mejor organización y visualización.*
- **Vista previa rápida de producto** (v. 3): *Como usuario quiero obtener una vista rápida previa del producto para hacerme una pequeña idea de las especificaciones del producto.*
- **Visualización de anuncios** (v. 3): *Como usuario quiero visualizar anuncios de páginas externas para tener recomendaciones de distintas páginas web.*

Cliente Página del producto

- **Func. Detalles del producto (nombre, descripción, disponibilidad..)** (v. 1): *Como*

usuario quiero ver toda la información del producto antes de comprarlo para asegurar que es el producto que finalmente deseo.

- **Opciones de compra (cantidad, color...)** (v. 1): Como usuario quiero ver todas las opciones posibles de compra para ver los productos que cumplan con las características que deseo.
- **Valoraciones del producto** (v. 2): Como usuario quiero leer las reseñas de otros usuarios para, con su ayuda, decidir si verdaderamente es un buen producto.
- **Galería de imágenes** (v. 2): Como usuario quiero acceder a una galería de fotos del producto para comprobar de forma visual que se adapta a mis necesidades.
- **Opciones de compra avanzadas (personalización, proveedores...)** (v. 2): Como usuario quiero tener un apartado de opciones de compra avanzada para especificar determinados campos de compra (personalización, proveedores, tipos de envío...).
- **Botones de compartir y redes sociales** (v. 3): Como usuario quiero compartir el producto con otras personas mediante redes sociales para difundir la existencia del producto u obtener opiniones de este.
- **Botón añadir al carrito de compra** (v. 3): Como usuario quiero poder añadir el producto al carrito de la compra para seguir realizando búsquedas de otros productos.
- **Botón comprar ahora** (v. 3): Como usuario quiero tener la opción de comprar el producto directamente (sin añadirlo al carrito) para agilizar la compra en caso de querer un único producto.

Cliente

Compra

- **Visualización del carrito de compra** (v. 1): Como usuario quiero tener una visualización del carrito de la compra antes de proceder al pago para verificar que son los productos y cantidades que deseo.
- **Introducir detalles de envío y facturación** (v. 1): Como usuario quiero introducir detalles de envío y facturación para proceder a la realización de la compra.
- **Seleccionar método de pago** (v. 1): Como usuario quiero seleccionar un método de pago específico para pagar con un método de pago del que yo disponga.

- **Realizar pago según método seleccionado** (v. 1): Como usuario quiero realizar el pago (según el método seleccionado) para finalmente adquirir el producto.
- **Recepción de email de confirmación** (v. 1): Como usuario quiero recibir un email con una confirmación de mi compra para tener los detalles de mi compra de forma segura.
- **Vales y descuentos** (v. 2): Como usuario quiero introducir un código de vale o descuento para reducir el precio total de mi compra.

Administrador

Gestión de pedido

- **Visualización del pedido y procesamiento** (v. 1): Como administrador quiero ver los pedidos que se han hecho en mi tienda para gestionarlos y proceder a enviarlos.
- **Respuesta al cliente con el número de seguimiento** (v. 1): Como administrador quiero responder a un cliente con un número de seguimiento para que pueda saber cómo va su pedido.

Cliente

Posventa

- **Solicitud de devolución** (v. 1): Como usuario quiero devolver un producto para recuperar mi dinero.
- **Garantía y soporte técnico (SAT)** (v. 1): Como usuario quiero tener una garantía y soporte técnico del producto para asegurar el buen estado y funcionamiento de este de forma temporal (acorde con la duración de garantía).
- **Valoración de productos** (v. 2): Como usuario quiero añadir opiniones de los productos que he adquirido para ayudar a otros usuarios en su decisión de compra.

Administrador

Gestión de retorno

- **Aprobación del retorno por devolución o garantía** (v. 1): Como administrador quiero poder aprobar (o denegar) una solicitud de devolución de un producto para que el usuario haga uso de sus derechos ya sea de devolución o de garantía.

Cliente

Retorno

- **Recepción de email de confirmación e información e retorno (RMA)** (v. 1): *Como usuario quiero recibir un email que confirme que he solicitado la devolución de un producto para saber que dicha solicitud ha sido enviada con éxito.*

Administrador

Finalización del retorno

- **Confirmación de llegada y realización del reembolso** (v. 1): *Como administrador quiero tener la certeza de que un producto haya sido devuelto para proceder al reembolso.*

2.b. Gestión del sistema

Administrador

Gestión del sistema

- **Gestionar productos y contenidos** (v. 1): *Como administrador de la tienda quiero gestionar mis productos y sus contenidos de tal forma que se mantengan actualizados para los distintos usuarios.*
- **Gestionar usuarios** (v. 1): *Como administrador quiero ver los usuarios que accedan a la tienda y compren en ella para poder utilizar esa información en futuras estadísticas.*
- **Visualizar estadísticas** (v. 1): *Como administrador quiero tener estadísticas de mis productos más vendidos para tener información extra para satisfacer al cliente.*
- **Gestionar y moderar valoraciones** (v. 2): *Como administrador quiero gestionar las valoraciones hechas de mis productos para mantener el buen clima y eliminar comentarios ofensivos o inapropiados.*
- **Gestionar ofertas y promociones** (v. 2): *Como administrador quiero gestionar el tipo de ofertas y promociones que se aplican sobre los artículos para tener el control sobre el precio dependiendo de si las aplico o no.*
- **Gestionar anuncios** (v. 3): *Como administrador quiero gestionar los anuncios que aparecerán en el sistema para poner unos acordes con las necesidades de los clientes.*
- **Integración con redes sociales** (v. 3): *Como administrador quiero gestionar qué redes sociales están disponibles para que los usuarios puedan posteriormente compartir en ellas los distintos artículos.*

- **Gestionar idiomas** (v. 3): *Como administrador quiero gestionar qué idiomas están disponibles en el sistema multiagente para que este tenga llegue a un público mayor.*

2.c. Gestión del perfil

Cliente

Gestión del perfil

- **Gestionar datos personales y tratamiento** (v. 1): *Como usuario quiero modificar los datos que inicialmente introduje en mi registro para mantenerlos actualizados.*
- **Gestionar pedidos** (v. 1): *Como usuario quiero poder gestionar mis pedidos para poder cancelarlo, ver actualizaciones...*
- **Gestionar notificaciones** (v. 1): *Como usuario quiero gestionar mis notificaciones para ocultar (o no) algunas.*
- **Foto de perfil** (v. 2): *Como usuario quiero cambiar mi foto de perfil para mantenerla actualizada.*

3. Arquitectura de la aplicación

Para documentar la arquitectura de la aplicación, utilizamos el modelo C4. Explicamos la estructura de conjunto del sistema y, progresivamente, desarrollamos cada parte.

Los colores de los diferentes componentes del diagrama indican si ya existían (gris), si se han construido (azul claro) o si son usuarios (azul oscuro).

3.1. Diagrama de contexto

(Ver gráfico en el Anexo 2. Modelo C4 - Diagrama nivel 1 - Contexto del sistema.)

El primer nivel del modelo C4 consiste en un diagrama de contexto del sistema, que muestra cómo encaja la aplicación en un contexto amplio, incluyendo usuarios y otros sistemas software con los que interactúa.

Como podemos observar en el gráfico, Ennvas se presenta como un sistema software de arquitectura cliente-servidor. Recibe datos de tiendas webs –otros sistemas software– y con el que los usuarios finales puede interactuar mediante peticiones de búsqueda.

3.2. Diagrama de contenedor

(Ver gráfico en el Anexo 3. Modelo C4 - Diagrama nivel 2 - Contenedor del sistema.)

El segundo nivel del modelo C4 consiste en un diagrama de contenedor del sistema, que *hace zoom* sobre el sistema software desarrollado y muestra los contenedores (componentes, bases de datos, servicios, etc.) que lo componen. Aquí también mostramos las decisiones tecnológicas.

En nuestro caso, el sistema principal se divide en una interfaz de comunicación con el usuario final, una base de datos para almacenar la información sobre los productos y un contenedor principal que agrupa toda la lógica del sistema y se comunica con el resto de participantes.

3.3. Diagrama de componentes

(Ver gráfico en el Anexo 4. Modelo C4 - Diagrama nivel 3 - Componentes del sistema.)

El tercer nivel del modelo C4 consiste en un diagrama de componentes, que expande el contenedor principal del sistema que se ha desarrollado para mostrar los componentes que contiene.

En nuestro caso, la arquitectura es de microservicios y se distribuye en cuatro componentes: **front end**, **orchestrator**, **agent manager** y **recommender**. En la implementación, cada componente se presenta como una aplicación de Spring Boot individual.

4. Diseño detallado

En este apartado, que se constituye como cuarto nivel del modelo C4, ampliamos cada componente para mostrar cómo se implementa.

A excepción del diagrama de secuencia del primer subapartado, todos los diagramas utilizados para describir los componentes son diagramas de clases UML que se centran únicamente en documentar los diseños de especial relevancia en el proyecto.

4.1. Secuencia de una búsqueda normal

(Ver gráfico en el Anexo 5. Diagrama de secuencia de una búsqueda normal.)

En primer lugar y antes de analizar cada componente, presentamos la secuencia de una búsqueda normal, es decir, la activación de cada componente cuando un usuario realiza una petición de búsqueda.

Para explicar esto, utilizamos un diagrama de secuencia que muestra la interacción secuencial entre componentes y su activación.

4.2. Comunicación mediante REST

Como mencionábamos anteriormente, la arquitectura fundamental del sistema se presenta distribuida en cuatro aplicaciones de Spring Boot individuales, una para cada componente.

Estos componentes se comunican haciendo uso de una API REST a través de HTTP. Por ello, cada componente tiene su controlador REST (patrón *front controller*) y sus correspondientes *endpoints* para acceder a cada servicio.

Estos endpoints se documentan en la [sección de pruebas](#) de esta memoria.

4.3. Componente recomendador recommender

(Ver gráfico en el Anexo 6. Diagrama UML de clases del componente recomendador.)

El componente recomendador es el que lleva a cabo la lógica principal del sistema, que es ejecutar y servir el algoritmo de recomendación basado en inteligencia artificial.

Para ello, recibe una consulta (*Query*) y unos datos de productos por parte del componente orquestador, y devuelve una lista de productos como resultado. La consulta proviene del usuario final y los datos de productos del componente de gestión de agentes.

Para la recomendación, utilizamos un algoritmo basado en utilidad.

Cuando se reciben la consulta y los datos, se instancia la clase `MainAlgorithm` con el método `processQuery()`, que, para cada producto, utiliza el método `UtilityFunction.calculate()`. Este método calcula un valor entero al que llamamos utilidad y que representa una *puntuación* del producto en cuestión para la consulta proporcionada. La utilidad se calcula de la siguiente forma:

- Si el producto no cumple los requisitos de filtros de la consulta (todos los campos excepto

phrase), se determina que su utilidad es -1 .

- Por cada palabra que el usuario haya introducido en el campo de texto de la búsqueda (campo `phrase` de la consulta) que coincida con otra del nombre, descripción, marca o tipo del producto, se incrementa su utilidad.

Cuando se ha aplicado la función de utilidad a toda la lista de productos, todos aquellos que tengan al menos una utilidad mínima se introducen en una nueva lista de productos de un tamaño máximo. Estas dos propiedades, la utilidad mínima y el tamaño máximo de la lista, se especifican como argumentos de arranque de la aplicación.

4.4. Componente de gestión de agentes `agent manager`

(Ver gráfico en el Anexo 7. Diagrama UML de clases del componente de gestión de agentes.)

La funcionalidad principal del `agent manager` es la gestión de los agentes. En el momento de entregar esta memoria, no se ha llevado a cabo la implementación del sistema multiagente, por lo que se ha utilizado un *mockup* para la información que proveían los agentes.

Así, al arrancar la aplicación, es necesario proporcionar como único argumento una ruta a un fichero JSON con los datos de demostración. Estos datos son procesados por `DemoGeneratorBean`, que se encarga de analizar el fichero e insertar los datos en la base de datos a través del repositorio.

Los datos de productos se almacenan de forma persistente, siendo, por el momento, el único componente que hace uso de persistencia de datos, a través de la Java Persistence API. Destacamos aquí el uso de los patrones `repository`, `data access object` (DAO) y `data transfer object` (DTO). En nuestro caso, hemos optado por utilizar una base de datos de tipo MariaDB (conector MySQL para Spring Data JPA).

Cuando el `agent manager` recibe una petición del componente orquestador, le devuelve un listado completo con toda la información que tiene almacenada.

4.5. Componente orquestador `orchestrator`

(Ver gráfico en el Anexo 8. Diagrama UML de clases del componente orquestador.)

El componente orquestado se encarga de coordinar el resto de componentes. Cuando recibe una petición del `front end`, pide los datos de los productos al `agent manager`, hace una petición al `recommender` y devuelve los resultados al usuario.

La aplicación recibe como argumentos de arranque dos URL correspondientes al `agent manager` y al `recommender`. Estas le permiten invocarlos cuando es necesario, a través de sus correspondientes API REST.

4.6. Componente de interfaz `front end`

(Ver gráfico en el Anexo 9. Diagrama UML de clases del componente de interfaz.)

La funcionalidad principal del `front end` es la comunicación con el usuario final mediante una aplicación web. En este caso el componente se encarga de servir la aplicación como de presentar la API REST de la que la interfaz hará uso.

El `front end` presenta la API REST para hacer como puente entre la interfaz web y el componente orquestador ya que, por políticas de seguridad, solo se pueden realizar llamadas AJAX en JavaScript a recursos del mismo sitio (*Same-origin policy*).

La aplicación web servida por este componente se compone básicamente de un documento HTML que utiliza el framework Bootstrap para mostrar un formulario. El formulario se compone de un campo principal de texto y distintos campos de filtro. Todo el formulario en conjunto compone la información necesaria para una consulta (Query).

Cuando el usuario envía el formulario de la interfaz web, se realiza una petición a través de AJAX al mismo `front end`. Este convierte la petición a un objeto `Query` y lo envía al componente orquestador.

La aplicación recibe como argumento único de arranque la URL del `orquestador`, que es la que se utiliza para invocar al mismo a través de su API REST cuando se recibe una consulta.

Cuando se recibe una respuesta, la lógica de JavaScript y jQuery de la interfaz web la procesa y genera los productos para mostrarlos visualmente.

Otro patrón utilizado en este componente es el MVC, o modelo-vista-controlador, que es utilizado por la naturaleza de arquitectura cliente-servidor del mismo.

5. Instalación y prueba

5.1. Instalación

Ennvas se presenta como una aplicación que sigue una arquitectura distribuida, por lo tanto, los cuatro componentes pueden estar corriendo en una única máquina o en varias máquinas distintas, siempre que exista una conexión de red entre ellas, ya que usan API REST a través de HTTP para comunicarse.

El primer paso será inicializar la base de datos en la máquina en la que vaya a correr el componente de gestión de agentes. Las credenciales de la misma pueden modificarse en:

```
.\agentmanager\src\main\resources\ \
application.properties
```

Después, hay que compilar el código fuente de cada componente. Para ello, hay que ejecutar el siguiente comando desde el directorio raíz de cada uno: Download

Source

PDF

Actions

Copy Project Word Count

Sync

Dropbox

Git

GitHub

Settings Compiler TeX Live version Main document Spell check Auto-complete Auto-close Brackets Code check Editor theme Overall theme Key-bindings Font Size Font Family Line Height PDF Viewer Help

Show Hotkeys Documentation Contact Us

is2-practica-junio-2020-cbers 4

Editor mode.

```
$ mvn clean package
```

5.2. Despliegue simple

1. Inicializar la base de datos MySQL.

2. Inicializar el componente **agent manager**. Recibe un argumento, que es la ruta al fichero JSON con los datos de prueba.

```
$ java -jar agent-manager.jar \
demo-products.json
```

3. Inicializar el componente **recommender**. Recibe dos argumentos, los cuales son un número mínimo de utilidad para que el producto se muestre en el resultado final y la cantidad de productos que se mostrarán.

```
$ java -jar recommender.jar 1 6
```

4. Inicializar el componente **orchestrator**. Recibe dos argumentos, los cuales son las URL del **agent manager** y del **recommender**.

```
$ java -jar orchestrator.jar \
http://localhost:60002 \
http://localhost:60004
```

5. Inicializar el componente **front end**. Recibe un argumento, que es la URL del **orchestrator**.

```
$ java front-end.jar \
http://localhost:60003
```

6. Visitar la interfaz web del front end, con URL <http://localhost:60005> (por defecto).

5.3. Pruebas

En la siguiente sección, enseñamos como mandar peticiones de prueba a los componentes mediante un cliente REST, por ejemplo, el programa *Advanced Rest Client*.

Componente de gestión de agentes

- Endpoint REST:
[/ennvas/agm/rest/retrieve](#)
- Método: GET

Componente de recomendación

- Endpoint REST:
[/ennvas/rcm/rest/process](#)
- Método: POST
- Content-type: `application/json`
- Body: ver anexo TODO

Componente orquestador

- Endpoint REST:
[/ennvas/orch/rest/search](#)
- Método: POST
- Content-type: `application/json`
- Body: ver anexo TODO

Componente de interfaz

- Endpoint REST:
[/ennvas/front/rest/search](#)
- Método: POST
- Content-type: `application/json`
- Body: ver anexo TODO

6. Conclusiones

El desarrollo de la asignatura se ha visto bastante alterado debido a las circunstancias extraordinarias en la que nos encontramos. Esta situación ha

sido causada por la pandemia del virus SARS-CoV-2, que además, como ya sabemos todos, trajo problemas colaterales como fue el confinamiento. Este último nos ha impedido hacer reuniones presenciales con los miembros del grupo, que habrían facilitado mucho la realización del proyecto, pues algunos miembros se encontraban con condiciones técnicas deplorables. Sin embargo, si hubiéramos tenido acceso a los recursos informáticos de la facultad, no se habrían visto limitadas la capacidad de aportación al proyecto de ningún miembro del grupo. Por culpa de todos estos inconvenientes, nos ha sido imposible abordar todos los aspectos planteados en el alcance que fue previsto inicialmente.

En futuras versiones se aumentaría el alcance de la aplicación, implementando nuevas funcionalidades como la indexación web en el Agent Manager. La posibilidad de registrar usuarios, y que estos puedan realizar compras en la aplicación, que haría de intermediario con el vendedor del producto. Esta funcionalidad permite guardar información de los usuarios en el sistema, lo que abre la posibilidad de mejorar el recomendador existente mediante una hibridación. Por ejemplo un híbrido de cascada del recomendador basado en la utilidad con otro basado en el contenido. Esto no conllevaría cambios significativos en la implementación actual, ya que este segundo método refina el filtrado más burdo del primero.

Nos ha parecido muy interesante trabajar con tecnologías actuales como Spring Boot que se usan en el mundo laboral, sobre todo en una asignatura de segundo curso de la carrera. Así como aprender a trabajar y organizarnos en un equipo más grande de lo que estamos habituados, y construir un proyecto desde cero.

« Sólo podemos ver poco del futuro, pero lo suficiente para darnos cuenta de que hay mucho que hacer. »

Alan Turing

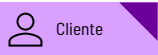
7. Bibliografía

1. Burke, Robin. (2002). *Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction*. 12. 10.1023/A:1021240730564
2. Cervigón, Carlos. *Patrones de diseño I*. Material no publicado
3. Cervigón, Carlos. *Patrones de diseño II*. Material no publicado
4. Concepción Cuevas, Eugenio Pablo. *Introducción a la Arquitectura del Software*. Material no publicado
5. Concepción Cuevas, Eugenio Pablo. *Guía para la documentación de la práctica final*. Material no publicado
6. Concepción Cuevas, Eugenio Pablo. *Introducción a la arquitectura en capas*. Material no publicado
7. Concepción Cuevas, Eugenio Pablo. *Introducción a REST*. Material no publicado
8. Concepción Cuevas, Eugenio Pablo. *Introducción a Spring Framework*. Material no publicado
9. Concepción Cuevas, Eugenio Pablo. *Introducción al desarrollo de aplicaciones con Spring Boot*. Material no publicado
10. Concepción Cuevas, Eugenio Pablo. *Las aventuras de una petición REST en Spring*. Material no publicado
11. *Curso SpringBoot—7 Spring Data (JPA) + Spring MVC*. (2019). <https://www.youtube.com/watch?v=C3Xwu7wuYAO>
12. *How to read JSON data in Spring Boot and write to a database*. (2019). <https://www.youtube.com/watch?v=rGdKmF2UzSc>
13. Manero, Borja. *Lenguaje Unificado de Modelado UML (curso anterior)*. Material no publicado
14. The Project Lombok Authors. (2009, 2020). *@Data*. Project Lombok. <https://projectlombok.org/features/Data>
15. VMware. (2020). *Accessing Data with JPA*. <https://spring.io/guides/gs/accessing-data-jpa/>
16. VMware. (2020). *Accessing data with MySQL*. Spring. <https://spring.io/guides/gs/accessing-data-mysql/>
17. VMware. (2020). *Consuming a RESTful Web Service*. VMware. <https://spring.io/guides/gs/consuming-rest/>
18. VMware. (2020). *Consuming a RESTful Web Service*. VMware. <https://spring.io/guides/gs/consuming-rest/>
19. VMware. (2020). *Consuming a RESTful Web Service with jQuery*. VMware. <https://spring.io/guides/gs/consuming-rest-jquery/>
20. VMware. (2020). *Serving Web Content with Spring MVC*. Spring. Recuperado 17 de mayo de 2020, de <https://spring.io/guides/gs/serving-web-content/>
21. VMware. (2020). *Spring Quickstart Guide*. VMware. Recuperado 4 de abril de 2020, de <https://spring.io/quickstart>

Aplicaciones utilizadas para el desarrollo

- **Visual Studio Code** para el desarrollo del código en Java, gestión de dependencias con Maven y gestión de proyecto Spring Boot, además de trabajo remoto en directo mediante la extensión Live Share.
- **Wamp Server** para iniciar una instancia de servidor MariaDB (MySQL).
- **Advanced Rest Cliente (ARC)** para las pruebas individuales de los distintos componentes a través de sus API REST.
- **GitHub (git)** para la gestión de versiones y colaboración.
- **Inkscape** para trabajo y diseño de gráficos (mapa de historias de usuario, diagramas UML, diagramas del modelo C4, etc.).
- **StarUML** para generar inicialmente los diagramas UML, aunque luego fueron retocados con Inkscape.

- **Google Meet** para las reuniones de equipo, puesta en común de avances y toma de decisiones.
- **Google Drive** para el amacenamiento y com-
partición de recursos gráficos y metadatos del proyecto.
- **Overleaf (LaTeX)** para la composición de este documento de memoria.

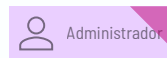


Viaje de usuario (user journey) parcial

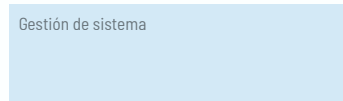
Registro de usuario	Introducir detalles de búsqueda	Resultados de búsqueda y recomendaciones	Página del producto	Compra	Gestión de pedido	Posventa	Gestión de retorno	Retorno	Finalización del retorno
---------------------	---------------------------------	--	---------------------	--------	-------------------	----------	--------------------	---------	--------------------------

Versión 1	Introducir datos del usuario	Proporcionar datos de búsqueda	Visualizar resultados	Detalles del producto (nombre, descripción, disponibilidad...)	Opciones de compra (cantidad, color...)	Visualización del carrito de compra	Introducir detalles de envío y facturación	Visualización del pedido y procesamiento	Devolución	Aprobación del retorno por devolución o garantía	Recepción de email de confirmación e información de retorno (RMA)	Confirmación de llegada y realización del reembolso
	Enviar formulario y controlar errores	Filtrar por categoría	Ordenar resultados			Seleccionar método de pago	Realizar pago según método seleccionado	Respuesta al cliente con el número de seguimiento	Garantía y soporte técnico (SAT)			
	Verificar dirección de correo electrónico					Recepción de email de confirmación						
Versión 2	Foto de perfil	Filtrar por detalles específicos (color, tamaño, marca...)	Sugerencias de búsqueda "quizás quisiste decir..."	Valoraciones del producto	Galería de imágenes	Vales y descuentos			Valoración de productos			
			Paginar resultados	Opciones de compra avanzadas (personalización, proveedores...)								
Versión 3		Ofrecer resultados mientras se introducen los detalles	Vista previa rápida de producto	Botones de compartir y redes sociales	Botón añadir al carrito de compra							
			Visualización de anuncios	Botón comprar ahora								

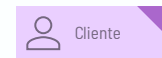
Viaje de usuario (user journey) completo
Gestión del sistema



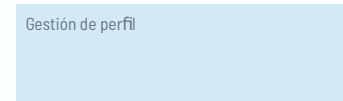
Viaje de usuario (user journey) parcial



Viaje de usuario (user journey) completo
Gestión del perfil



Viaje de usuario (user journey) parcial



Versión 1

Gestionar productos
y contenidos

Gestionar usuarios

Visualizar
estadísticas

Versión 1

Gestionar datos
personales y
tratamiento

Gestionar pedidos

Gestionar
notificaciones

Versión 2

Gestionar y moderar
valoraciones

Gestionar ofertas y
promociones

Versión 2

Foto de perfil

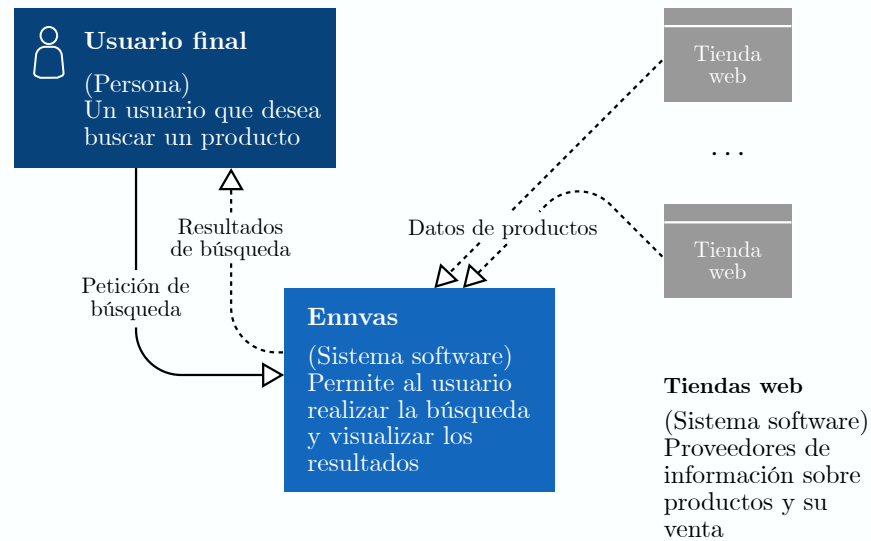
Versión 3



Gestionar anuncios

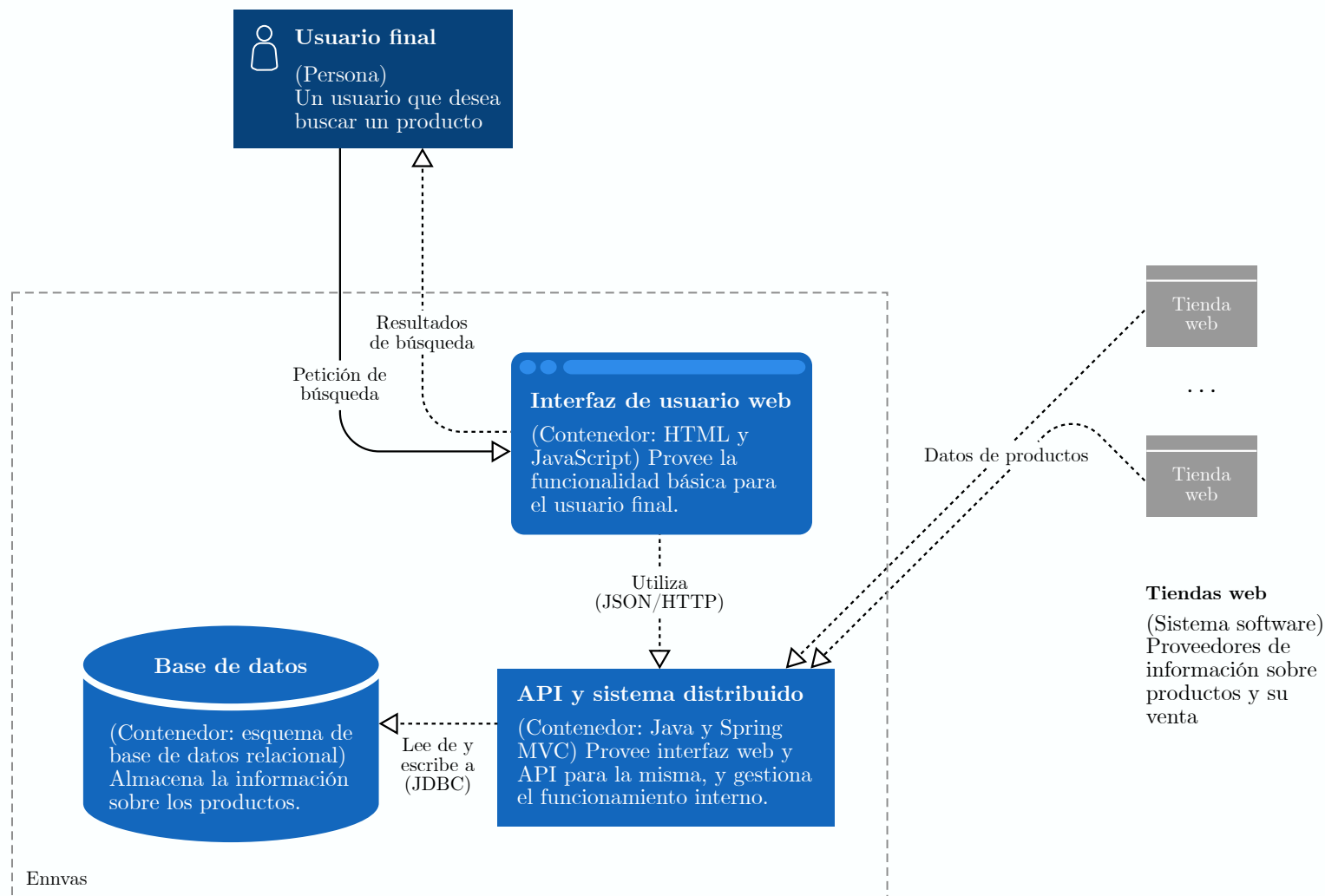
Integración con
redes sociales

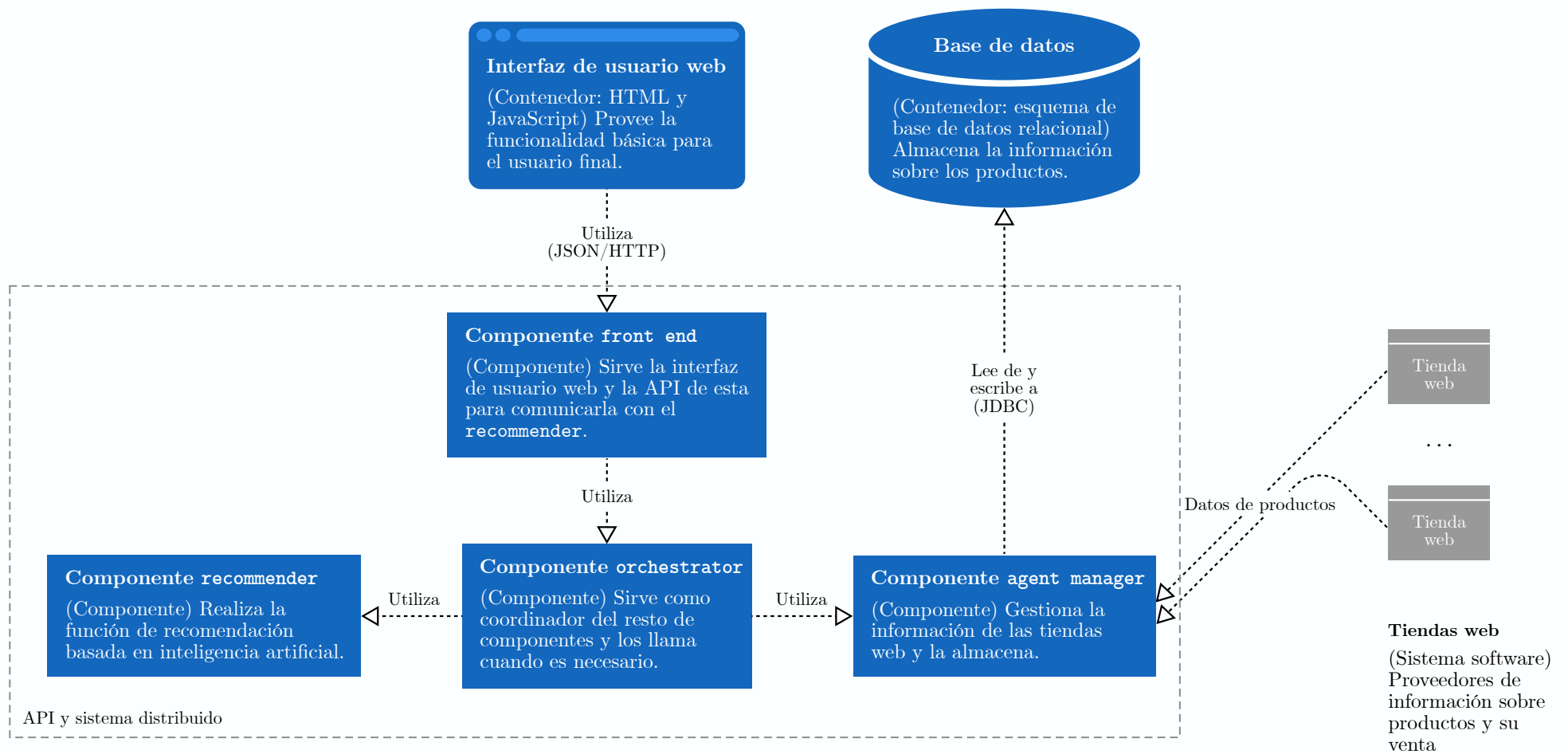
Gestionar idiomas

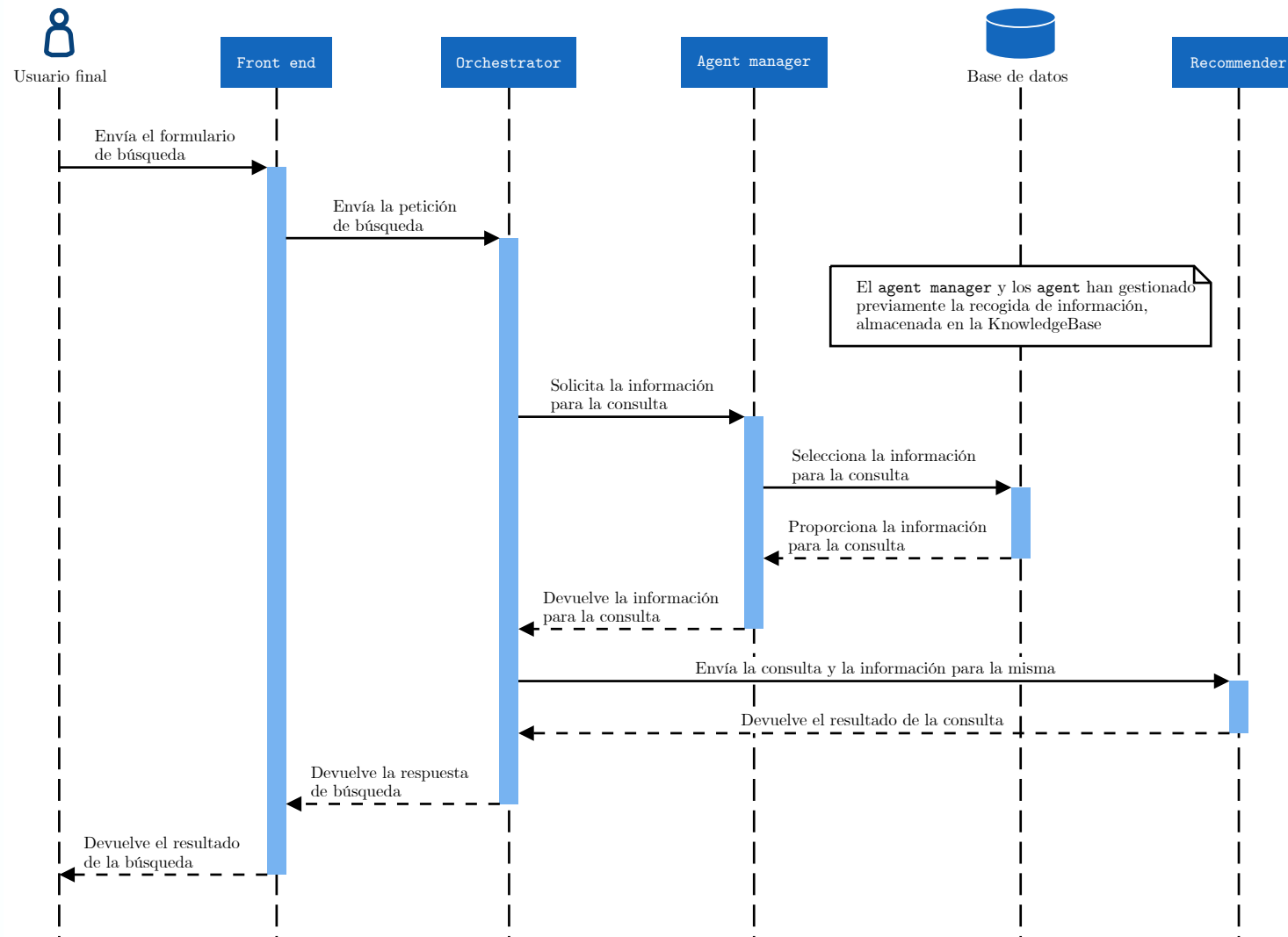
Versión 3

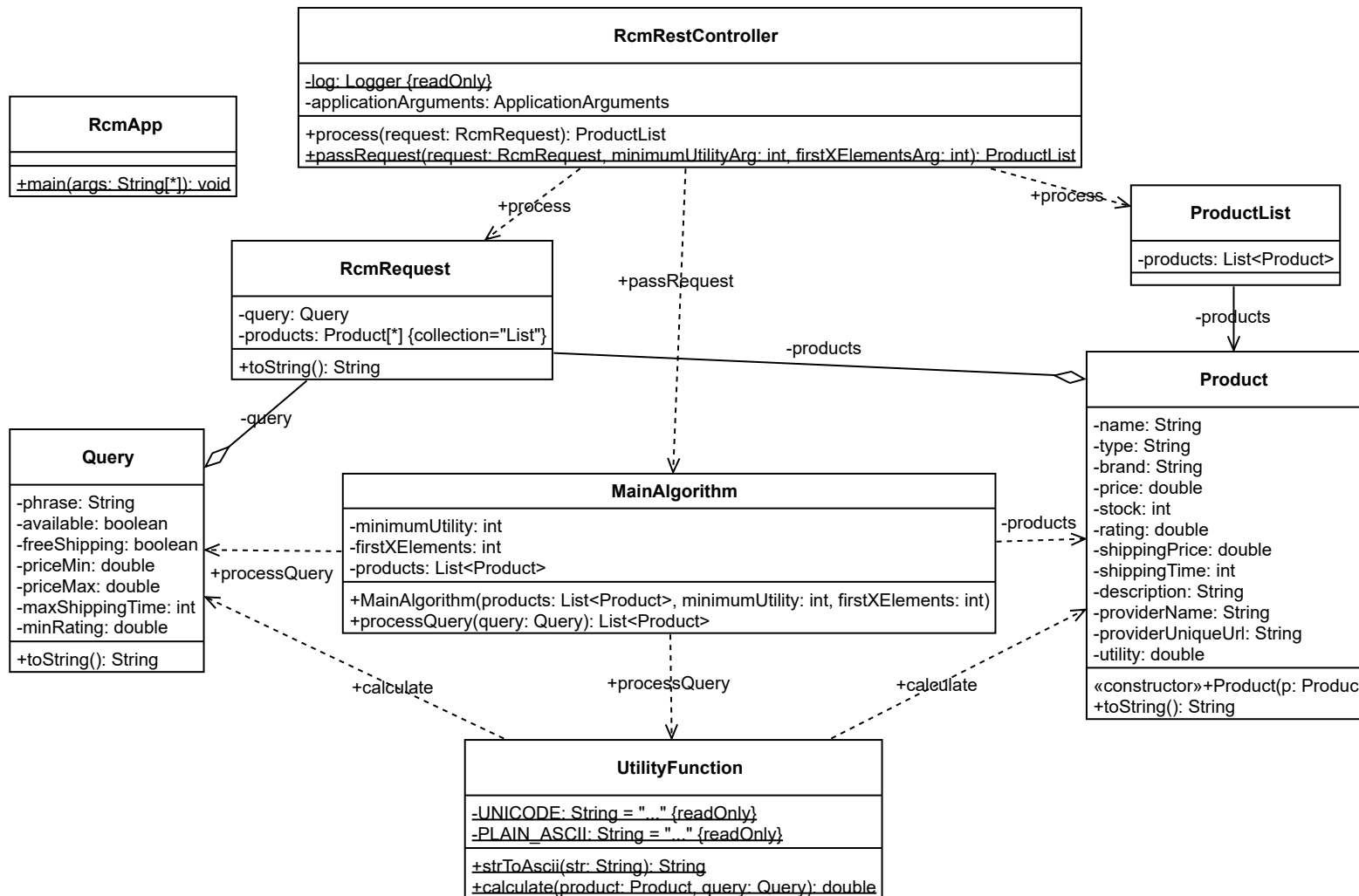


 Ennvas Sistema de tienda online multiagente basado en inteligencia artificial	Anexo 2	Proyecto de prácticas de Ingeniería del Software II	Profesor: Eugenio Pablo Concepción Cuevas	 UNIVERSIDAD COMPLUTENSE MADRID
	Modelo C4 - Diagrama nivel 1 - Contexto del sistema Página 1 de 1	Autores: Raquel Pérez González de Ossuna, Nicolás Pardina Popp, Juan Francisco Carrión Molina, Melany Daniela Chicaiza Quezada y Olga Posada Iglesias	Grupo B Convocatoria de junio - Curso 2019/2020 - Grado en Ingeniería Informática - Facultad de Informática	









Ennvas

Sistema de tienda
online multiagente
basado en inteligencia
artificial

Anexo 6

Diagrama UML de clases del
componente de recomendador.
Página 1 de 1

Proyecto de prácticas de Ingeniería del Software II

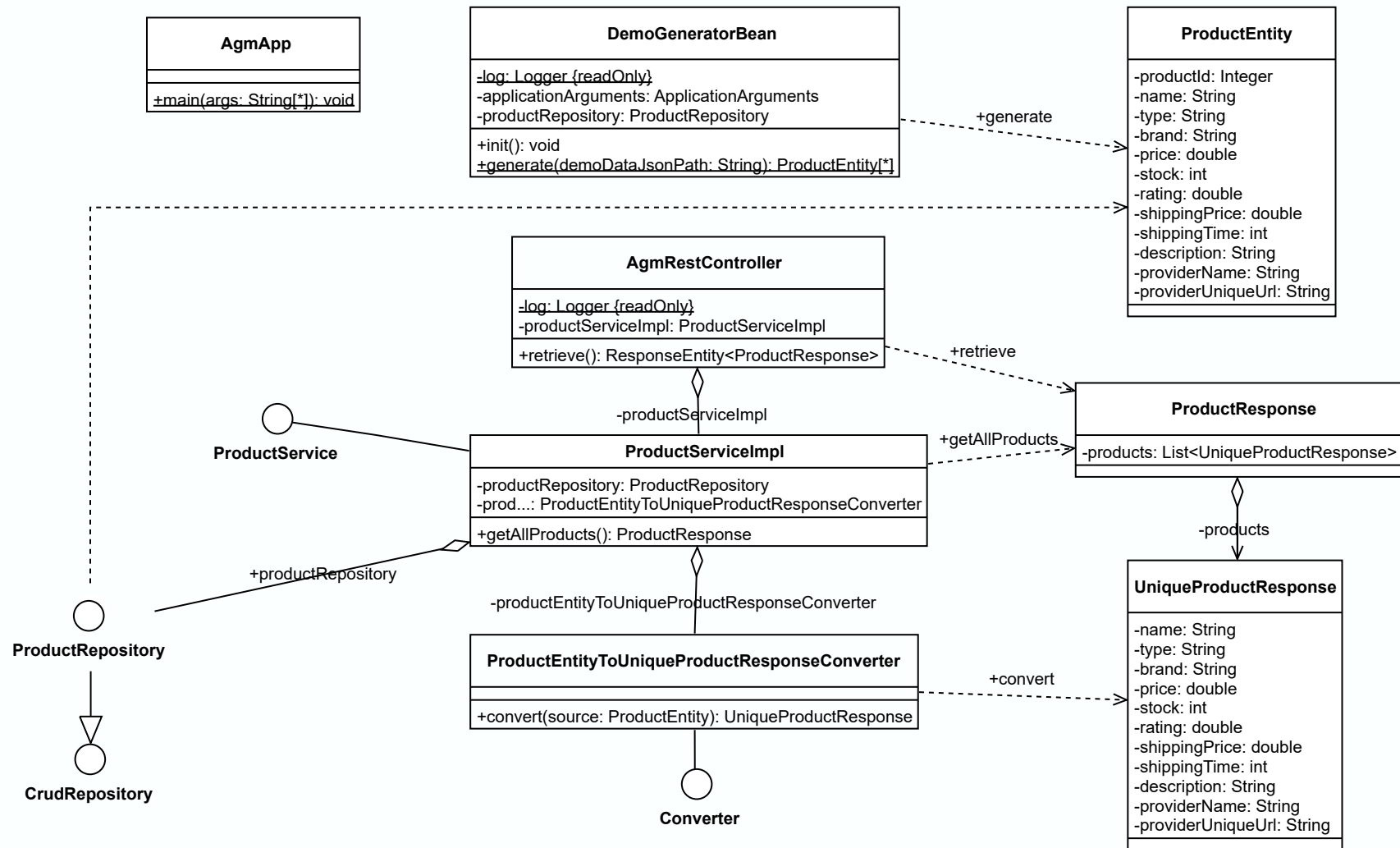
Autores: Raquel Pérez González de Ossuna,
Nicolás Pardina Popp, Juan Francisco Carrión Molina,
Melany Daniela Chicaiza Quezada y Olga Posada Iglesias

Profesor: Eugenio Pablo Concepción Cuevas

Grupo B
Convocatoria de junio - Curso 2019/2020 -
Grado en Ingeniería Informática - Facultad de Informática



**UNIVERSIDAD
COMPLUTENSE**
MADRID



Ennvas

Sistema de tienda
online multiagente
basado en inteligencia
artificial

Anexo 7

Diagrama UML de clases del
componente de gestión de agentes.
Página 1 de 1

Proyecto de prácticas de Ingeniería del Software II

Autores: Raquel Pérez González de Ossuna,
Nicolás Pardina Popp, Juan Francisco Carrión Molina,
Melany Daniela Chicaiza Quezada y Olga Posada Iglesias

Profesor: Eugenio Pablo Concepción Cuevas

Grupo B
Convocatoria de junio - Curso 2019/2020 -
Grado en Ingeniería Informática - Facultad de Informática



**UNIVERSIDAD
COMPLUTENSE
MADRID**

