PRÁCTICA DE PROGRAMACIÓN ORIENTADA A OBJETOS
-
CURSO 2022 / 2023

Nombre y apellidos: Salvador Moreno Sánchez
Centro: Madrid - Las Tablas

1. Objetivo.

El objetivo principal en la composición del programa ha sido el de digitalizar y automatizar las labores administrativas de la cooperativa, lo cual va a permitir, en primer lugar, el ahorro de costes económicos y temporales y, en segundo lugar, la posesión de una organización del organigrama de la cooperativa a nivel productores, así como de los agentes que participan en el transporte; además de ofrecer tanto a los distribuidores como a los consumidores finales una forma de interactuar de forma telemática con la cooperativa.

2. Especificaciones.

2.1. Funciones del sistema.

La idea inicial de desarrollo de la aplicación ha sido ambiciosa, ya que se ha querido desde un principio realizar un programa similar a aquellos que los usuarios se encuentran en su día a día, que permita el registro y el inicio de sesión de los mismos, pudiéndole denominar como MVP (*Minimum Viable Product*). De esta forma, el desarrollo siempre ha intentado tener en cuenta las posibles interacciones del usuario con la aplicación para que la navegación sea lo más intuitiva posible. A raíz de esto, se nos obligaba a pensar en cómo atender en la persistencia de datos, gran punto fuerte de nuestra aplicación porque, además de poseer un registro en archivos externos de todos los usuarios, su rol dentro del programa, sus datos, tablas de correspondencia de productos, empresas de logística, consumidores, etc., posee la capacidad de modificar dichos datos guardados y salvarlos en una base de datos interna al programa que aglutina en matrices los distintos objetos instanciados de cada uno de los actores de nuestra aplicación. Dicha base de datos se carga de los archivos externos una vez que se inicia el programa. Este significa un punto clave, ya que he considerado fundamental tener solidez en la estructuración de los datos y su relativo salvado, por lo que he dedicado gran parte de mi tiempo a dicha tarea. De hecho, en el main() del programa se pueden descomentar dos líneas relativas a las pruebas con la base de datos en pos de ver cómo se ha estructurado.

La comunicación entre los archivos externos y la base de datos, así como la escritura, lectura y modificación de los primeros; ha permitido elaborar un conjunto de métodos albergados en la clase *GestorArchivos* que ha facilitado la navegación interna por los mismos. Además, se ha decidido que todos los métodos sean de carácter *static* ya que no requieren el estado de un objeto en particular y no necesitan de crear un objeto de clase, lo cual nos ahorra memoria y optimiza el rendimiento de nuestras búsquedas y manipulación de archivos.

También, desde un inicio, se ha intentado desarrollar atendiendo a la escalabilidad y la cohesión del código, siguiendo un diseño Modelo-Vista-Controlador y persiguiendo una buena legibilidad del código dentro de las posibilidades y habilidades que se poseen.

Por otro lado, una vez que un usuario está registrado y ya ha dicho cuál es su rol (productor, empresa de logística o consumidor), además de cumplimentar una serie de datos relativos a su función; ya puede iniciar sesión con su email (el cual es único) y contraseña. Directamente será redirigido a su sección según su rol y ahí podrá realizar las funciones que le sean permitidas. Aquí debemos resaltar que sólo está desarrollada la sección del consumidor, donde se discrimina si es distribuidor o consumidor final para, luego, dirigirse a la sección del que corresponda. Se ha querido desarrollar de forma prioritaria esta parte ya que es fundamental para la realización de pedidos y almacenamiento de costes y beneficios para la cooperativa y las empresas de logística.

Una vez que se identifica al usuario como distribuidor o consumidor final se aplican los datos que le corresponden a según cual a la hora de calcular costes, tal y como se describe en el enunciado de la práctica. Se han tenido en cuenta las diversas variables según el tipo de producto (perecedero o no perecedero), así como sus datos fijos en relación a su valor de referencia, los cuales son extraídos de una tabla de correspondencia almacenada externamente y, también, en la base de datos asociada a cada producto. Por otra parte, se ha obviado la realización de una tabla de correspondencia de distancias de lugares de destino, ya que se ha optado por adjudicar de manera aleatoria los kilómetros de distancia en un rango de 200km.

2.2. Tareas no funcionales.

La ambición del proyecto y la falta de tiempo para su desarrollo ha provocado que la documentación del código sea limitada, siendo uno de los grandes puntos a mejorar en pos de que otros desarrolladores puedan seguir trabajando sobre el mismo.

Por otro lado, debo decir que comencé desarrollando en el entorno de desarrollo BlueJ, pero llegado un momento en el que el proyecto se hacía cada vez más grande, decidí migrarlo al entorno NetBeans debido a que la interfaz de BlueJ me estaba resultando cada vez menos intuitiva y eficiente. En cambio, considero que NetBeans ofrece una interfaz más completa y personalizable que se adapta a diferentes estilos de trabajo.

Respecto a aquellos puntos en los que no se ha podido incidir y, por ende, desarrollar, destacamos los siguientes:

- No se han podido extraer los importes obtenidos por cada uno de los productores (desglosados por producto) en los datos estadísticos.
- No se han podido calcular los beneficios de la cooperativa por cada uno de los productos en los datos estadísticos.
- No se ha podido extraer una gráfica con la evolución de los precios de referencias de cada producto.
- No se ha podido asociar de forma proporcionar la venta de un producto con cada productor según su aportación en hectáreas.
- No se ha implementado la opción de que aparezcan las hectáreas disponibles de cada producto para que el usuario vea cuánto material hay disponible.
- No se ha implementado un método para que los precios de los productos se revisen semanalmente y puedan experimentar subidas o bajadas.

3. Diagrama de clases.

Se ha intentado seguir el patrón de diseño Modelo-Vista-Controlador que, aunque no es contenido de la asignatura, se ha recopilado información y se ha intentado llevar a cabo. De esta forma, poseemos nuestro main() llamado *Cooperativa.java* en la carpeta *com;* nuestro Controlador en la carpeta *controller,* acompañado de las clases *BBDD*, *DatosEstadísticos* y *GestorArchivos*, las cuales se podrían albergar en otra carpeta para mayor organización y claridad, pero, aquí se ha optado por esta forma debido a que no resulta confuso; nuestro Modelo en la carpeta *model,* en la que se encuentran todas las clases relativas a los actores de nuestro programa (Consumidor, Productor, Pedido, etc.); y nuestra Vista en la carpeta *view.*

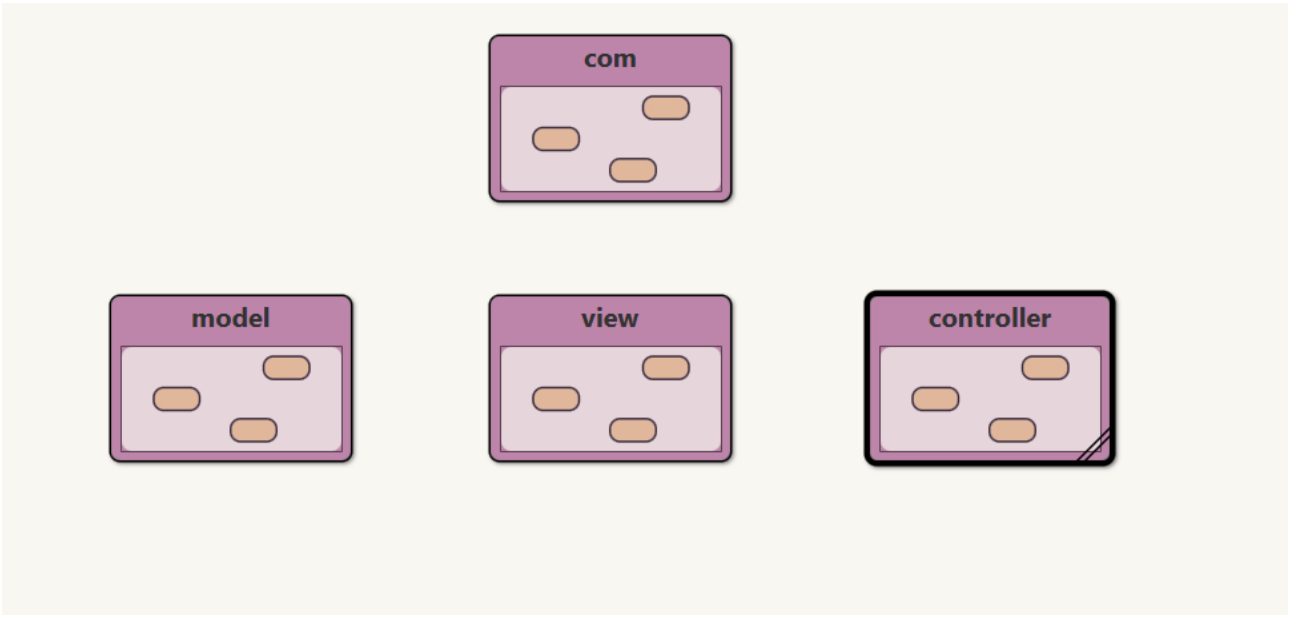

Figura 1. Estructura y diseño general del proyecto (MVC)

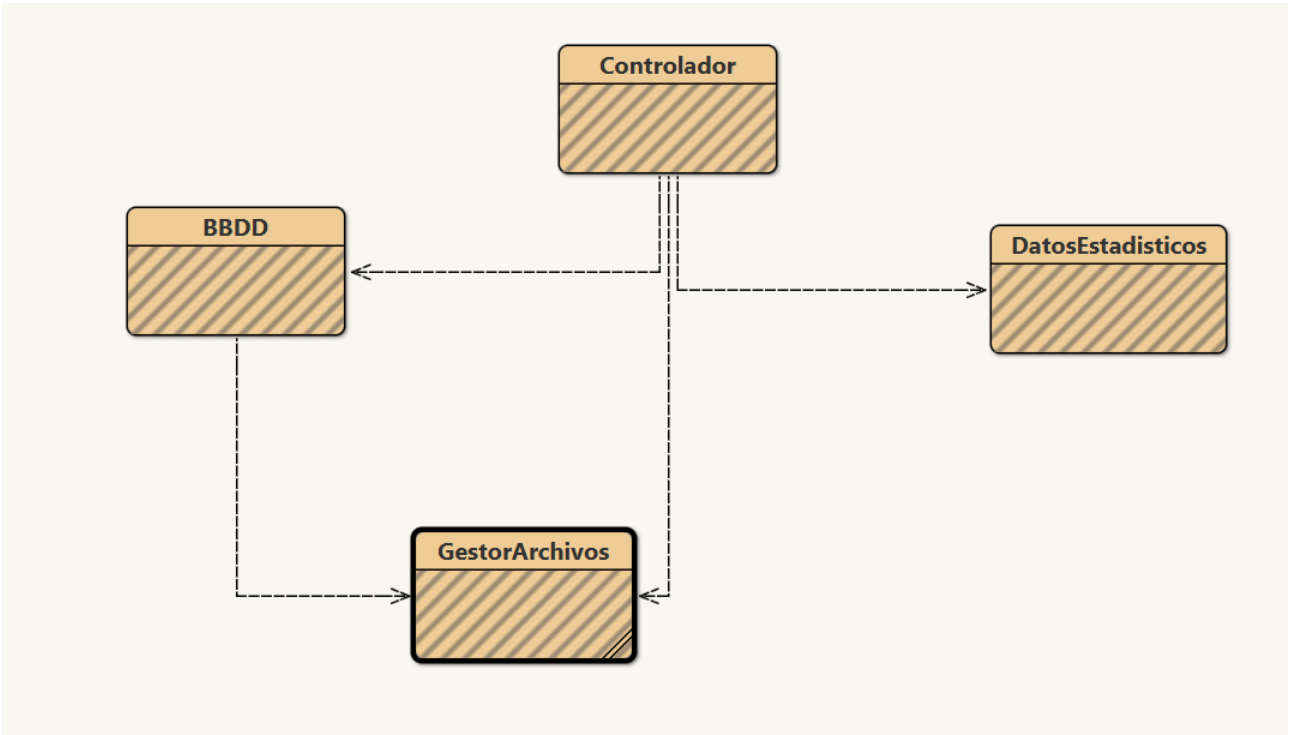

Figura 2. Carpeta *controller*

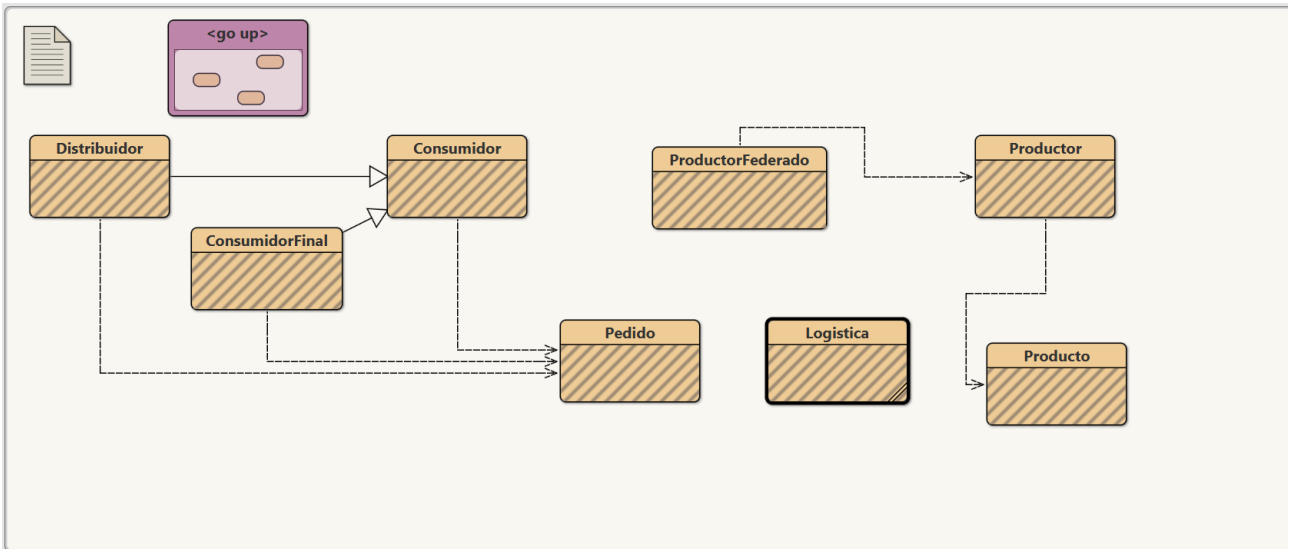

Figura 3. Carpeta *model*

Como podemos observar en la Figura 3, la única herencia empleada ha sido con la clase Consumidor. Se ha considerado así porque la especial organización de la información en los archivos externos y, sobre todo, en la base de datos nos ha permitido gestionar de manera óptima otras situaciones donde se podría haber considerado el uso de la herencia, tal como en la diferencia entre grandes y pequeños productores o en la pequeña o gran logística. Sin embargo, debido a las casuísticas más complejas y, sobre todo, a la diferencia de variables que poseen, se ha apostado por el uso de la herencia solamente en la clase Consumidor y no en el resto, ya que no era altamente necesario.

4. Conclusiones.

Aunque no he conseguido cubrir todas las funcionalidades y variables que se exigían en la práctica, me siento orgulloso del trabajo realizado y del esfuerzo que ha supuesto, lo cual me ha permitido explorar muchas de las formas de trabajar con el lenguaje de programación Java, descubrir distintos posibles caminos en el desarrollo y trabajar con multitud de variables y estructuras diferentes.

Además, aprendiendo del gran error que cometí en la práctica de la asignatura de programación del primer cuatrimestre, donde comencé a desarrollar el programa sin haber realizado una planificación previa en sucio y sin tener en cuenta exigencias posteriores del programa; esta vez, he dedicado gran parte del tiempo a elaborar esquemas en papel del posible funcionamiento de la aplicación y de las clases que quería elaborar, así como sus posibles pantallas de interacción con el usuario. Aún así, siempre surgen ideas, mejoras y cambios a lo largo del desarrollo que me han hecho rectificar y añadir atributos o métodos a clases, sobre todo cuando he querido desarrollar la parte de la extracción de datos estadísticos, la cual no tenía muy bien planificada y no había pensado mucho en ella durante la planificación del resto de las clases.