Distribució de productes a un supermercat

Entrega 2
Versió del lliurament: 2.0

PROP - QT2425

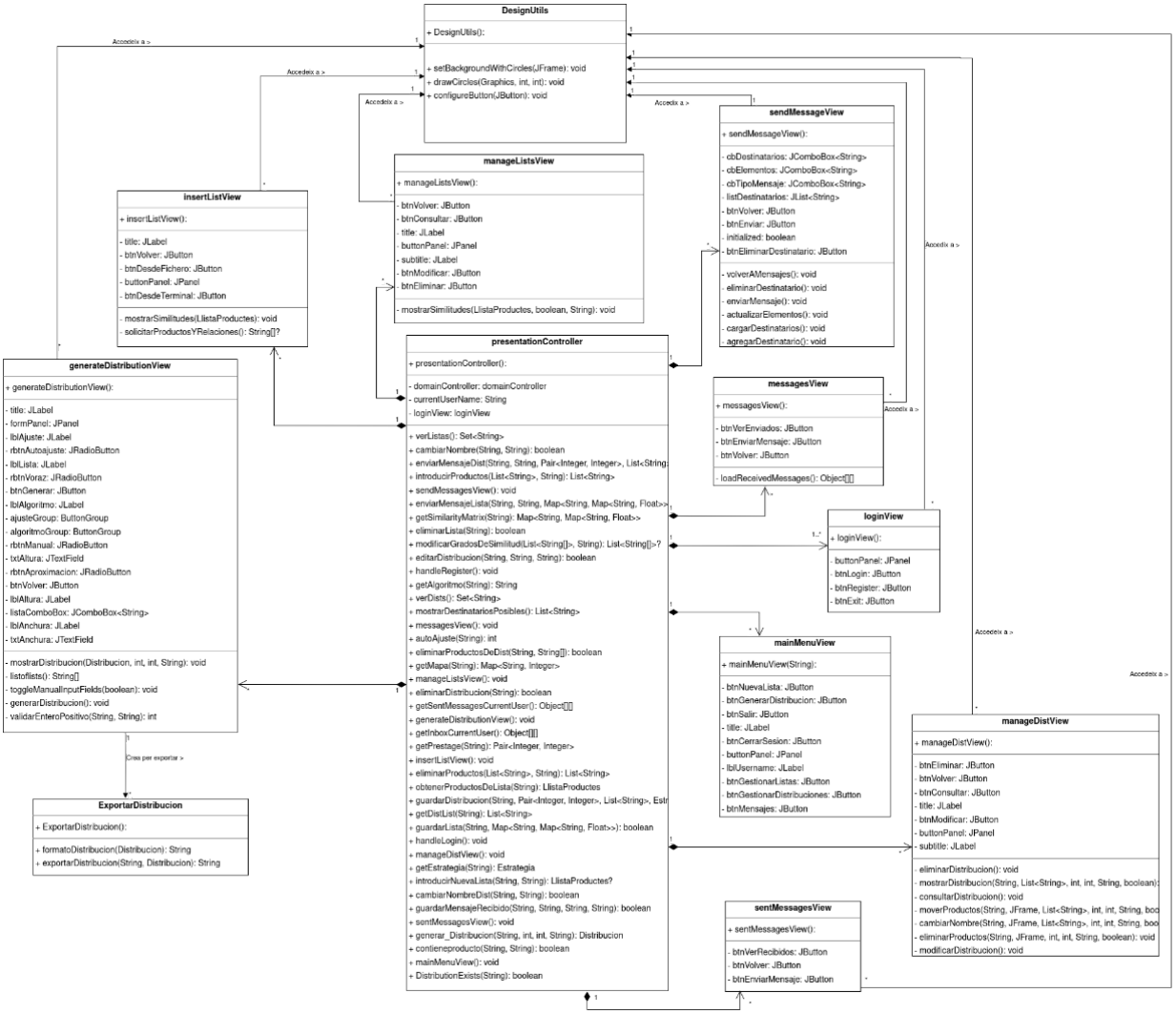
Subgrup 21.1

Roger Cot Londres: roger.cot@estudiantat.upc.edu
Nadia Khire: nadia.khier@estudiantat.upc.edu
David Mas Escudé: david.mas@estudiantat.upc.edu
David Sanz Martínez : david.sanz.martinez@estudiantat.upc.edu

1. Diagrama de les classes i controladors de la capa de presentació

A continuació mostrem el disseny del diagrama de les classes i controladors de la capa de presentació. El diagrama es pot trobar detallat a la carpeta DOCS.

1.1 Descripció del model conceptual



1.2. Explicació de les classes de presentació

Controlador de presentació

Descripció general de la classe

El controlador de presentació és el controlador principal del programa, és la primera classe en ser creada i la de la qual es partirà per la resta del programa. És l'encarregat de crear totes les vistes i comunicar-les. També controlar l'accés a la capa de domini de tal manera que si es vol accedir a la capa de domini o la de persistencia s'ha de passar per aquest controlador. A més a més també fa algun càlcul o crea algun objecte segons les dades de domini perquè les vistes obtinguin aquesta informació en el format que necessiten.

Atributs
Aquest classe només compta amb 3 atributs. Un controlador, una vista i el nom de l'usuari.

- **private static domainController:** Controlador de la capa de domini. Crea el controlador de persitència així que és l'accés a la capa de domini i a la capa de dades des de la capa de presentació.
- **private loginView:** Primera vista del programa. És la vista on l'usuari farà l'inici de sessió o el registre.
- **private static currentUserName:** És el nom que identifica a l'usuari que està utilitzant el programa.

Mètodes

Constructora

- **public presentationController()**
  Inicialitza el controlador de domini i una vista loginView. És el primer mètode que es crida al programa i només es crida 1 cop. És la constructora de presentationController.

<u>Creadores de vistes</u>

Creadores "handle"
Són mètodes que gestionen l'inici de sessió i el registre d'usuari però sempre acaben creant una nova vista.

- **public static void handleLogin()**
  Aquest mètode es cridat per loginView per gestionar el login de l'usuari. L'usuari decideix si vol fer login o no (amb un pop-up). Després, es mostra un pop-up on l'usuari pot introduir el seu nom d'usuari i contrasenya. En cas d'introduir credencials correctes crearà una vista de mainMenuView i currentUserName passarà a ser el seu userName, sinó crearà una nova vista loginView.
- **public static void handleRegister()**
  Aquest mètode es cridat per loginView (o per si mateix) per gestionar el registre d'un usuari. L'usuari decideix si vol registrar-se o no (amb un pop-up). Després, es mostra un pop-up on l'usuari pot introduir les seves dades d'usuari. En cas d'introduir tota la informació vàlida es crearà una vista de mainMenuView i currentUserName passarà a ser el seu userName. Si no s'introdueixen tots els camps es torna a cridar a handleRegister() i si hi ha algun error es crea una nova vista loginView.

Creadores de vistes normals

- **public static void mainMenuView()**
  Crea una nova vista mainMenuView (crida a la constructora de la vista) i li envia el currentUserName.
- **public static void insertListView()**
  Crea una nova vista insertListView (crida a la constructora de la vista).
- **public static void manageListsView()**
  Crea una nova vista manageListsView (crida a la constructora de la vista).
- **public static void generateDistributionView()**
  Crea una nova vista generateDistributionView (crida a la constructora de la vista).

- **public static void manageDistView()**
  Crea una nova vista manageDistView (crida a la constructora de la vista).
- **public static void sendMessagesView()**
  Crea una nova vista sendMessageView (crida a la constructora de la vista).
- **public static void sentMessagesview()**
  Crea una nova vista sentMessagesView (crida a la constructora de la vista).
- **public static void messagesView()**
  Crea una nova vista messagesView (crida a la constructora de la vista).

Gestió de la capa de domini

Creadores de la capa de domini

- **public static LlistaProductes introducirNuevaLista(String productos, String relaciones)**
  Aquest mètode agafa el String “productos” i el String “relaciones”, s’assegura que el format amb que han estat enviats els Strings sigui el correcte (si no llença excepcions) i crida al mètode nuevaLista del controlador de domini per crear la llista al domini (però no guardar-la a persistència) i aquest mètode retorna la matriu de graus de similitud “SM” de la llista creada. Amb aquesta matriu crea i retorna una nova LlistaProductes.
- **public static Dstribucion generar_Distribucion(String lista, int h, int w, String algoritmo)**
  El primer que fa aquest mètode és cridar al mètode generarDistribucion del controlador de domini perquè creï una nova distribució però no la guardi a persistència i retorna la llista de productes a la distribució (atribut “dist”). Amb la llista de productes i els altres atributs que té crea i retorna una nova distribució.

Destructores de la capa de domini

- **public static boolean eliminarLista(String lista)**
  Crida al controlador de domini per eliminar de domini i persistència la llista amb nom “lista”. Retorna un booleà indicant si s’ha pogut o no eliminar la llista.
- **public static List<String> eliminarProductos(List<String> productos, String listName)**
  Crida al controlador de domini per eliminar certs productes, els productes presents a la llista “productos”, de la llista “listName” i fer els canvis tant a domini com a persistència. Retorna la llista de productes efectivament eliminats, si no s’ha eliminat cap retorna null.
- **public static boolean eliminarProductosDeDist(String nombreDist, String[] prods)**
  Crida al controlador de domini per eliminar certs productes, els productes presents a la llista “prods”, de la distribució “nombreDist” i fer els canvis tant a domini com a persistència. Retorna un booleà indicant si s’ha pogut o no eliminar els productes i guardar els canvis.
- **public static boolean eliminarDistribucion(String nameDist)**
  Crida al controlador de domini per eliminar una distribució amb el nom “nameDist” de domini i persistència. Retorna un booleà indicant si s’ha pogut o no eliminar la distribució.

Getters de la capa de domini

- **public static Set<String> verListas()**
- **public static LlistaProductes obtenerProductesDeLista(String seleccion)**
- **public static Set<String> verDists()**
- **public static Object[][] getInboxCurrentUser()**
- **public static Object[][] getSentMessagesCurrentUser()**
- **public static Map<String, Map<String, Float>> getSimilarityMatrix(String name)**
- **public static Pair<Integer,Integer> getPrestatge(String name)**
- **public static List<String> getDistList(String nameDist)**
- **public static Estrategia getEstrategia(String nameDist)**
- **public static String getAlgoritmo(String nameDist)**
- **public static Map<String, Integer> getMapa(String nameDist)**
- **public static List<String mostrarDestinatariosPosibles()**

Consultores de la capa de domini

- **public static boolean DsitributionExists(String nameDist)**
  Retorna cert si a la capa de domini existeix una distribució amb nom “nameDist”, sinó retorna fals.
- **public static boolean contieneproducto(String nombreDist, String prod)**
  retorna cert si a la capa de domini existeix el producte “prod” dins de la distribució “nombreDist”, sinó retorna fals.
- **public static int autoAjuste(String lista)**
  Cirda al controlador de domini per obtenir la llista amb nom “lista” i calcula un enter necessari per fer l’autoajustament de la prestatgeria al generar una distribució.

Modificadores de la capa de domini

- **public static boolean cambiarNombre(String oldName, String newName)**
  Crida al controlador de domini per canviar el nom de la llista “oldName” per el seu nou nom “newName”. Retorna cert si s’ha pogut canviar i fals si no ha estat possible.
- **public static List<String> introducirProductos(List<String> productos, String lista)**
  Crida al controlador de domini per afegir els productes de la llista “productos” a la llista “lista”. Retorna els productes que ha pogut introduir, sinó retorna null.
- **public static List<String[]> modificarGradosDeSimilitud(List<String[]> relaciones, String lista)**
  Crida al controlador de domini per modificar els graus de similitud de la llista “lista” segons el List<String[]> relaciones. Retorna una llista amb les modificacions realitzades, o null.
- **public static boolean editarDistribucion(String nombreDist, String prod1, String prod2)**
  Crida al controlador de domini per intercanviar de lloc el producte “pord1” i el producte “prod2” a la distribució “nombreDist”. Retorna un cert si ha pogut fer els canvis a domini i persistència, sinó retorna fals.
- **public static cambiarboolean NombreDist(String oldName, String newName)**
  Crida al controlador de domini per canviar el nom de la distribució “oldName” per el seu nou nom “newName”. Retorna cert si s’ha pogut canviar i fals si no ha estat possible.

Mètodes que guarden a la capa de persistencia per primer cop

- **public static boolean guardarLista(String nombre, Map<String, Map<String, Float>> lista)**
  Crida al controlador de domini per crear i guardar a domini i persistència una llista amb nom “nombre” i matriu de similituds “lista”. Retorna cert si s’ha pogut guardar i fals si no s’ha pogut guardar.
- **public static boolean guardarDistribucion(String nameDist, Pair<Integer,Integer> prestatge, List<String> dist, Estrategia estrategia, Map<String,Integer> mapa)**
  Crida al controlador de domini per crear i guardar a domini i persistència una distribució amb nom “nameDist” i amb la resta de paràmetres com els seus atributs. Retorna cert si s’ha pogut guardar i fals si no s’ha pogut guardar.
- **public static void enviarMensajeLista(String destinatario, String name, Map<String, Map<String,Float>> similaritymatrix)**
  Crida al controlador de domini perquè creï un missatge d’aquest usuari amb una llista amb els paràmetres “name” i “similaritymatrix” amb el destinatari “destinatario”, envï el missatge i el guardi a persistència.
- **public static void enviarMensajeDist(String destinatario, String nameDist, Pair<Integer,Integer> prestatge, List<String> dist, Estrategia estrategia, Map<String, Integer> mapa)**
  Crida al controlador de domini perquè creï un missatge d’aquest usuari amb una distribució amb els paràmetres “nameDist”, “prestatge”, “dist”, “estrategia” i “mapa” amb el destinatari “destinatario”, envï el missatge i el guardi a persistència.
- **public static boolean guardarMensajeRecibido(String sender, String title, String type, String newName)**
  Crida al controlador de domini perquè guardi al usuari corresponent (domini i persistència) la informació d’un missatge rebut. Retorna true si s’ha pogut guardar i fals si no s’ha pogut guardar.

DesignUtils

Descripció general de la classe

Aquest classe encapsula mètodes que s’utilitzen a l’hora de decorar les vistes principals. O sigui, configurar el background d’una vista al fons rosa amb cercles d’un rosa més fosc i configurar els botons d’aquests dos colors també.

Mètodes

- **public static void drawCircles(graphics g, int width, int height)**
  Dibuixa cercles aleatòriament per al fons d’un rosa més fosc que el color principal del fons.
- **public static void setBackgroundWithCircles(JFrame frame)**
  Posa el fons de color rosa i li afegeix cercles creats mitjançant drawCircles. Aquest mètode es cridat per totes les vistes per configurar els seus JPanels del fons.
- **public static void configureButton(JButton button)**
  Configura els botons perquè tinguin els dos colors de rosa utilitzats per al fons, la font de la lletra que volem i treure l’efecte de clic.

ExportarDistribucion

Descripció general de la classe
Aquest classe encapsula dos mètodes utilitzats per exportar distribucions.

Mètodes
- **public String formatoDistribucion(Distribucion distribucion)**
  Genera un String amb el contingut de la distribució en el format que hem decidit per poder exportar la distribucó a un arxiu de text.
- **public String exportarDistribucion(String nombre, Distribucion distribucion)**
  Exporta una distribució segons un fileChooser i retorna la ruta completa del arxiu exportat, retorna null si ha ocorregut un error.

loginView

Descripció general de la classe
És la primera vista amb la que l'usuari interactua. Gestiona l'entrada a la resta de funcionalitats del programa mitjançant el sistema d'usuari. Bàsicament aquí l'usuari pot iniciar sessió si ja té creat un usuari, registrar-se o tancar el programa.

Atributs
- **private final JPanel buttonPanel**
  Panell principal de la vista on es troben tots els objectes interactuables o que es volen mostrar a l'usuari (botons, labels, etc).
- **private final JButton btnLogin**
  Botó per procedir a l'inici de sessió.
- **private final JButton btnRegister**
  Botó per procedir al registre d'usuari.
- **private final JButton btnExit**
  Botó per tancar el programa.

Mètode
- **public loginView()**
  Mètode únic i principal de la vista, vindria a ser la constructora. Aquí es configura la vista, és on es fa visible i és on estan implementades les funcionalitats del botons.

mainMenuView

Descripció general de la classe
Aquesta és la vista principal del programa. Fa de menú per accedir a totes les funcionalitats del programa després d'haver fet l'inici de sessió o registre exitosament.

Atributs
- **private final JPanel buttonPanel**
- Panell principal de la vista on es troben tots els objectes interactuables o que es volen mostrar a l'usuari (botons, labels, etc).
- **private final JLabel title**
- **private final JLabel lblUsername**

Botons
Cadascun dels següents botons porta a una funcionalitat diferent i per tant a les diferents vistes excepte els últims dos, el penúltim permet tancar sessió i l'últim tancar el programa directament.

- **private final JButton btnNuevaLista**
  Botó per insertar una nova llista, ens porta a la vista insertListView.
- **private final JButton btnGestionarListas**
  Botó per gestionar les llistes creades, ens porta a la vista manageListsView.
- **private final JButton btnGenerarDistribucion**
  Botó per generar una nova distribució, ens porta a la vista generateDistView.
- **private final JButton btnGestionarDistribuciones**
  Botó per gestionar les distribucions creades, ens porta a la vista managesDistView.
- **private final JButton btnMensajes**
  Botó per gestionar els missatges, ens porta a la vista messagesView.
- **private final JButton btnCerrarSesion**
- **private final JButton btnSalir**

Mètode

- **public mainMenuView(String username)**
  Mètode únic i principal de la vista, vindria a ser la constructora. Aquí es configura la vista, és on es fa visible i és on estan implementades les funcionalitats del botons.

insertListView

Descripció general de la classe
Aquesta vista és la que permet a l'usuari introduir noves llistes de productes. L'usuari aquí introdueix les dades per crear una nova llista i guarda la llista.

Atributs

- **private final JPanel buttonPanel**
  Panell principal de la vista on es troben tots els objectes interactuables o que es volen mostrar a l'usuari (botons, labels, etc).
- **private final JLabel title**
- **private final JButton btnDesdeTerminal**
  Botó per decidir introduir una llista de productes escrivint els noms dels productes i els seus graus de similitud.
- **private final JButton btnDesdeFichero**
  Botó per introduir la informació d'una nova llista de productes mitjançant un fitxer de l'usuari.
- **private final JButton btnVolver**
  Botó per retornar al menú principal.

Mètodes

- **public insertListView()**
  Mètode principal de la vista, vindria a ser la constructora. Aquí es configura la vista, és on es fa visible i és on estan implementades (o es fan les crides a les funcions que ho fan) les funcionalitats del botons.
- **private void mostrarSimilitudes(LlistaProductes lista)**

Aquest mètode mostra un pop-up amb la llista introduïda (productes i graus de similitud). Es crida abans de que l'usuari guardi la llista perquè vegi visualment quina és la llista que ha introduït.

- **private String[] solicitarProductosYRelaciones()**
  Aquest mètode és el que crea els pop-ups per introduir els productes i els seus graus de similitud mitjançant un formulari gràfic. Aquest mètode només es crida quan es decideix introduir els productes mitjançant l'opció "Desde terminal" ja que és l'únic cas on l'usuari ha d'escriure els productes i graus de similitud que vol per a la seva nova llista.

manageListView

Descripció general de la classe
Aquesta vista és la que permet a l'usuari realitzar les funcionalitats dins del que seria gestionar les llistes de productes ja creades, per consultar-les, eliminar-les o modificar-les.

Atributs

- **private final JPanel buttonPanel**
  Panell principal de la vista on es troben tots els objectes interactuables o que es volen mostrar a l'usuari (botons, labels, etc).
- **private final JLabel title**
- **private final JLabel subtitle**
- **private final JButton btnConsultar**
  Botó que ens porta a la funcionalitat de consultar una llista.
- **private final JButton btnModificar**
  Botó que ens porta a la funcionalitat de modificar una llista. Dins de modificar una llista es pot cambiar el nom de la llista, introduir més productes a la llista, eliminar productes de la llista i modificar els graus de similitud dels productes de la llista.
- **private final JButton btnEliminar**
  Botó que ens porta a la funcionalitat de eliminar una llista.
- **private final JButton btnVolver**
  Botó per retornar al menú principal.

Mètodes

- **public manageListsView()**
  Mètode principal de la vista, vindria a ser la constructora. Aquí es configura la vista, és on es fa visible i és on estan implementades (o es fan les crides a les funcions que ho fan) les funcionalitats del botons.
- **private void mostrarSimilitudes(LlistaProductes lista, boolean editable, String accion)**
  Aquest mètode mostra un pop-up amb la llista seleccionada. Es crida quan es consulta una llista, quan s'introdueixen nous productes, quan s'eliminen productes o quan es modifiquen els graus de similitud. En el cas de consultar mostra la llista, en el cas d'introduir nous productes també però abans d'introduir els nous canvis. En el cas d'eliminar productes o modificar graus de similitud no només mostra la llista fa tota la seva funcionalitat. En el primer cas l'usuari selecciona quins productes s'eliminen i després s'eliminen (o no si falla alguna cosa) i en el segon cas pot modificar els graus de similitud i els canvis s'apliquen (o no si falla alguna cosa).

generateDistributionView

Descripció general de la classe
Aquesta vista és la que permet a l'usuari generar noves distribucions. L'usuari aquí introdueix la llista de productes que vol utilitzar, si vol definir l'usuari la prestatgeria o que s'autoajusti i amb quin algoritme vol generar la distribució.

Atributs
- **private final JPanel formPanel**
  Panell principal de la vista on es troben tots els objectes interactuables o que es volen mostrar a l'usuari (botons, labels, etc).

Labels que es mostren a l'usuari
- **private final JLabel title**
- **private final JLabel lblLista**
- **private final JLabel lblAjuste**
- **private final JLabel lblAlgoritmo**
- **private final JLabel lblAltura**
- **private final JLabel lblAnchura**

- **private final JComboBox<String> listaComboBox**
  ComboBox per escollir la llista de productes a utilitzar.

Els següents objectes són per definir la prestageria.
- **private final JRadioButton rbtnAutoajuste**
- **private final JRadioButton rbtnManual**
- **private final ButtonGroup ajusteGroup**
- **private final JTextField txtAltura**
- **private final JTextField txtAnchura**

Els següents objectes són per escollir l'algoritme que es vol utilitzar.
- **private final JRadioButton rbtnVoraz**
- **private final JRadiobutton rbtnAproximacion**
- **private final ButtonGroup algoritmoGroup**

- **private final JButton btnGenerar**
  Botó que ens porta a generar la distribució segons la informació introduïda, si aquesta és incorrecta o no està completa no es generarà la distribució.
- **private final JButton btnVolver**
  Botó per retornar al menú principal.

Mètodes
- **private String[] listoflists()**
  Mètode per seleccionar les llistes creades abans de mostrar la vista, si l'usuari no ha creat llistes no li deixa accedir a la funcionalitat de la llista.
- **public generateDistributionView()**
  Mètode principal de la vista, vindria a ser la constructora. Aquí es configura la vista, és on es fa visible i és on estan implementades (o es fan les crides a les funcions que ho fan) les funcionalitats del botons i els altres objectes.
- **private void toggleManualInputFields(boolean enable)**

Mètode mostra o deixa de mostrar els objectes per introduir manualment els valors per crear el prestatge de la distribució manualment.

- **private void generarDistribucion()**
  Mètode cridat des de generateDistributionView() quan es clica el botó de “Generar” (btnGenerar). Aquest mètode implementa tota la generació de la distribució i com es guarda. A més a més també crida al mètode mostrarDistribucion per mostrar la distribució resultant.
- **private int validarEnteroPositivo(string input, String campo)**
  Mètode que s'assegura que els valors de la prestatgeria introduïts manualment siguin correctes. Agafa l’input com a String, el transforma en enter i s’assegura de que sigui positiu, sinó llença una excepció.
- **private void mostrarDistribucion(Distribucion distribucion, int h, int w, String algoritmo)**
  Mètode que mostra la distribució generada i implementa guardar distribució, guardar i exportar i distribució, i cancelar després de mostrar-la.

manageDistView

Descripció general de la classe
Aquesta vista és la que permet a l’usuari realitzar les funcionalitats dins del que seria gestionar les distribucions ja generades, per consultar-les, eliminar-les o modificar-les.

Atributs

- **private final JPanel buttonPanel**
- Panell principal de la vista on es troben tots els objectes interactuables o que es volen mostrar a l’usuari (botons, labels, etc).
- **private final JLabel title**
- **private final JLabel subtitle**
- **private final JButton btnConsultar**
  Botó que ens porta a la funcionalitat de consultar una distribució.
- **private final JButton btnModificar**
  Botó que ens porta a la funcionalitat de modificar una distribució. Dins de modificar una distribució es pot cambiar el nom de la distribució, es pot editar la distribució per canviar dos productes de lloc i es pot eliminar productes de la distribució.
- **private final JButton btnEliminar**
  Botó que ens porta a la funcionalitat de eliminar una distribució.
- **private final JButton btnVolver**
  Botó per retornar al menú principal.

Mètodes

- **public manageDistView()**
  Mètode principal de la vista, vindria a ser la constructora. Aquí es configura la vista, és on es fa visible i és on es fan les crides a les funcions que fan la implementació de les funcionalitats del botons.
- **private void consultarDistribucion()**
  Mètode que implementa la funcionalitat de consultar una distribució. Mostra la distribució seleccionada.
- **private void modificarDistribucion()**

Mètode que fa les crides necessàries i part de la funcionalitat de modificar un adistribució. Mostra la distribució seleccionada.

- **private void eliminarDsitribucion()**
  Mètode que implementa la funcionalitat de eliminar una distribució. Mostra la distribució seleccionada.
- **private void mostrarDistribucion(String nombre, List<String> dist, int h, int w, String algoritmo, boolean modificar)**
  Mètode cridat per consultarDistribucion per mostrar la distribució seleccionada (la que té nom “nombre”) i cridat també per modificarDistribucion que en aquest cas mostra la distribució seleccionada (la que té nom “nombre”) i fa la crida a un mètode que implementa una de les 3 opcions de modificar distribució.
- **private void moverProductos(String nombre, JFrame parentFrame, List<String> dist, int h, int w, String algoritmo, boolean modificar)**
  Mètode que implementa intercanviar 2 productes de lloc en la distribució amb nom “nombre”, és una de les 3 opcions de modificar distribució.
- **private void eliminarProductos(String nombre, JFrame parentFrame, int h, int w, String algoritmo, boolean modificar)**
  Mètode que implementa eliminar productes de la distribució amb nom “nombre”, és una de les 3 opcions de modificar distribució.
- **private void cambiarNombre(String nombre, JFrame parentFrame, List<String> dist, int h, int w, String algoritmo, boolean modificar)**
  Mètode que implementa canviar el nom de la distribució amb nom “nombre”, és una de les 3 opcions de modificar distribució.

messagesView

Descripció general de la classe
Aquesta vista és la que permet a l’usuari realitzar les funcionalitats relacionades amb missatges. Ja sigui mirar els enviar un missatge o mirar els missatges enviats i rebuts.

Atributs

- **private final JButton btnVerEnviados**
  Botó que ens porta a la funcionalitat de veure els missatges enviats.
- **private final JButton btnEnviarMensaje**
  Botó que ens porta a la funcionalitat de enviar un missatge.
- **private final JButton btnVolver**
  Botó per retornar al menú principal.

Mètodes

- **public messagesView()**
  Mètode principal de la vista, vindria a ser la constructora. Aquí es configura la vista, és on es fa visible i és on estan implementades les funcionalitats del botons i es mostren els missatges rebuts (i els pots consultar).
- **private Object[][] loadReceiveMessages()**
  Mètode que carga els missatges rebuts de l’usuari actual.

sendMessageView

Descripció general de la classe
Aquesta vista és la que permet a l'usuari enviar un missatge. Es poden enviar llistes o distribucions a altres usuaris.

Atributs

- **private final JComboBox<String> cbDestinatarios**
  Permet escollir un destinatari.
- **private final JComboBox<String> cbTipoMensaje**
  Permet escollir el tipus de missatge.
- **private final JComboBox<String> cbElementos**
  Permet escollir l'objecte a enviar.
- **private final JButton btnEliminarDestinatario**
  Botó que permet eliminar els desirnataris seleccionats.
- **private final JButton btnVolver**
  Botó per retornar al menú principal.
- **private final JButton btnEnviar**
  Botó que envia el missatge amb la informació seleccionada.
- **private final JList<String> listDestinatarios**
  Llista dels destinataris, aquí es poden seleccionar els usuaris a eliminar del missatge.
- **private boolean initialized**
  Aquest booleà servei per evitar que el primer destinatari s'afegeixi per defecte.

Mètodes

- **public sendMessageView()**
  Mètode principal de la vista, vindria a ser la constructora. Aquí es configura la vista, és on es fa visible i és on estan implementades (o es fan les crides a les funcions que ho fan) les funcionalitats del botons i els altres objectes.
- **private void cargarDestinatarios()**
  Mètode que carrega els possibles destinataris del missatge, o sigui els altres usuaris, i els afegeix al JComboBox cbDestinataris.
- **private void actualizarElementos()**
  Canvia els objectes de cbElementos, canvia entre distribucions i llistes de productes.
- **private void agregarDestinatario()**
  Afegeix el destinatari seleccionat a cbDestinatarios a listDestinatarios, o sigui, afegeix aquest usuari als destinataris del missatge.
- **private void eliminarDestinatario()**
  Elimina al destinatari seleccionat a listDestiantaios dels usuaris destinataris del missatge.
- **private void enviarMensaje()**
  Mètode que implementa l'enviament del missatge. S'assegura que la informació sigui correcta i envia el missatge. Aquest mètode es cridat quan es clica el botó Enviar (btnEnviar).
- **private void volverAMensajes()**
  Botó per tornar a la vista de missatges messagesView.

sentMessagesView

Descripció general de la classe
Aquesta vista és la que permet a l'usuari veure els missatges que ha enviat. Pot també anar a les altes dos vistes de missatges (messagesView i sendmessageView), o sigui, pot anar a enviar missatge i a veure missatges rebuts. A més a més, por tornar al menú principal.

Atributs

- **private final JButton btnVerRecibidos**
  Botó que porta a la vista per veure els missatges rebuts messagesView, o sigui, per anar a veure els missatges rebuts.
- **private final JButton btnEnviarMensaje**
  otó que ens porta a la funcionalitat de enviar un missatge.
- **private final JButton btnVolver**
  Botó per retornar al menú principal.

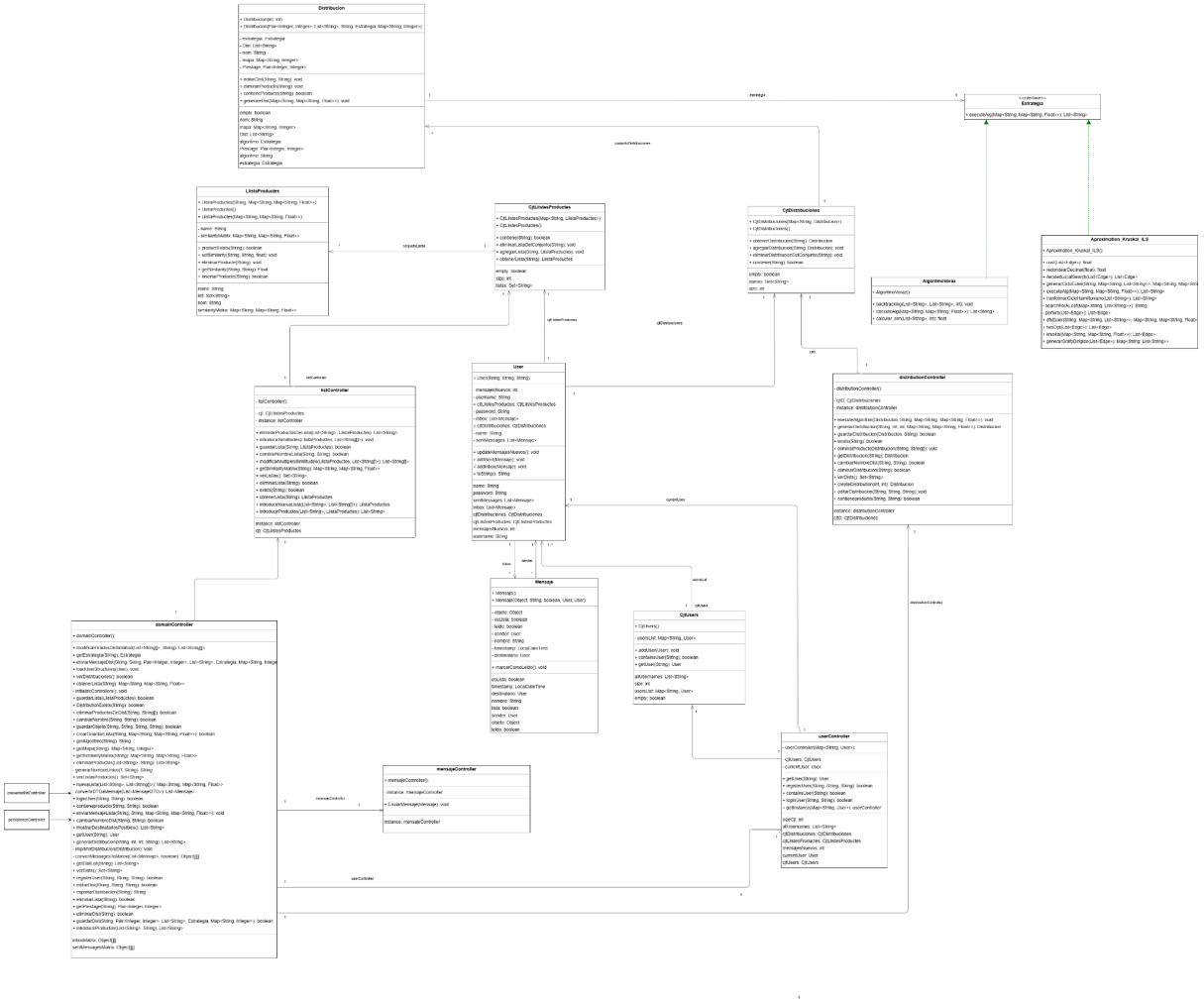
Mètode

- **public sent MessagesView()**
  Mètode únic i principal de la vista, vindria a ser la constructora. Aquí es configura la vista, és on es fa visible i és on estan implementades les funcionalitats del botons i mostra els missatges enviats (i els pots consultar).

2. Diagrama de les classes i controladors de la capa de domini

2.1 Descripció model conceptual

A continuació es presenta el Diagrama de les classes i controladors de la capa de domini, que es troba també a DOCS/DiagramaDeCapaDeDomini.svg



2.2 Explicació de les classes del domini

2.2.1. User

La classe User representa un usuari en el sistema. Aquesta classe gestiona la informació personal de l'usuari, com el seu nom, nom d'usuari i contrasenya. També administra la seva comunicació, com ara missatges rebuts i enviats, i permet accedir als conjunts de distribucions i llistes de productes associades a l'usuari.

- Atributs:

- String username: Nom d'usuari.
- String name: Nom real de l'usuari.

- String password: Contrasenya de l'usuari.
- int mensajesNuevos: Nombre de missatges nous sense llegir.
- List<Mensaje> inbox: Llista de missatges rebuts.
- List<Mensaje> sentMessages: Llista de missatges enviats.
- CjtDistribuciones cjtDistribuciones: Conjunt de distribucions associades a l'usuari.
- CjtLlistesProductes cjtLlistesProductes: Conjunt de llistes de productes associades a l'usuari.

    - Mètodes:

- User(String username, String name, String password):
  Constructor que inicialitza l'usuari amb el nom d'usuari, el nom real i la contrasenya. També inicialitza les llistes de missatges i els conjunts de distribucions i llistes de productes.
- int getMensajesNuevos():
  Retorna el nombre de missatges nous sense llegir i després restableix el comptador a zero.
- void updateMensajesNuevos():
  Restableix el comptador de missatges nous a zero.
- void addInbox(Mensaje mensaje):
  Afegeix un missatge a la bandeja d'entrada i incrementa el comptador de missatges nous.
- void addSent(Mensaje mensaje):
  Afegeix un missatge a la llista de missatges enviats.
- CjtLlistesProductes getCjtLlistesProductes():
  Retorna el conjunt de llistes de productes associades a l'usuari.
- void setCjtLlistesProductes(CjtLlistesProductes cjtLlistes):
  Estableix el conjunt de llistes de productes de l'usuari.
- CjtDistribuciones getCjtDistribuciones():
  Retorna el conjunt de distribucions associades a l'usuari.
- void setCjtDistribuciones(CjtDistribuciones cjtDistribuciones):
  Estableix el conjunt de distribucions de l'usuari.
- String toString():
  Retorna una representació en cadena de la informació de l'usuari.

2.2.2. CjtUsers

La classe **CjtUsers** representa un conjunt de tots els usuaris del sistema. Aquesta classe permet gestionar llistes d'usuaris mitjançant un mapa on les claus són els noms d'usuari i els valors són objectes de la classe **User**.

- Atributs:

- **Map<String, User> usersList**: Mapa que conté els usuaris. La clau és el nom d'usuari, i el valor és l'objecte **User** associat.

- Mètodes:

- **CjtUsers()**:
  Constructor que inicialitza un conjunt buit d'usuaris.
- **boolean containsUser(String username)**:
  Comprova si existeix un usuari amb el nom d'usuari indicat al conjunt.
  - **Paràmetre**: *username* - Nom d'usuari a verificar.
  - **Retorn**: *true* si l'usuari existeix, *false* en cas contrari.
- **void addUser(User user)**:
  Afegeix un usuari al conjunt. El nom d'usuari ha de ser únic.
  - **Paràmetre**: *user* - Objecte **User** que es vol afegir.
- **User getUser(String username)**:
  Retorna l'usuari associat al nom d'usuari indicat.
  - **Paràmetre**: *username* - Nom d'usuari.
  - **Retorn**: L'objecte **User** corresponent, o *null* si no existeix.
- **int getSize()**:
  Retorna el nombre total d'usuaris en el conjunt.
- **List<String> getAllUsernames()**:
  Retorna una llista amb tots els noms d'usuari presents al conjunt.
- **boolean isEmpty()**:
  Comprova si el conjunt d'usuaris està buit.
  - **Retorn**: *true* si el conjunt està buit, *false* en cas contrari.

Descripció general:
Aquesta classe centralitza la gestió dels usuaris en un únic conjunt. Permet operacions bàsiques per mantenir i consultar els usuaris registrats, assegurant un accés ràpid gràcies a l'ús d'un mapa. Es pot utilitzar en controladors per gestionar les accions relacionades amb la sessió i la interacció entre usuaris.

2.2.3. LlistaProductes

La classe **LlistaProductes** representa una col·lecció de productes amb una matriu de similitud associada. Aquesta matriu emmagatzema el grau de similitud (GdS) entre els productes.

● Atributs:

- **String name**: Nom de la llista de productes.
- **Map<String, Map<String, Float>> similarityMatrix**: Matriu que associa cada producte amb un mapa dels graus de similitud respecte als altres productes.

● Constructors:

- **LlistaProductes()**:
  Inicialitza una llista de productes amb nom "*Sin_nombre*" i una matriu buida.

    - **LlistaProductes(Map<String, Map<String, Float>> similarityMatrix)**: Inicialitza la llista amb nom "*Sin_nombre*" i la matriu de similitud proporcionada.
- **LlistaProductes(String name, Map<String, Map<String, Float>> similarityMatrix)**:
  Inicialitza la llista amb el nom i la matriu de similitud proporcionats.

  ● Mètodes:

Informació:

- **Set<String> getList()**:
  Retorna el conjunt de noms de productes presents a la llista.
- **Map<String, Map<String, Float>> getSimilarityMatrix()**:
  Retorna la matriu de similitud completa.
- **boolean productExists(String product)**:
  Comprova si un producte existeix a la llista.
    - **Paràmetre**: *product* - Nom del producte.
    - **Retorn**: *true* si el producte existeix, *false* en cas contrari.
- **Float getSimilarity(String prod1, String prod2)**:
  Retorna el grau de similitud entre dos productes.
    - **Paràmetres**:
        - *prod1* - Nom del primer producte.
        - *prod2* - Nom del segon producte.
    - **Retorn**: Grau de similitud o *null* si algun producte no existeix.

Manipulació de la llista:

- **boolean insertarProducte(String product)**:
  Afegeix un producte a la llista si no existeix.
    - **Paràmetre**: *product* - Nom del producte a inserir.
    - **Retorn**: *true* si el producte es va afegir, *false* si ja existia.
- **void eliminarProducte(String product)**:
  Elimina un producte de la llista i totes les seves associacions de similitud.
    - **Paràmetre**: *product* - Nom del producte a eliminar.
- **void setSimilarity(String prod1, String prod2, float GdS)**:
  Estableix el grau de similitud entre dos productes.
    - **Paràmetres**:
        - *prod1* - Nom del primer producte.
        - *prod2* - Nom del segon producte.
        - *GdS* - Grau de similitud entre els productes.

Altres:

- **String getName()**:
  Retorna el nom de la llista de productes.
- **void setNom(String n)**:
  Assigna un nou nom a la llista de productes.

- **void setSimilarityMatrix(Map<String, Map<String, Float>> similarityMatrix)**:
  Assigna una nova matriu de similitud a la llista.

Descripció general:
Aquesta classe és útil per gestionar col·leccions de productes i definir relacions entre ells basades en graus de similitud. Això permet modelar associacions o agrupaments en funció d'aquests valors, essent especialment útil per a aplicacions de recomanació, classificació o cerca.

2.2.4. CjtLlistesProductes

La classe **CjtLlistesProductes** gestiona un conjunt de llistes de productes, representades per la classe **LlistaProductes**. Permet manipular múltiples llistes mitjançant un mapa que associa noms de llistes amb les seves instàncies corresponents.

● Atributs:

- **Map<String, LlistaProductes> conjuntoListas**:
  Mapa que associa el nom de cada llista de productes amb la seva instància corresponent.

● Constructors:

- **CjtLlistesProductes()**:
  Inicialitza un conjunt buit de llistes de productes.
- **CjtLlistesProductes(Map<String, LlistaProductes> conjuntoListas)**:
  Inicialitza el conjunt amb les llistes proporcionades en el mapa.

● Mètodes:

Informació:

- **int getSize()**:
  Retorna el nombre total de llistes de productes al conjunt.
- **boolean contiene(String name)**:
  Comprova si una llista amb el nom especificat existeix al conjunt.
  - **Paràmetre**: *name* - Nom de la llista a verificar.
  - **Retorn**: *true* si la llista existeix, *false* en cas contrari.
- **boolean isEmpty()**:
  Comprova si el conjunt de llistes està buit.
- **Set<String> getListas()**:
  Retorna el conjunt de noms de les llistes de productes presents al conjunt.
- **LlistaProductes obtenerLista(String nombre)**:
  Retorna la llista de productes associada al nom especificat.
  - **Paràmetre**: *nombre* - Nom de la llista.
  - **Retorn**: Instància de **LlistaProductes** o *null* si no existeix.

Manipulació del conjunt:

- **void agregarLista(String name, LlistaProductes lista)**:
  Afegeix una nova llista de productes al conjunt.
    - **Paràmetres**:
        - *name* - Nom de la llista.
        - *lista* - Instància de **LlistaProductes** que representa la llista.
- **void eliminarListaDelConjunto(String nombre)**:
  Elimina una llista de productes del conjunt.
    - **Paràmetre**: *nombre* - Nom de la llista a eliminar.

Descripció general:

Aquesta classe serveix com a contenidor centralitzat per gestionar múltiples llistes de productes (**LlistaProductes**). És especialment útil en aplicacions que requereixen organitzar i manipular diferents col·leccions de productes per a funcionalitats com comparacions, recomanacions o categoritzacions.

Ús potencial

- Organitzar múltiples categories o grups de productes, com "Electrodomèstics", "Mobles", o "Decoració".
- Facilitar l'accés ràpid a qualsevol llista per nom.
- Mantenir consistència i simplificar l'estructura del codi quan es treballa amb diverses col·leccions de productes.

2.2.5 Distribucion

Classe que representa un conjunt de productes organitzats dins d'una distribució espacial definida per dimensions i gestionada per un algoritme que s'escull amb una *estratègia*. Aquesta classe permet la creació, manipulació i consulta de distribucions.

La classe **Distribució** representa una distribució de productes en un espai bidimensional, gestionant les seves posicions, noms i un algorisme d'estratègia per generar l'ordre dels productes.

Atributs:

- **Pair<Integer, Integer> Prestage**: Dimensions de la distribució representades com *<alçada, longitud>*.
- **List<String> Dist**: Llista que conté els noms dels productes en la distribució.
- **String nom**: Nom de la distribució.
- **Estrategia estrategia**: Objecte que defineix l'algorisme d'estratègia utilitzat per generar la distribució.
- **Map<String, Integer> mapa**: Mapa que associa els noms dels productes amb les seves posicions en la distribució.

Constructors:

- **Distribució(Pair<Integer, Integer> prestage, List<String> dist, String nom, Estrategia estrategia, Map<String, Integer> mapa)**: Inicialitza una distribució amb dimensions, llista de productes, nom, estratègia i mapa de posicions.
- **Distribució(int h, int w)**: Inicialitza una distribució especificant només les dimensions *<alçada, longitud>*.

Mètodes d'informació:

- **boolean isEmpty()**: Verifica si la distribució està buida. Retorna *true* si no hi ha productes en la distribució, *false* en cas contrari.
- **boolean contéProducte(String name)**: Comprova si un producte amb el nom especificat està present a la distribució. Rep com a paràmetre *name* (nom del producte a cercar) i retorna *true* si el producte existeix o *false* en cas contrari.

Mètodes de manipulació:

- **void generateDist(Map<String, Map<String, Float>> LlistaProductes)**: Genera la llista de productes i el mapa de posicions en la distribució utilitzant l'algorisme d'estratègia. Rep com a paràmetre *LlistaProductes*, que és un mapa amb els productes i les seves similituds per a l'algorisme.
- **void editarDist(String nom1, String nom2)**: Intercanvia les posicions de dos productes en la distribució. Rep com a paràmetres *nom1* (nom del primer producte) i *nom2* (nom del segon producte).
- **void eliminarProducte(String nom)**: Elimina un producte de la distribució, reemplaçant-lo amb *null*. Rep com a paràmetre *nom* (nom del producte a eliminar).

Descripció general:

La classe **Distribución** permet gestionar de forma dinàmica col·leccions de productes disposats en un espai virtual o físic. És especialment útil en aplicacions de logística, planificació d'espais, o fins i tot jocs i simulacions.

Ús potencial

- Organització de productes en un magatzem o botiga.
- Visualització i optimització d'espais en aplicacions de decoració.
- Simulació de disposicions segons estratègies d'optimització.

2.2.6 CjtDistribuciones

La classe **CjtDistribucions** representa un conjunt de distribucions, gestionant-les a través d'un mapa que associa noms únics amb instàncies de la classe **Distribució**.

Atributs

- **Map<String, Distribució> conjuntDistribucions**: Mapa que emmagatzema les distribucions associades als seus noms.

Constructors

- **CjtDistribucions()**: Inicialitza un conjunt buit de distribucions.
- **CjtDistribucions(Map<String, Distribució> conjuntDistribucions)**: Inicialitza un conjunt de distribucions a partir d'un mapa donat.

Mètodes

- **int getSize()**: Retorna el nombre total de distribucions en el conjunt.
- **Set<String> getNames()**: Retorna un conjunt amb els noms de totes les distribucions emmagatzemades.
- **boolean conté(String nom)**: Verifica si existeix una distribució amb el nom especificat. Retorna *true* si la distribució existeix o *false* en cas contrari.
- **boolean isEmpty()**: Comprova si el conjunt de distribucions està buit. Retorna *true* si no hi ha distribucions al conjunt, o *false* en cas contrari.
- **void afegirDistribució(String nom, Distribució distribució)**: Afegeix una nova distribució al conjunt. Rep com a paràmetres el *nom* de la distribució i l'objecte **Distribució** corresponent.
- **Distribució obtenirDistribució(String nom)**: Retorna l'objecte **Distribució** associat al nom donat. Si no existeix, retorna *null*.
- **void eliminarDistribució(String nom)**: Elimina del conjunt la distribució associada al nom especificat.

Descripció general:

CjtDistribuciones simplifica la gestió d'un conjunt de distribucions. Cada distribució és únicament identificada pel seu nom i pot ser manipulada de manera eficient gràcies a l'ús d'un mapa (estructura clau-valor).

Ús potencial

- Administració d'una base de dades d'espais decoratius, magatzems o plans de distribució.
- Desenvolupament d'aplicacions que requereixen una gestió eficient de col·leccions d'objectes estructurats.
- Organització i recuperació d'informació d'una col·lecció de configuracions personalitzades per a decoració o simulació logística.

2.2.7 Missatge

La classe **Missatge** representa un missatge que conté un objecte (pot ser una llista o una distribució), juntament amb informació sobre el remitent, destinatari, estat de lectura i la marca temporal de creació.

Atributs

- **Object objecte**: Llista o distribució continguda en el missatge.
- **String nom**: Nom de l'objecte (llista o distribució).

- **boolean esLlista**: Indica si l'objecte és una llista (*true*) o una distribució (*false*).
- **boolean llegit**: Indica si el missatge ha estat llegit (*true*) o no (*false*).
- **LocalDateTime timestamp**: Marca temporal de creació del missatge.
- **User remitent**: Usuari que envia el missatge.
- **User destinatari**: Usuari que rep el missatge.

Constructors

- **Missatge()**: Constructor buit que inicialitza un missatge sense dades.
- **Missatge(Object objecte, String nom, boolean esLlista, User remitent, User destinatari)**: Constructor que inicialitza un missatge amb l'objecte, nom, tipus, remitent i destinatari. Assigna la marca temporal actual i marca el missatge com no llegit.

Mètodes

- **LocalDateTime getTimestamp()**: Retorna la marca temporal de creació del missatge.
- **User getRemitent()**: Retorna l'usuari que ha enviat el missatge.
- **Object getObjecte()**: Retorna l'objecte contingut en el missatge.
- **boolean isLlista()**: Retorna *true* si el missatge conté una llista, *false* si conté una distribució.
- **boolean isLlegit()**: Retorna *true* si el missatge ha estat llegit, *false* en cas contrari.
- **void marcarComLlegit()**: Marca el missatge com a llegit.
- **User getDestinatari()**: Retorna l'usuari destinatari del missatge.
- **String getNom()**: Retorna el nom de l'objecte contingut en el missatge.
- **void setObjecte(Object objecte)**: Estableix un nou objecte per al missatge.
- **void setNom(String nom)**: Estableix un nou nom per al missatge.
- **void setEsLlista(boolean esLlista)**: Defineix si el missatge conté una llista (*true*) o una distribució (*false*).
- **void setLlegit(boolean llegit)**: Defineix si el missatge ha estat llegit o no.
- **void setTimestamp(LocalDateTime timestamp)**: Estableix una nova marca temporal per al missatge.
- **void setRemitent(User remitent)**: Defineix un nou remitent per al missatge.
- **void setDestinatari(User destinatari)**: Defineix un nou destinatari per al missatge.

2.2.8 AlgoritmoVoraz

Descripció:

El **AlgoritmoVoraz** és un algorisme que utilitza **backtracking** per trobar la millor ordre de productes en funció de la similitud entre ells. Busca maximitzar la suma de les similituds entre productes consecutius en una llista

Atributs:

- **best_sim**: Emmagatzema la major similitud acumulada trobada.
- **lordenada**: Llista amb la millor ordre de productes calculat.
- **Matrix**: Matriu que emmagatzema les similituds entre els productes.
- **visitados**: Llista per rastrejar els productes ja processats.

Funcionalitat:

Explicat al punt 4.2.1

2.2.9 Aproximation_Kruskal_ILS

Descripció:

L'algorisme **Aproximation_Kruskal_ILS** implementa una aproximació per resoldre problemes de rutes i cicles en grafos utilitzant l'algorisme de **Kruskal** combinat amb la **cerca local iterativa (ILS)**. L'objectiu és trobar solucions aproximades minimitzant els costos, generant **arbres d'expansió mínima (MST)** i transformant-los en **cicles hamiltonians**.

Atributs i Classes Internes:

- **Edge**: Representa una aresta entre dos productes amb un pes associat.
- **MFS**: Estructura per gestionar conjunts disjunts mitjançant unió i cerca (per a l'algorisme de Kruskal).
- **parent**: Mapa que emmagatzema l'arrel de cada conjunt.
- **rank**: Mapa que emmagatzema el rang de cada conjunt per optimitzar la unió.

Explicat al punt 4.2.2

2.3. Explicació dels controladors de domini

2.3.1 domain controller

Descripció general de la classe

La classe **domainController** és un controlador principal que actua com a mediador entre diferents subcontroladors.. S'encarrega de coordinar accions relacionades amb la gestió d'usuaris, llistes de productes, distribucions i missatges, integrant també la capa de persistència i presentació.

Atributs

Els atributs de la classe són principalment instàncies de controladors especialitzats i dades necessàries per gestionar les operacions.

- **userController**: Controlador responsable de la gestió dels usuaris.
- **listController**: Controlador que gestiona les llistes de productes.
- **distributionController**: Controlador que gestiona les distribucions.

- **mensajeController**: Controlador encarregat de gestionar els missatges (inbox, enviats, etc.).
- **persistenceController**: Controlador per accedir a la capa de persistència.
- **newMessages**: Enter que guarda el nombre de nous missatges de l'usuari.

Mètodes principals

Constructor

- **public domainController()**
  Inicialitza tots els controladors, carregant dades des de fitxers de configuració i inicialitzant el comptador de missatges nous.

Mètodes per inicialitzar controladors

- **private void initializeControllers()**
  Configura els controladors de llistes i distribucions amb les dades de l'usuari actual, incloent les llistes de productes i distribucions.

Gestió d'usuaris

- **public boolean registerUser(String username, String name, String password)**
  Registra un nou usuari i inicialitza els controladors si el registre té èxit.
- **public boolean loginUser(String username, String password)**
  Permet que un usuari iniciï sessió, carregant les seves estructures i dades personals.
- **public void loadUserStructures(User user)**
  Carrega des de fitxers les llistes de productes, distribucions, inbox, i missatges enviats d'un usuari concret.

Gestió de llistes de productes

- **public Map<String, Map<String, Float>> nuevaLista(List<String> productos, List<String[]> relaciones)**
  Permet crear una nova llista de productes amb les relacions entre productes, i retorna la matriu de similitud.
- **public boolean guardarLista(LlistaProductes lista)**
  Guarda una llista de productes en la persistència.
- **public boolean eliminarLista(String lista)**
  Elimina una llista de productes tant de memòria com de la persistència.
- **public List<String> introducirProductos(List<String> productos, String listName)**
  Afegeix productes nous a una llista existent.

Gestió de distribucions

- **public boolean guardarDist(String nameDist, Pair<Integer, Integer> prestage, List<String> dist, Estrategia estrategia, Map<String, Integer> mapa)**
  Crea i guarda una distribució amb les dades especificades.

- **public boolean eliminarDist(String dist)**
  Elimina una distribució de la memòria i persistència.
- **public List<String> generarDistribucion(String nom_llista, int h, int w, String algoritmo)**
  Genera una nova distribució a partir d'una llista de productes i un algoritme.

Gestió de missatges

- **public void enviarMensajeLista(String destinatario, String name, Map<String, Map<String, Float>> similarityMatrix)**
  Envia un missatge que conté una llista de productes a un destinatari.
- **public Object[][] getInboxMatrix()**
  Retorna els missatges de l'inbox en forma de matriu, incloent informació com l'enviador, el tipus i la data.

2.3.2 UserController

La classe UserController implementa el patró Singleton per garantir una instància única que gestiona els usuaris registrats i l'usuari que actualment està loguejat.

Atributs:

- **private static UserController instance**: Instància única de la classe segons el patró Singleton.
- **private CjtUsers cjtUsers**: Conjunt d'usuaris registrats al sistema.
- **private User currentUser**: Usuari actualment loguejat.

Mètodes:

- **private UserController(Map<String, User> users)**: Constructora privada que inicialitza el conjunt d'usuaris a partir d'un mapa amb les dades inicials.
- **public static UserController getInstance(Map<String, User> users)**: Retorna la instància única de la classe, creant-la si no existeix.
- **public User getCurrentUser()**: Retorna l'usuari actualment loguejat.
- **public User getUser(String username)**: Cerca un usuari pel seu nom d'usuari i el retorna; retorna *null* si no existeix.
- **public CjtUsers getCjtUsers()**: Retorna el conjunt d'usuaris gestionat per la classe.
- **public int getSizeCjt()**: Retorna el nombre total d'usuaris registrats.
- **public List<String> getAllUsernames()**: Retorna una llista amb tots els noms d'usuari registrats.
- **int getMensajesNuevos()**: Retorna el nombre de missatges no llegits de l'usuari actual.
- **public CjtLlistesProductes getCjtLlistesProductes()**: Retorna el conjunt de llistes de productes de l'usuari actual.
- **public CjtDistribuciones getCjtDistribuciones()**: Retorna el conjunt de distribucions de l'usuari actual.

- **public boolean containsUser(String username)**: Verifica si un usuari existeix al conjunt d'usuaris.
- **public boolean registerUser(String username, String name, String password)**: Registra un nou usuari al sistema, validant que no existeixi ja. Retorna *true* si el registre és exitós, o *false* en cas contrari.
- **public boolean loginUser(String username, String password)**: Permet iniciar sessió amb un nom d'usuari i contrasenya. Retorna *true* si l'inici de sessió és correcte, o *false* en cas contrari.

Comportament general

Aquesta classe centralitza la gestió d'usuaris del sistema, oferint funcionalitats per consultar, modificar i autenticar usuaris. També gestiona aspectes específics com els missatges no llegits i els conjunts associats de llistes de productes i distribucions. Amb la implementació Singleton, es garanteix que només hi hagi una instància en funcionament, assegurant la consistència i la seguretat de les dades gestionades.

.

2.3.3 listController

La classe **ListController** implementa un controlador centralitzat que gestiona totes les operacions relacionades amb les llistes de productes. Aquesta classe segueix el patró **Singleton**, garantint que només hi hagi una instància activa en tot el sistema.

Atributs:

- **private static ListController instance**: Instància única de la classe per implementar el patró Singleton.
- **private CjtLlistesProductes cjt**: Conjunt de llistes de productes que seran gestionades pel controlador.

Mètodes:

- **private ListController()**: Constructora privada per evitar la creació d'instàncies externes.
- **public static ListController getInstance()**: Retorna la instància única de la classe, creant-la si no existeix.
- **public void setCjt(CjtLlistesProductes cjt)**: Assigna el conjunt de llistes de productes a gestionar.
- **public LlistaProductes obtenerLista(String nombre)**: Retorna una llista de productes pel seu nom.
- **public Map<String, Map<String, Float>> getSimilarityMatrix(String nombre)**: Retorna la matriu de similituds de productes d'una llista.
- **public boolean exists(String name)**: Comprova si existeix una llista amb el nom proporcionat.

- **public LlistaProductes introducirNuevaLista(List<String> productos, List<String[]> relaciones)**: Crea una nova llista de productes, afegint productes i similituds inicials si són proporcionats.
- **public List<String> introducirProductos(List<String> productos, LlistaProductes LP)**: Introdueix una llista de productes a una llista específica.
- **public void introducirSimilitudes(LlistaProductes lista, List<String[]> relaciones)**: Defineix les similituds entre productes d'una llista. Llenca excepcions en cas d'errors de format o valors incorrectes.
- **public List<String[]> modificarMultiplesSimilitudes(LlistaProductes lista, List<String[]> relaciones)**: Modifica simultàniament múltiples graus de similitud d'una llista i retorna les modificacions realitzades.
- **public Set<String> verListas()**: Retorna el conjunt de noms de totes les llistes existents.
- **public boolean guardarLista(String name, LlistaProductes lista)**: Desa una nova llista al conjunt o la modifica si ja existeix.
- **public List<String> eliminarProductosDeLista(List<String> productos, LlistaProductes lista)**: Elimina una llista de productes específica d'una llista.
- **public boolean cambiarNombreLista(String nombreAntiguo, String nombreNuevo)**: Modifica el nom d'una llista, assegurant-se que el nou nom no existeixi ja.
- **public boolean eliminarLista(String nombre)**: Elimina una llista del conjunt després de verificar-ne l'existència.

Comportament general:

Aquesta classe facilita la gestió de llistes de productes mitjançant operacions com la creació, consulta, modificació i eliminació. També gestiona graus de similitud entre productes i permet l'edició massiva d'aquests. Amb l'ús del patró **Singleton**, garanteix que totes les operacions es centralitzen en una única instància, evitant inconsistències en la gestió del conjunt de llistes.

2.3.4 distributionController

La classe **distributionController** implementa un controlador que gestiona un conjunt de distribucions. Aquesta classe segueix el patró Singleton, garantint que només hi hagi una única instància al llarg de l'execució del programa. És responsable d'afegir, modificar, eliminar, consultar i exportar distribucions.

Atributs

- **private static distributionController instance**: Instància única del controlador, necessària per implementar el patró Singleton.
- **private CjtDistribuciones cjtD**: Conjunt de distribucions que el controlador gestiona. Aquesta estructura conté les distribucions identificades pel seu nom.

Mètodes

1. **Constructores**

- **private distributionController()**: Constructor privat per restringir la creació directa de noves instàncies de la classe.

2. **Patró Singleton**

public static distributionController getInstance():
Retorna l'única instància del controlador, creant-la si encara no existeix.

3. **Gestió del conjunt de distribucions**
   - **public void setCjtD(CjtDistribuciones cjtD)**:
     Assigna un nou conjunt de distribucions al controlador.
   - **public Distribucion getDistribucion(String nombre)**:
     Retorna una distribució a partir del seu nom.
   - **public boolean exists(String nombre)**:
     Comprova si una distribució amb un nom específic existeix al conjunt.
4. **Operacions sobre distribucions**
   - **public boolean guardarDistribucion(Distribucion dist, String name)**:
     Afegeix una distribució nova al conjunt si el nom no està duplicat ni és buit.
   - **public boolean eliminarDistribucion(String nombre)**:
     Elimina una distribució a partir del seu nom, si existeix.
   - **public boolean cambiarNombreDist(String nombreAntiguo, String nombreNuevo)**:
     Canvia el nom d'una distribució si el nou nom no està ja en ús.
   - **public void editarDistribucion(String nombreDist, String prod1, String prod2)**:
     Modifica una distribució per canviar la relació entre dos productes.
   - **public void eliminarProducteDistribucion(String nombreDist, String[] prods)**:
     Elimina un o més productes d'una distribució específica.
   - **public Distribucion generarDistribucion(String algoritmo, int h, int w, Map<String, Map<String, Float>> LlistaGdS)**:
     Crea una nova distribució aplicant un dels algoritmes disponibles (`Voraz` o `Aproximació`).
5. **Utilitats internes**
   - **public Distribucion createDistribution(int h, int w)**:
     Crea una nova instància de **Distribucion** amb les dimensions indicades.
   - **public void executeAlgorithm(Distribucion distribution, String option, Map<String, Map<String, Float>> LlistaProductes)**:
     Executa un algorisme específic per omplir una distribució a partir d'una llista de productes i la configuració escollida.
   - **public Set<String> verDists()**:
     Retorna tots els noms de les distribucions existents al conjunt.

2.3.4 MensajeController

La classe **MensajeController** implementa un controlador per gestionar l'enviament de missatges dins del sistema. Utilitza el patró Singleton per garantir que només existeixi una

instància d'aquesta classe al llarg del programa. Aquest controlador s'encarrega de gestionar la creació i l'enviament de missatges entre usuaris.

Atributs

- **private static mensajeController instance**: Instància única de la classe. És necessària per implementar el patró Singleton i garantir que només hi hagi una única instància activa.

Mètodes

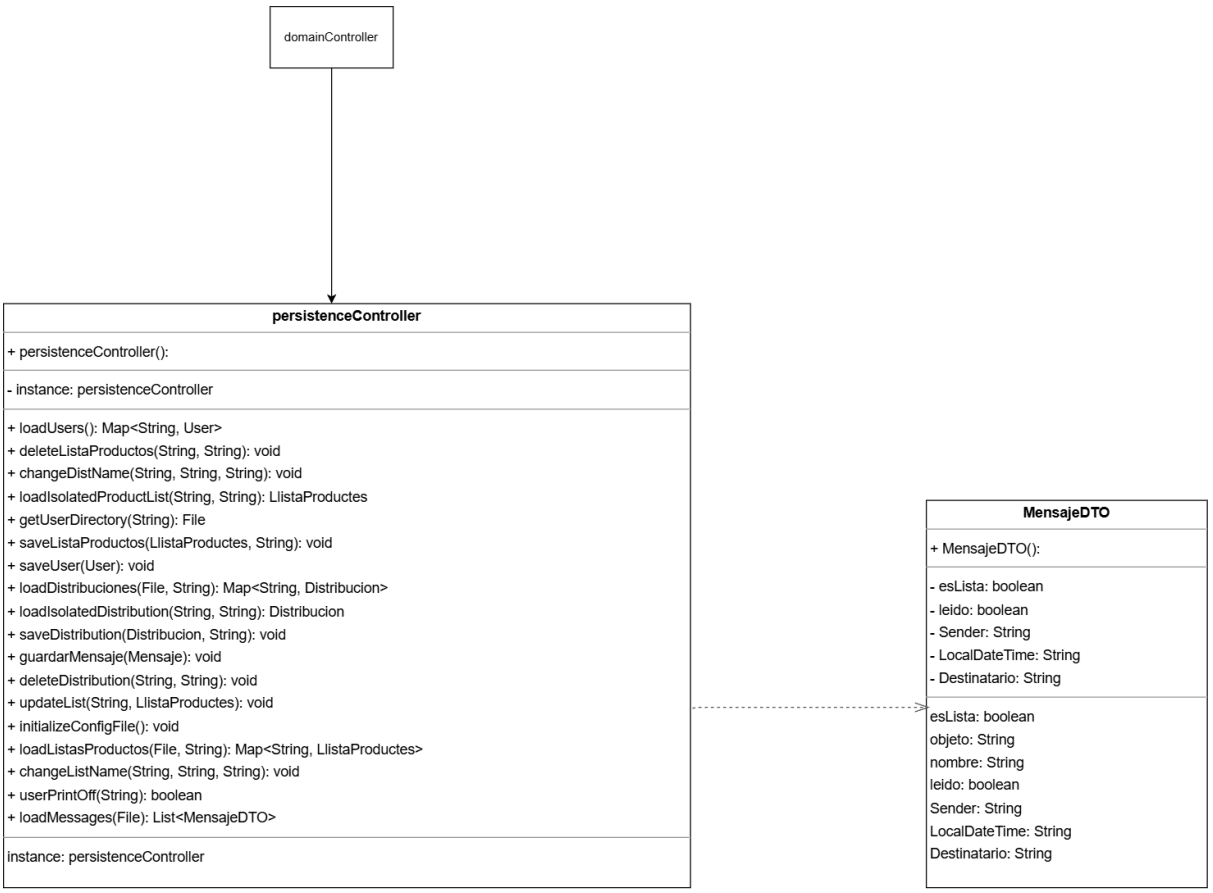
- **mensajeController()**:
  Constructor privat que inicialitza l'objecte. No inclou cap lògica específica.
- **static mensajeController getInstance()**:
  Mètode estàtic que retorna l'única instància de la classe. Si aquesta encara no existeix, la crea abans de retornar-la..
- **void EnviarMensaje(Mensaje mensaje)**:
  Mètode públic que s'encarrega d'enviar un missatge. Actualitza els atributs dels usuaris implicats, afegint el missatge a les llistes de missatges enviats del remitent (*Sender*) i de missatges rebuts del destinatari (*Destinatario*). Finalment, imprimeix un missatge de confirmació a la consola.

Comentari

Aquesta classe és senzilla però essencial per garantir que els missatges siguin processats i registrats correctament dins del sistema. L'ús del patró Singleton assegura que l'enviament de missatges es gestioni de manera centralitzada i coherent.

3. Diagrama de les classes i controladors de la capa de persistència

A continuació mostrem el disseny del diagrama de les classes i controladors de la capa de persistència. El diagrama es pot trobar detallat a la carpeta DOCS.



3.1. Explicació de les classes de persistència

MensajeDTO

S'encarrega de representar un missatge en format transferible, incloent informació com l'objecte del missatge, el remitent, el destinatari i altres atributs. Aquesta classe està pensada per ser utilitzada en la serialització i deserialització a formats com JSON.

Atributs:

- **object**: Representa l'objecte del missatge (p. ex., "Distribución" o altres categories similars).
- **name**: El nom associat al missatge.
- **esLista**: Boolean que indica si el missatge és una llista o no.
- **leido**: Boolean que indica si el missatge ja ha estat llegit.
- **LocalDateTime**: Data i hora del missatge, emmagatzemat com a *String* per preservar el format en JSON.
- **Sender**: Nom d'usuari del remitent del missatge.
- **Destinatario**: Nom d'usuari del destinatari del missatge.

Mètodes:

- **getObjeto()**: Retorna l'atribut *object* que indica l'objecte del missatge.
- **getNombre()**: Retorna l'atribut *name* que conté el nom associat al missatge.

- **isEsLista()**: Retorna *true* si el missatge és una llista, *false* altrament.
- **isLeido()**: Retorna *true* si el missatge ha estat llegit, *false* altrament.
- **getLocalDateTime()**: Retorna l'atribut *LocalDateTime* que conté la data i hora del missatge com a cadena de text.
- **getSender()**: Retorna el nom d'usuari del remitent (*Sender*).
- **getDestinatario()**: Retorna el nom d'usuari del destinatari (*Destinatario*).

3.2. Explicació del controlador de persistència persistenceController

df La classe *persistenceController* s'encarrega de gestionar la persistència de dades en un sistema d'organització que utilitza directoris i fitxers JSON per emmagatzemar informació d'usuaris, distribucions, llistes de productes i missatges.

Hem escollit la persistència amb fitxers JSON per al nostre projecte de programació perquè ofereix una solució senzilla, flexible i adequada per gestionar dades estructurades com les **llistes de productes** i les **distribucions** per a cada usuari.

Els JSON són fàcils de llegir i escriure tant per a màquines com per a humans, cosa que simplifica el desenvolupament i la depuració del sistema. A més, són compatibles amb la majoria de llenguatges de programació, fet que ens permet integrar-los fàcilment en la nostra aplicació.

Aquesta opció també és eficient per a projectes de mida petita o mitjana, on no es requereix la complexitat d'una base de dades relacional. Guardar les dades en fitxers JSON ens permet mantenir una estructura jeràrquica clara, útil per representar relacions entre productes i distribucions de manera natural. Finalment, aquesta elecció facilita la portabilitat i el desplegament, ja que només cal moure els fitxers per preservar la informació.

Atributs

- **instance**: Instància única de la classe per implementar el patró Singleton.
- **baseDirPath**: Ruta base on s'emmagatzema tota la informació del sistema.
- **configFilePath**: Ruta al fitxer de configuració que registra les dades dels usuaris.
- **userLog**: Mapa que associa noms d'usuari amb informació sobre el seu estat i directori personal.

Mètodes

Inicialització

- **persistenceController()**: Constructor que crea els directoris base i el fitxer de configuració si no existeixen.
- **getInstance()**: Retorna la instància única de la classe (patró Singleton).
- **initializeConfigFile()**: Crea el fitxer de configuració *config.txt* si no existeix.

Gestió d'Usuaris

- **saveUser(User user)**: Desa un nou usuari, crea el seu directori personal amb subcarpetes predefinides i registra la seva informació al fitxer de configuració.
- **loadUsers()**: Carrega tots els usuaris registrats des del fitxer de configuració i valida l'existència dels seus directoris.
- **getUserDirectory(String username)**: Retorna el directori associat a un usuari i el marca com a actiu en el registre *userLog*.
- **userPrintOff(String username)**: Comprova si un usuari està inactiu al sistema.

Gestió de Distribucions

- **loadDistribuciones(File distributionsDir, String username)**: Carrega totes les distribucions emmagatzemades en un directori específic d'un usuari.
- **loadIsolatedDistribution(String username, String distributionName)**: Carrega una distribució específica des d'un fitxer JSON.
- **saveDistribution(Distribucion distribucion, String username):** Desa una distribució en un fitxer JSON al directori corresponent de l'usuari.
- **deleteDistribution(String distributionName, String username)**: Elimina una distribució específica del sistema.
- **changeDistName(String oldName, String newName, String username)**: Canvia el nom d'una distribució assegurant-se que el nou nom no existeixi.

Gestió de Llistes de Productes

- **loadListasProductos(File listasDir, String username)**: Carrega totes les llistes de productes des d'un directori específic d'un usuari.
- **loadIsolatedProductList(String username, String nameList)**: Carrega una llista de productes específica des d'un fitxer JSON, incloent-hi la seva matriu de similituds.
- **saveListaProductos(LlistaProductes lista, String username)**: Desa una llista de productes com a fitxer JSON.
- **deleteListaProductos(String lista, String username)**: Elimina una llista de productes específica del sistema.
- **changeListName(String oldName, String newName, String username)**: Canvia el nom d'una llista de productes verificant que no existeixi una altra amb el mateix nom.
- **updateList(String username, LlistaProductes lista)**: Actualitza una llista existent afegint o eliminant productes i modificant els seus graus de similitud.

Gestió de Missatges

- **loadMessages(File MessagesDir)**: Carrega tots els missatges emmagatzemats en un directori d'entrada (inbox) i els converteix en objectes *MensajeDTO.*
- **guardarMensaje(Mensaje mensaje)**: Desa un missatge a l'inbox del destinatari i al directori de missatges enviats (*sentMessages*) del remitent.

Comentaris Addicionals

- Aquesta classe és fonamental per a la persistència de dades del sistema, utilitzant el format JSON per emmagatzemar estructures complexes.
- Utilitza la biblioteca *Gson* per a la serialització i deserialització de dades, simplificant les operacions amb JSON.

- Inclou la gestió d'errors amb missatges detallats per facilitar la depuració i millorar l'experiència de l'usuari.

4. Estructures de dades i algorismes

4.1 Estructures de dades

1. CjtDistribuciones:

- **Map (HashMap):**

  Propòsit: Emmagatzemar i gestionar un conjunt de distribucions.

  Estructura: L'atribut **conjuntDistribucions** es un **Map<String, Distribucion>,** on:

  - **Clau:** Representa el nom únic de cada distribució.
  - **Valor:** Es la instancia de la clase **Distribucion** associada al nom.

  Avantatges:

  1. **Recerca eficient:** Permet accedir a les distribucions en temps promig constant O(1) mitjançant el seu nom.
  2. **Gestió flexible:** Facilita l'addició, eliminació i verificació de l'existència de distribucions.
  3. **Accés a claus:** El mètode **getNames** utilitza **keySet()** per retornar un conjunt de noms de distribucions sense emmagatzemar duplicats.

2. CjtLlistesProductes:

- **Map (HashMap):**

  Propòsit: Emmagatzemar i gestionar un conjunt de llistes de productes.

  Estructura: L'atribut **conjuntoListas** és un **Map<String, LlistaProductes>**, on:

  - **Clau:** Representa el nom únic de cada llista de productes.
  - **Valor**: És la instància de la classe **LlistaProductes** associada al nom.

  Avantatges: Ídem que **CjtDistribuciones**.

3. CjtUsers:

- **List(ArrayList):**

  Propòsit: Retornar els noms d'usuari en forma de llista ordenada.

  Estructura: S'utilitza al mètode **getAllUsernames()**, que converteix les claus del **Map** en una **ArrayList<String>**

  Avantatges:

  1. **Accés seqüencial:** Ideal per retornar els noms d'usuari de manera ordenada i fàcilment iterables.

- **Map (HashMap):**

  Propòsit: Emmagatzemar i gestionar un conjunt d'usuaris.

  Estructura: L'atribut **usersList** és un **Map<String, User>**, on:

  - **Clau**: Representa el nom únic de cada usuari (username).
  - **Valor:** És la instància de la classe **User** associada a aquest nom.

Avantatges: Ídem que **CjtDistribuciones**.

4. Distribucion:

- **Pair**

  Propòsit: Emmagatzemar les dimensions de la distribució (alçada i longitud).
  Estructura: **Pair<Integer, Integer> Prestage**, on:
  - El primer element representa l'alçada.
  - El segon element representa la longitud.

  Ús: Es fa servir per definir i consultar les dimensions de la distribució.

- **List (ArrayList)**

  Propòsit: Gestionar la llista ordenada de productes que formen part de la distribució.
  Estructura: **List<String> Dist**, on cada element és el nom d'un producte.
  Avantatges:
  1. Permet l'accés seqüencial als noms dels productes.
  2. És fàcilment iterable i manipulable.

- **Map (HashMap)**

  Propòsit: Mantenir una associació entre els noms dels productes i les seves posicions dins la distribució.
  Estructura: **Map<String, Integer> mapa**, on:
  - La clau (**String**) és el nom del producte.
  - El valor (**Integer**) és la seva posició a la distribució.
    Ús:
  - Cerca ràpida de la posició d'un producte (O(1)).
  - Manipulació eficient durant operacions com intercanviar o eliminar productes.

- **Map anidat**

  Propòsit: Rebre les dades d'entrada per generar una distribució mitjançant una estratègia.
  Estructura: **Map<String, Map<String, Float>>** al mètode **generateDist.**
  - El primer nivell del mapa (**Map<String, Map<String, Float>>**) associa el nom d'un producte amb un altre **Map**.
  - El segon nivell del mapa (**Map<String, Float>**) enllaça altres productes amb un grau de similitud (Float).

  Ús: El mapa anidat s'utilitza al mètode **generateDist**, on s'envia com a entrada a l'estratègia definida. L'estratègia processa aquestes relacions per decidir l'ordre final dels productes.

5. LlistaProductes:

- **Map anidat**

  Propòsit: Gestionar una llista de productes amb una matriu de similituds entre ells.
  Estructura: **Map<String, Map<String, Float>> similarityMatrix,** on:
  - El primer nivell del mapa (**Map<String, Map<String, Float>>**) associa el nom d'un producte amb un altre mapa.

- El segon nivell del mapa (**Map<String, Float>**) conté els productes relacionats amb un grau de similitud (de tipus **Float**).

Ús: Aquest mapa s'utilitza per emmagatzemar la matriu de similituds entre productes. Quan es vol obtenir la similitud entre dos productes, es consulta aquest mapa. També permet afegir nous productes i eliminar-los juntament amb les seves relacions de similitud. A més, es poden actualitzar els valors de similitud entre productes mitjançant el mètode **setSimilarity**.

6. User:

- **List (ArrayList)**

Propòsit: Gestionar una llista de productes amb una matriu de similituds entre ells.
Estructura: **List<Mensaje> inbox, List<Mensaje> sentMessages**.
Els dos atributs són llistes (**List<Mensaje>**) que emmagatzemen els missatges rebuts (**inbox**) i els enviats (**sentMessages**).
Ús: Aquestes llistes s'utilitzen per gestionar els missatges del usuari, emmagatzemant els missatges rebuts i els enviats. El comptador de missatges nous (**mensajesNuevos**) s'incrementa cada vegada que s'afegeix un nou missatge a la safata d'entrada.

Algorismes

1.1.1. Algoritme voraç

L'AlgorismeVoraz és un algorisme que utilitza backtracking per trobar la millor distribució de productes en funció de la similitud entre ells. Busca maximitzar la suma de les similituds entre productes consecutius en una llista.

1) Atributs

a) **best_sim**: Emmagatzema la major similitud acumulada trobada.
b) **lordenada**: Llista amb la millor ordre de productes calculat.
c) **Matrix**: Matriu que emmagatzema les similituds entre els productes.
d) **visitats**: Llista per rastrejar els productes ja processats.

2) Mètodes

a) **backtracking**
Genera totes les combinacions possibles de productes per trobar la millor.
- Si es completa una llista, calcula la seva similitud acumulada.
- Si la similitud és millor que l'anterior, actualitza els resultats.
- Continua construint la llista fins processar tots els productes.

b) **calcular_sim**

Suma les similituds entre els productes consecutius en una llista parcial. Ajuda a avaluar si una combinació actual és millor que l'anterior.

c) **executeAlg**

Inicialitza les variables i executa l'algorisme per trobar la millor llista.

- Rep una matriu de similituds com a entrada.
- Torna l'ordre òptim de productes.

1.1.2. Algoritme d'aproximació

Aquest algoritme implementa l'algoritme d'aproximació de Kruskal amb Cerca Local Iterativa (ILS), utilitzat per trobar solucions aproximades a problemes de rutes amb cost mínim. El procés es desenvolupa en diverses etapes per generar un cicle Hamiltonià a partir d'un arbre d'expansió mínima (MST) i optimitzar-lo mitjançant tècniques com la cerca local i el redisseny d'arcs. A continuació es descriu cada part de l'algoritme:

1) Definició de les classes internes:

a) **Edge:** Representa una aresta entre dos nodes amb el seu pes associat (en aquest cas, la similitud entre productes).

b) **MFS (Union-Find):** Implementa l'estructura de conjunts disjunts per gestionar la unió i la cerca de conjunts de nodes durant l'algoritme de Kruskal, prevenint la formació de cicles en el MST.

2) Algoritme de Kruskal (obtenir MST):

a) S'extreuen totes les arestes possibles entre els nodes del graf, les quals es classifiquen per pes (similitud).

b) Utilitzant l'estructura MFS, s'uneixen els nodes, afegint només aquelles arestes que no causin un cicle, obtenint així un arbre d'expansió mínima (MST).

3) Cerca Local Iterativa (ILS):

a) L'algoritme utilitza l'algoritme 2-opt, que realitza millores locals en el MST invertint l'ordre de subllistes d'arestes per reduir el cost global.

b) S'introdueix una **perturbació aleatòria** en cada iteració per evitar quedar-se en òptims locals, millorant contínuament la solució en un màxim de 100 iteracions.

4) Generació del graf dirigit:

a) Es crea un graf dirigit a partir de l'MST optimitzat, amb les arestes que representen les connexions entre nodes de manera que cada node apunta als seus veïns.

5) Generació del Cicle Eulerià:

a) Mitjançant un recorregut DFS (cerca en profunditat), es construeix un cicle Eulerià a partir del graf dirigit, prioritzant les connexions més fortes (les que tenen més similitud entre els nodes).

6) Transformació en un Cicle Hamiltonià:

a) El cicle Eulerià es transforma en un cicle Hamiltonià eliminant nodos repetits, de manera que cada producte aparegui una sola vegada en el cicle final.

7) Execució de l'algoritme:

El mètode *executeAlg* integra tots els passos anteriors:

1. Calcula el MST amb Kruskal.
2. Optimitza el MST mitjançant ILS.
3. Genera el grafo dirigit i el cicle Eulerià.
4. Finalment, transforma el cicle Eulerià en un cicle Hamiltonià i el retorna com a solució.

5. Canvis entre entregues

En aquesta darrera entrega del projecte, hem introduït canvis significatius en l'arquitectura i funcionalitat dels controladors per adaptar-los a la incorporació d'una interfície visual. A continuació, detallem les modificacions principals:

1. **Simplificació dels controladors**
   Les funcions dels controladors s'han centrat exclusivament en la gestió de les estructures de dades i en els càlculs necessaris per al funcionament de l'aplicació. Això ha estat possible gràcies a la integració de la interfície visual, que elimina la necessitat d'introduir dades mitjançant el terminal.
2. **Eliminació de funcionalitats obsoletes**
   Totes les funcions destinades a gestionar l'entrada i sortida de dades per terminal han estat eliminades. Aquesta neteja no només redueix el codi redundant, sinó que també millora la llegibilitat i mantenibilitat del projecte, seguint els principis de *clean code*.
3. **Addició de noves funcionalitats per a la interfície visual**
   S'han desenvolupat noves funcions específiques per gestionar les interaccions amb la interfície gràfica. Això inclou elements com l'actualització dinàmica de dades en pantalla, la captura d'events dels usuaris i la integració fluida entre el backend i la interfície gràfica.

Raons per aquests canvis

- **Separació de responsabilitats:** La nova estructura facilita la implementació del principi SOLID, concretament el principi de responsabilitat única, ja que els controladors es dediquen exclusivament a la lògica de negoci.
- **Escalabilitat:** L'eliminació de codi lligat al terminal i la integració d'una interfície visual ens prepara per futures ampliacions o adaptacions a altres plataformes, com aplicacions mòbils o web.
- **Experiència d'usuari:** Una interfície visual és més intuïtiva i eficient per a l'usuari final, millorant la usabilitat del sistema.

- **Facilitació de proves:** La separació entre la lògica de negoci i la interfície visual simplifica les proves unitàries, ja que podem verificar el comportament dels controladors sense dependre de la interfície gràfica.

En conclusió, aquests canvis reflecteixen una evolució natural del projecte cap a un sistema més modular, mantenible i orientat a l'usuari. Això no només facilita el desenvolupament futur, sinó que també garanteix una experiència òptima per als usuaris finals.