Projecte de programació

Grup 5.3

Oriol Deiros Tort

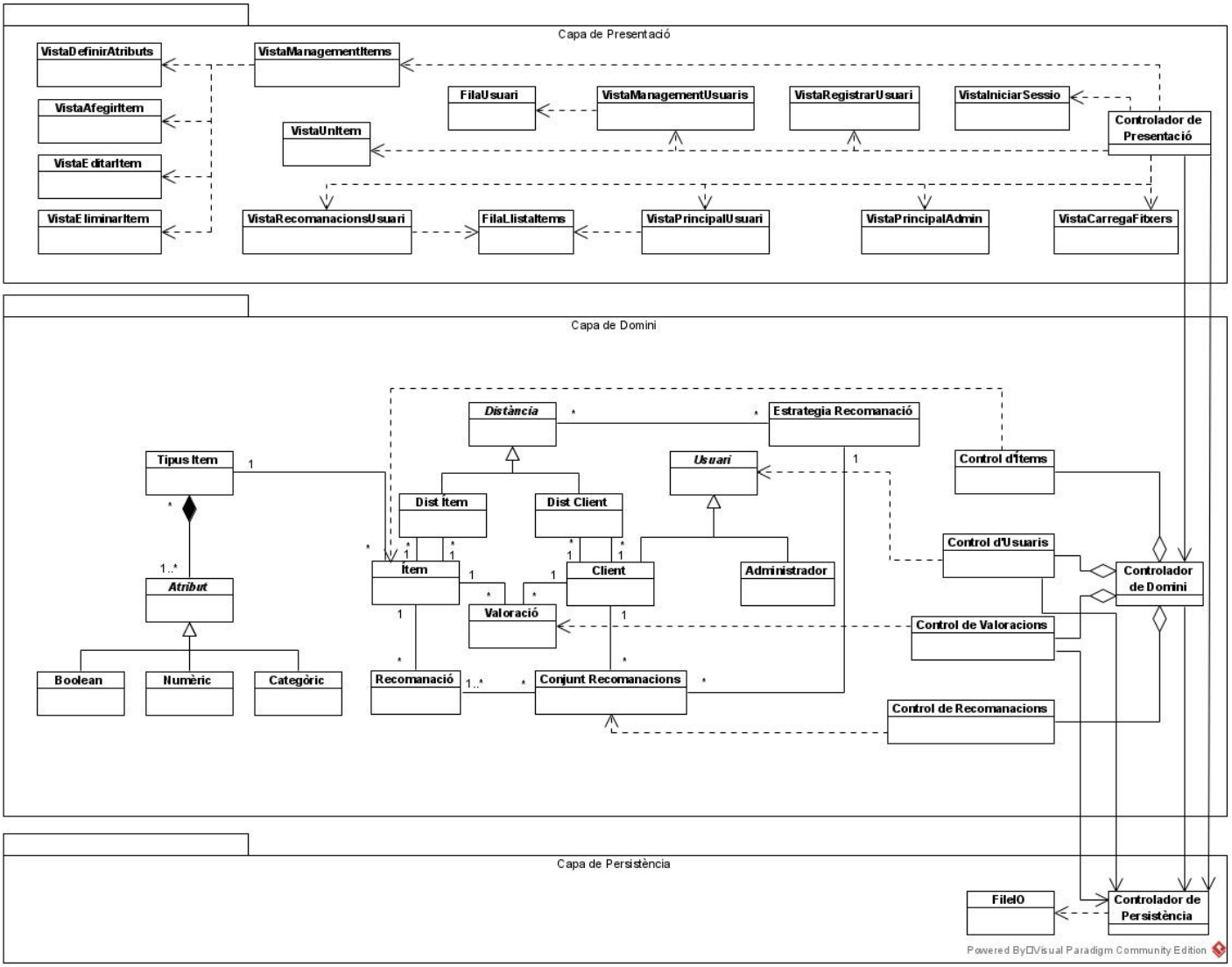
Pau Antonio Soler

Jiabo Wang

Hector Godoy Creus

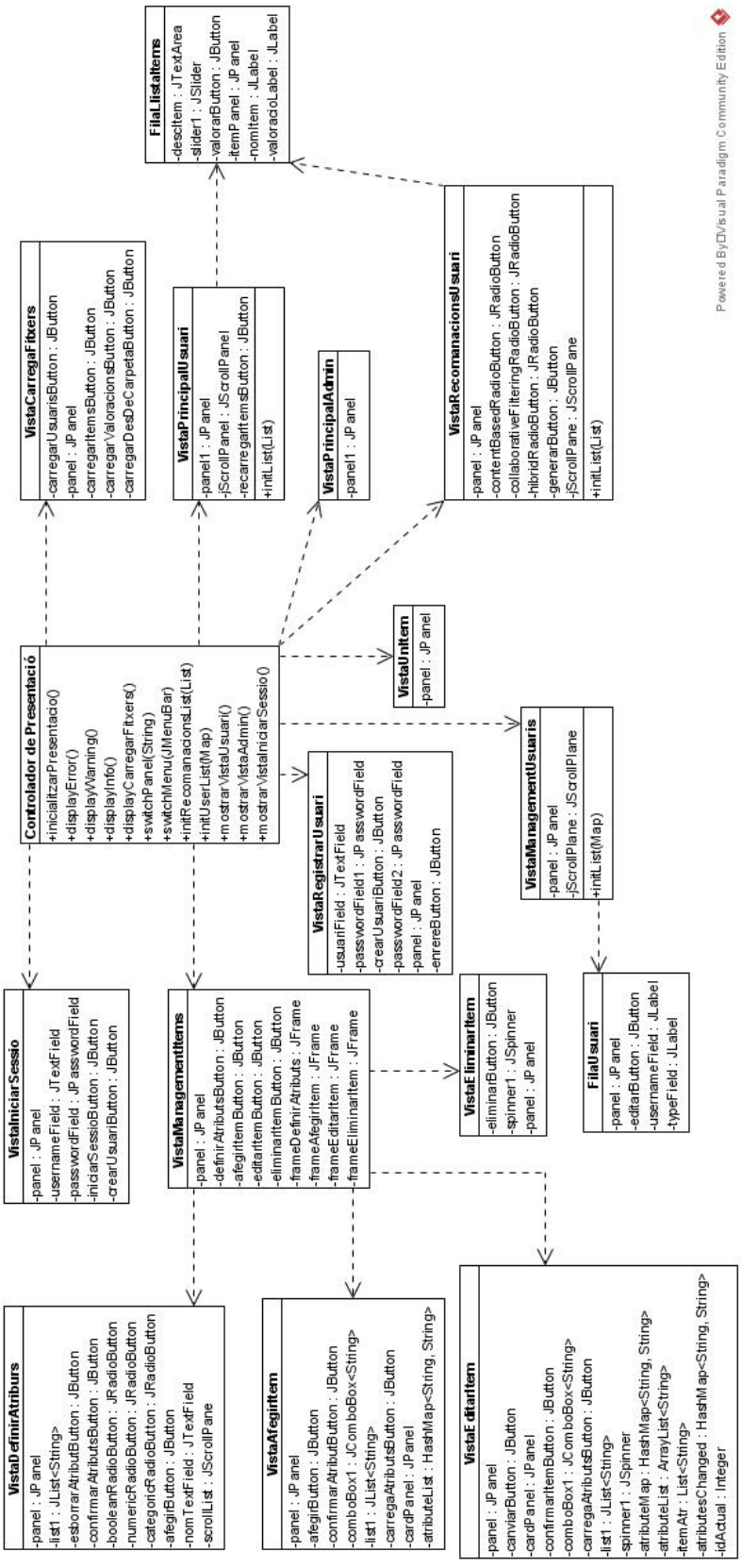
Índex

1. Disseny de l'arquitectura en tres capes

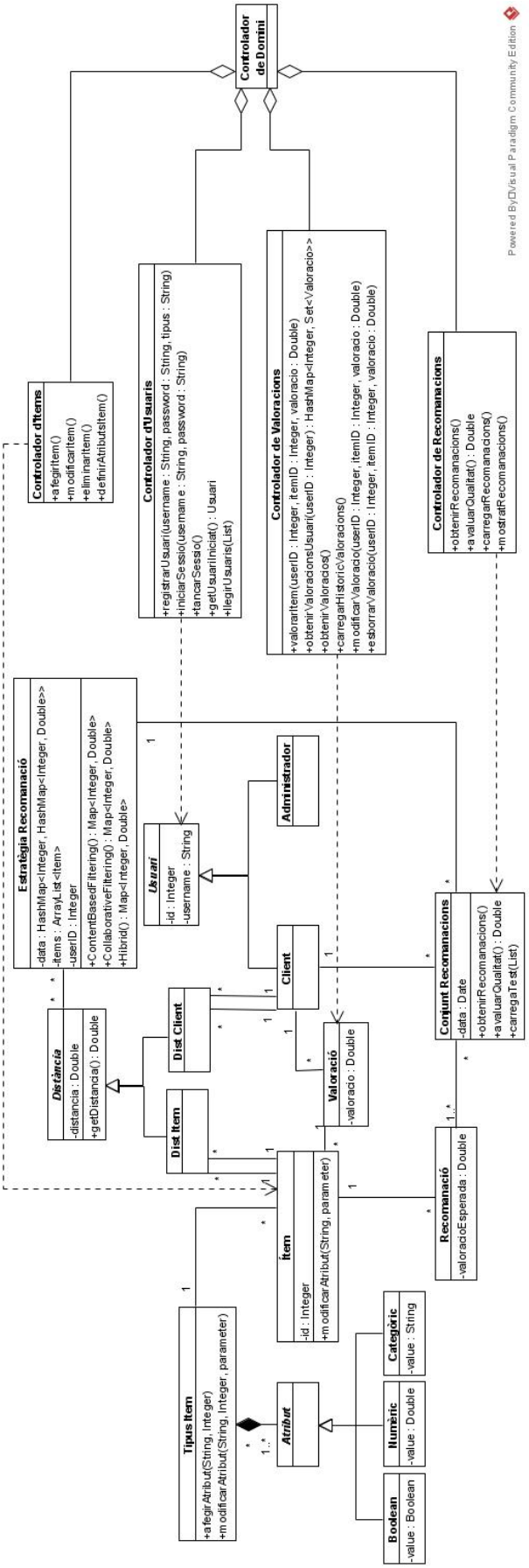


2. Diagrames UML estàtics

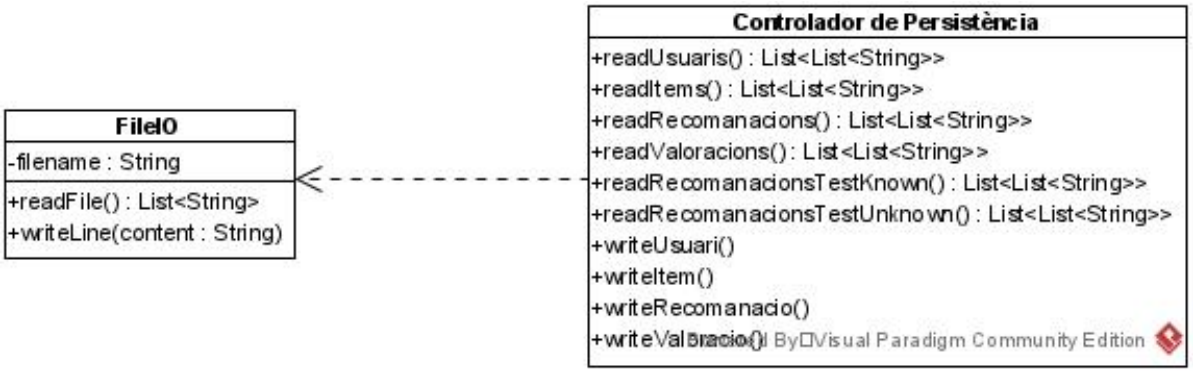
2.1 Capa de Presentació



2.2 Capa de Domini



2.3 Capa de Persistència



3. Descripció de les classes

3.1 Descripció breu de les classes pertanyents a la Capa de Domini

CtrlItems

Breu explicació: CtrlItems és la classe encarregada de gestionar totes les operacions relacionades amb el control dels items. És on s'hi emmagatzema el conjunt d'ítems, i té operacions per redefinir els seus atributs.

Cardinalitat: Només pot existir un sol objecte d'aquesta classe durant l'execució del programa.

Atributs

Tots els atributs són exclusius de la classe.

```
private static CtrlItems instance;
```

Atribut que conté la única instància de la classe que pot existir. Explicat amb més detall al mètode getInstance()

```
private static ArrayList<Item> cjtItems;
```

Vector encarregat d'emmagatzemar els Ítems.

```
private static ArrayList<String> nomAtributs;
```

Vector que conté amb el mateix ordre que han estat afegits, els noms dels atributs.

```
private static TipusItem plantillaTipus;
```

Aquest atribut és una plantilla de TipusItem. Quan es carreguin els noms dels atributs d'un arxiu, s'emplenarà amb aquests noms. Serveix per afegir els ítems més ràpidament, ja que es copia aquesta plantilla a l'ítem nou.

Mètodes:

```
public final ArrayList<String> getAtributeList()
```

Resultat: ArrayList<String> nomAtributs

El mètode retorna com a variable de tipus final el vector contenidor dels noms dels atributs del conjunt.

```
public HashMap<String, String> getAtributesByType()
```

Resultat: Diccionari que conté el nom dels atributs a les claus i el valor de cadascuna és el tipus d'atribut.

```
private int getAtributeType(String firstCheck, String secondCheck)
```

Paràmetres:

- firstCheck – Valor del primer atribut a comprovar
- secondCheck – Valor del segon atribut a comprovar

Resultat: 0, 1 o 2. Corresponent als atributs indicats prèviament: TIPUS_BOOLEAN; TIPUS_NUMERIC; TIPUS CATEGORIC.

Aquest mètode permet obtenir el tipus d'item el qual pertanyen les dues Strings passades com a paràmetres. S'hi passen dos Strings (corresponents a dos valors diferents del conjunt), per assegurar-se de que els valors llegits son correctes i evitar errors.

En cas que el conjunt d'Ítems sigui massa petit, es pot passar només el primer paràmetre i deixar el segon null.

```
private void definirAtributsItem(List<List<String>> conjunt)
```

Paràmetres:

- conjunt – Matriu rebuda del controlador de persistència (s'espera que la primera fila contingui els noms dels atributs, i les següents, el valor dels Items)

El mètode és l'encarregat d'inicialitzar l'estructura "plantillaTipus". Afegeix tots els atributs com si es tractés d'un Item, categoritzant-los en funció del seu tipus.

Un cop inicialitzada "plantillaTipus", servirà per crear nous items, partint del nom dels atribut que conté.

```
public void afegirItem(List<String> item)
```

Paràmetres:

- item – Vector d'Strings que conté els valors dels atributs del nou item. Aquest vector ha d'estar ordenat de la mateixa manera en la que s'ha llegit el vector de noms d'atributs.

Aquest mètode permet afegir un nou ítem al conjunt. Per fer-ho crea una nova instància d'Item i de TipusItem, i itera sobre el vector per escriure els atributs utilitzant el mètode modificarAtribut de TipusItem.

```
private Item getItem(int id)
```

Paràmetres:

- id – Identificador de l'Item

Resultat: Item

El mètode retorna un Item si aquest existeix en el conjunt

```
public void modificarItem(int id, String nomAtribut, String nouValor)
```

Paràmetres:

- id – Identificador de l'Item
- nomAtribut – Nom de l'atribut a modificar
- nouValor – Nou valor que ha de prendre l'atribut

Aquest mètode crida al mètode de l'item modificarAtribut amb els valors passats per paràmetres. L'item ha d'existir en el conjunt.

public void eliminarItem(int id)

Paràmetres:

- id – Identificador de l'Item

El mètode permet eliminar un item del conjunt. Si no hi és, llença una excepció.

public boolean existeixItem(int id)

Paràmetres:

- id – Identificador de l'item

Resultat: booleà

Mètode que comprova si un item existeix dins del conjunt.

public int mida()

Resultat: Mida del conjunt

Aquest mètode retorna la mida del conjunt d'items

public void carregarConjuntItems(List<List<String>> conjunt)

Paràmetres:

- conjunt – Matriu rebuda del controlador de persistència (s'espera que la primera fila contingui els noms dels atributs, i les següents, el valor dels Items)

El mètode és l'encarregat d'interpretar els valors llegits de persistència i convertir-los en estructures de dades d'Ítems. Per fer-ho, inicialitza el conjunt d'ítems buit, el vector 'nomAtributs', amb la primera línia de la matriu i, a continuació, itera per totes les altres línies de la matriu conjunt per afegir els valors dels atributs de cada Item.

public List<String> serialitzarItemPerId(int id)

Paràmetres:

- id – Identificador de l'Item

Resultat: Llista de strings amb els atributs de l'item, incloent l'identificador.

El mètode retorna els atributs de l'item identificat per l'id, que existeix en el conjunt. Els atributs es retornen en forma de llista i ordenats igual que la llista de noms dels atributs 'nomAtributs'.

```
private List<String> serialitzarItem(Item item)
```

Paràmetres:

- item – Item a serialitzar

Resultat: Llista de strings amb els atributs de l'item, incloent l'identificador.

El mètode retorna els atributs de l'item Els atributs es retornen en forma de llista i ordenats igual que la llista de noms dels atributs 'nomAtributs', variable que es troba en la classe CtrlItems.

```
public List<List<String>> serialitzarConjuntItems()
```

Resultat: Matriu de strings, la primera fila conté el nom dels atributs, i les següents el valor de cadascun dels atributs de cada item del conjunt.

Item

Breu explicació: Aquesta classe representa un item, conté tots els atributs que defineixen l'item.

Cardinalitat: Poden existir tantes instàncies com items s'hagin carregat des de la capa de persistència.

Atributs:

```
private int itemID;
```

Enter que conté l'identificador de l'item

```
private TipusItem tipus;
```

Instancia de la classe TipusItem exclusiva de l'item, conté tots els atributs d'aquest.

Mètodes:

```
public final int getID()
```

Resultat: Identificador de l'item

Es tracta d'un mètode que retorna l'identificador de l'item, emmagatzemat com a atribut de la instancia.

```
public final HashMap<String, Atribut> GetAtrB()

public final HashMap<String, ArrayList<AtribCategoric>> GetAtrC()

public final HashMap<String, Atribut> GetAtrN()
```

Resultat: Diccionari contenidor dels atributs de l'item.

Aquests mètodes invoquen els corresponents mètodes de la classe TipusItem per retornar com a variables final, els diccionaris de la classe TipusItem.

```
public void modificarAtribut(String nomAtribut, String valorNou)
```

Paràmetres:

- nomAtribut – Nom de l'atribut a modificar
- valorNou – Valor nou de l'atribut

Aquest mètode permet modificar el valor de l'atribut de l'item, per fer-ho invoca el mètode de la classe TipusItem encarregat de fer-ho.

final TipusItem getTipus()

Resultat: TipusItem de la instancia.

El mètode retorna el tipus de l'ítem com a final per a evitar ser modificat.

TipusItem

Breu explicació: La classe s'encarrega de gestionar tots els atributs propis d'un item, excepte l'ID. Els emmagatzema separant-los per tipus d'atribut. Cada Item té una sola instancia d'aquesta classe.

Cardinalitat: Existeix una instancia d'aquesta classe per cada instància d'Item que existeix dins del context d'execució del programa.

Atributs:

```
private HashMap<String, Atribut> atributs_numerics;

private HashMap<String, ArrayList<AtribCategoric>> atributs_categorics;

private HashMap<String, Atribut> atributs_booleans;
```

Els dos mètodes creadors de la classe inicialitzaran tots tres atributs, ja sigui buits o partint dels diccionaris d'una altre instancia de la classe TipusItem.

Aquests atributs son diccionaris que emmagatzemen els atributs d'un item. Segons el tipus d'atribut aniran a un diccionari concret. La seva clau és un String, que representa el nom de l'atribut en qüestió i el valor és una instància de l'Atribut concret.

Pel cas del diccionari atributs_numerics, els valors a emmagatzemar son de la classe AtribNumeric. Pel diccionari atributs_categorics, els valors que hi trobarem son vectors d'AtribCategoric, ja que es pot donar el cas que un mateix atribut categòric tingui més d'un valor. I pel diccionari atributs_booleans, els valors que hi trobarem son instancies de la classe AtribBoolea.

Mètodes:

```
public TipusItem()
```

Resultat: TipusItem

Classe constructora que inicialitza cada atribut amb un diccionari buit

```
public TipusItem(TipusItem ti)
```

Paràmetres:

- ti – Instancia TipusItem

Resultat: TipusItem

Classe constructora que inicialitza els atributs partint dels atributs d'una altre instancia de TipusItem.

final HashMap<String, Atribut> getAtrN()

final HashMap<String, ArrayList<AtribCategoric>> getAtrC()

final HashMap<String, Atribut> getAtrB()

Retornen un diccionari concret dels atributs de la classe. Son final, per a què no es puguin modificar.

void afegirAtribut(String nom, int tipus)

Paràmetres:

- nom – Nom de l'atribut a afegir
- tipus – Valor del tipus (Atribut.TIPUS_BOOLEA, Atribut.TIPUS_CATEGORIC, Atribut.TIPUS_NUMERIC)

Aquest mètode permet afegir un nou atribut a la instancia TipusItem sense inicialitzar el seu valor. Això ens serveix per definir els atributs quan comencem a llegir de persistència, quan només es coneix el nom dels atributs, i així crear una plantilla buida.

void modificarAtribut(String nom, int atrType, String val)

Paràmetres:

- nom – Nom de l'atribut
- atrType – Valor del tipus (Atribut.TIPUS_BOOLEA, Atribut.TIPUS_CATEGORIC, Atribut.TIPUS_NUMERIC)
- val – Nou valor de l'atribut

Aquest mètode permet modificar el valor d'un atribut ja present dins del seu diccionari corresponent. Si el paràmetre 'val' que conté el nou valor es buit, es modifica l'atribut i es marca com a no inicialitzat

int getTipus(String nomAtribut)

Resultat: Valor del tipus (Atribut.TIPUS_BOOLEA, Atribut.TIPUS_CATEGORIC, Atribut.TIPUS_NUMERIC)

Aquest mètode retorna un enter corresponent al tipus de l'atribut especificat pel nom del paràmetre. Si l'atribut no hi és al conjunt, es produeix una excepció.

Atribut

Breu explicació: Es tracta d'una classe abstracta dissenyada per encabir tots els mètodes dels atributs de diferent tipus, amb els seus corresponents valors.

Cardinalitat: No poden existir objectes d'aquesta classe degut a que es abstracta

Atributs:

```
public static final int TIPUS_BOOLEAN = 0;

public static final int TIPUS_NUMERIC = 1;

public static final int TIPUS_CATEGORIC = 2;
```

Serveixen per fer més atractiu visualment el tipus d'atribut

```
Boolean atrInitialized;
```

Valor per defecte: False. Indica si el valor que conté ja ha estat inicialitzat.

Mètodes:

```
public abstract int getTipus();
```

Resultat: 0, 1 o 2. Corresponent als atributs indicats prèviament: TIPUS_BOOLEAN; TIPUS_NUMERIC; TIPUS CATEGORIC.

El mètode retorna un valor diferent en funció de la classe filla en la qual el mètode s'ha executat.

Els altres mètodes s'expliquen en cada classe filla, ja que son abstractes.

AtribBoolea

Breu explicació: Classe filla de la classe abstracta Atribut. Serveix per emmagatzemar i gestionar un atribut del tipus Boolean.

Cardinalitat: Poden existir tants objectes d'aquest tipus com atributs d'aquest tipus tingui tots els ítems.

Atributs:

```
Boolean value;
```

Mètodes:

```
public AtribBoolea()
```

Resultat: AtribBoolea

Es tracta d'una constructora que marca com a no inicialitzat el seu valor

public AtribBoolea(Boolean val)

Paràmetres:

- val – valor que ha de prendre la variable | Resultat: AtribBoolea.

Es tracta d'una constructora que pren el valor de val i inicialitza l'objecte

public void setBoolValue(Boolean newVal)

Paràmetres:

- newVal – Valor que pren value. Aquest mètode modifica el valor de la variable value de la classe.

public Boolean getBoolValue()

Resultat: value.

Aquest mètode retorna el valor de la variable value.

AtribNumeric

Breu explicació: Classe filla de la classe abstracta Atribut. Serveix per emmagatzemar i gestionar un atribut del tipus Double.

Cardinalitat: Poden existir tants objectes d'aquest tipus com atributs d'aquest tipus tingui tots els items.

Atributs:

Double value;

Mètodes:

public AtribNumeric()

Resultat: AtribNumeric

Es tracta d'una constructora que marca com a no inicialitzat el seu valor

public AtribNumeric(Double val)

Paràmetres:

- val – valor que ha de prendre la variable | Resultat: AtribNumeric.

Es tracta d'una constructora que pren el valor de val i inicialitza l'objecte

public void setDoubleValue(Double newVal)

Paràmetres:

- newVal – Valor que pren value. Aquest mètode modifica el valor de la variable value de la classe.

public Boolean getDoubleValue()

Resultat: value.

Aquest mètode retorna el valor de la variable value.

AtribCategoric

Breu explicació: Classe filla de la classe abstracta Atribut. Serveix per emmagatzemar i gestionar un atribut del tipus String.

Cardinalitat: Poden existir tants objectes d'aquest tipus com atributs d'aquest tipus tingui tots els ítems.

Atributs:

String value;

Mètodes:

public AtribCategoric()

Resultat: AtribCategoric

Es tracta d'una constructora que marca com a no inicialitzat el seu valor

public AtribCategoric(String val)

Paràmetres: val – valor que ha de prendre la variable | Resultat: AtribCategoric.

Es tracta d'una constructora que pren el valor de val i inicialitza l'objecte

public void setStringValue(String newVal)

Paràmetres: newVal – Valor que pren value.

Aquest mètode modifica el valor de la variable value de la classe.

public Boolean getStringValue()

Resultat: value.

Aquest mètode retorna el valor de la variable value.

CtrlUsuari

Breu explicació: CtrlUsuari és la classe encarregada de gestionar totes les operacions relacionades amb el control dels usuaris. És on s'hi emmagatzema l'usuari inciat, i té operacions per iniciar i tancar sessió i registrar usuaris.

Cardinalitat: Només pot existir una instància d'aquesta classe

Atributs:

```
public static final String TYPE_CLIENT = "type_client";

public static final String TYPE_ADMIN = "type_admin";
```

Defineixen l'string en què es desa el tipus d'usuari.

```
private static final int ID_FIELD = 0;

private static final int USERTYPE_FIELD = 1;

private static final int USERNAME_FIELD = 2;

private static final int PASSWORD_FIELD = 3;
```

Corresponen a la posició on es troben els atributs dels usuaris en una fila del fitxer de dades.

```
private static CtrlUsuari instance;
```

Instància única de la classe

```
private Usuari usuariIniciat;
```

Instància de l'usuari que ha iniciat sessió. Per defecte, i quan no hi ha cap iniciat, el seu valor és null.

Mètodes:

```
private CtrlUsuari
```

Constructora privada, ja que només pot haver-hi una instància.

```
public static CtrlUsuari getInstance()
```

Resultat: Instància del controlador d'usuari

Aquest mètode retorna la instància única de CtrlUsuari, i si no existeix, la crea.

public void registrarUsuari(String username, String password, String tipus)

Paràmetres:

- username - Nom d'usuari de l'usuari a crear.
- password - Contrasenya de l'usuari a crear
- tipus - Tipus d'usuari a crear, entre CtrlUsuari.TYPE_CLIENT o CtrlUsuari.TYPE_ADMIN

Aquest mètode crea i escriu a persistència l'usuari indicat pels paràmetres. Si ja existeix, llança una UserException.

public void iniciarSessio(String username, String password)

Paràmetres:

- username - Nom d'usuari de l'usuari.
- password - Contrasenya de l'usuari

Aquest mètode canvia el valor d'usuariIniciat per l'usuari indicat. Si les credencials són incorrectes, l'usuari no existeix o ja hi ha una sessió iniciada, es llança una UserException.

public void tancarSessio()

Aquest mètode elimina la variable d'usuariIniciat i la posa a null. Si no hi ha cap usuari iniciar, es llança una UserException.

public Usuari getUsuariIniciat()

Resultat: Usuari que ha iniciat sessió

Aquest mètode retorna la instància de l'usuari que ha iniciat la sessió. Si no hi ha cap sessió iniciada, es llança una UserException.

Usuari

Breu explicació: Es tracta d'una classe abstracta que conté els mètodes i atributs generals per tots els usuaris.

Cardinalitat: és una classe abstracta, i per tant no pot tenir instàncies directes.

Atributs:

```
private final Integer id;
```

Enter que representa l'id de l'usuari, i que l'identifica en totes les operacions.

```
private final String username;
```

Nom d'usuari de l'usuari, que també ha de ser únic.

Mètodes:

```
Usuari(int id, String username)
```

Constructora, de visibilitat package, que només pot ser cridada per la controladora. L'única instància d'usuari a la que poden accedir les altres classes son la d'usuari iniciat.

```
public Integer getId()
```

Resultat: Id

Aquest mètode retorna l'identificador de l'usuari.

```
public String getUsername()
```

Resultat: username

Aquest mètode retorna el nom d'usuari de l'usuari.

```
public abstract String getTipus()
```

Mètode abstracte implementat a les classes filles.

Administrador

Breu explicació: Classe que hereta de la classe Usuari, que representa un usuari amb permisos d'administració.

Cardinalitat: Poden haver-hi tantes instàncies com administradors a la base de dades.

Mètodes:

public String getTipus()

Resultat: CtrlUsuari.TYPE_ADMIN

Aquest mètode retorna la string corresponent al tipus d'usuari.

Client

Breu explicació: Classe que hereda de la classe Usuari, que representa un usuari de tipus client.

Cardinalitat: Poden haver-hi tantes instàncies com clients a la base de dades.

Mètodes:

public String getTipus()

Resultat: CtrlUsuari.TYPE_CLIENT

Aquest mètode retorna la string corresponent al tipus d'usuari.

Distancia

Breu explicació: Es tracta d'una classe abstracta creada per tal de calcular la distància entre dos objectes, que poden ser entre ítems o entre clients. La distancia es guarda en un Double amb 2 decimals de precisió.

Cardinalitat: No poden existir objectes d'aquesta classe degut a que es abstracta.

Mètodes:

```
public abstract Double()
```

Mètode abstracta que retorna la distancia entre dos objectes

DistClient

Breu explicació: Es tracta d'una classe filla de la classe abstracta Distancia. És l'encarregat de calcular la distancia entre 2 clients fent servir les seves valoracions.

Cardinalitat: Poden existir tant objectes d'aquest tipus com distàncies volem calcular entre dos clients diferents.

Atributs:

```
private HashMap<Integer, Double> f1;
```

Diccionari que conté el conjunt de valoracions del client 1, l'enter representa l'identificador d'un ítem i Double representa la valoració que s'ha fet sobre aquell ítem.

```
private HashMap<Integer, Double> f2;
```

Diccionari que conté el conjunt de valoracions del client 2, l'enter representa l'identificador d'un ítem i Double representa la valoració que s'ha fet sobre aquell ítem.

```
private Double distancia;
```

Distància entre els dos clients, el valor per defecte es 0.0

Mètodes:

```
public DistClient(HashMap<Integer, Double> fz1, HashMap<Integer, Double> fz2)
```

Paràmetres:

- fz1: conjunt de valoracions d'un client 1
- fz2: conjunt de valoracions d'un client 2

Resultat: DistClient

Descripció: es tracta d'una constructora que pren els conjunts de valoracions de dos clients

```
public Double getDistancia()
```

Resultat: Double, representa la distància entre els dos clients

Descripció: es tracta d'una funció que calcula la distancia euclidiana entre els 2 conjunts de valoracions. es tracta d'una funció que calcula la distancia euclidiana entre els 2 conjunts de valoracions. És a dir, per cada valoració d'un ítem del client 1, si el client 2 també l'ha valorat, es fa la potencia de dos de la resta de les seves valoracions, si no, només es fa la potencia de 2 de la valoració del client 1. Un cop calculades totes les potències de 2 de les diferències, es fa

l'arrel quadrada d'aquest nombre, que representa la distància que existeix entre aquests dos clients.

DistItem

Breu explicació: Es tracta d'una classe filla de la classe abstracta Distancia. És l'encarregat de calcular la distancia entre 2 items fent servir els seus atributs.

Cardinalitat: Poden existir tant objectes d'aquest tipus com distàncies volem calcular entre dos items diferents.

Atributs:

```
private final Item item1;
private final Item item2;
```

Corresponen al dos ítems que els volem calcular la distancia

```
private Double distancia;
```

Distància entre els dos ítems, el valor per defecte es 0.0

```
private Double atributs;
```

Enter que correspon al nombre total d'atributs entre els dos items

Mètodes:

```
public DistItem(Item i2, Item i2)
```

Paràmetres:

- i1: ítem 1
- i2: ítem 2

Resultat: DistItem
Descripció: es tracta d'una constructora que pren com a paràmetres dos Items

```
private Double d_categorics()
```

Resultat: Double, que representa el nombre d'atributs categòrics que tenen en comú els dos ítems.
Descripció: es tracta d'una funció que tracta els atributs categòrics dels dos ítems i calcula el nombre d'atributs que coincideixen, li dona un major pes a l'atribut "genre" que representa el gènere d'una pel·lícula o sèrie i no es té en compte si l'atribut a tractar es "homepage", "img_url", "link", "poster_path" i "imbd_id", ja que considerem que aquests atributs no tenen rellevància per calcular el grau de similitud entre dos ítems.

```
private Double d_numerics()
```

Resultat: Double, que representa el nombre d'atributs numèrics que tenen en comú els dos ítems.
Descripció: es tracta d'una funció que tracta els atributs numèrics dels dos ítems i calcula el nombre d'atributs que coincideixen.

```
private Double d_booleans()
```

Resultat: Double, que representa el nombre d'atributs booleans que tenen en comú els dos ítems.

Descripció: es tracta d'una funció que tracta els atributs booleans dels dos ítems i calcula el nombre d'atributs que coincideixen.

public Double getDistancia()

Resultat: Double, representa la distància entre els dos items

Descripció: es tracta d'una funció que calcula la distancia entre dos ítems, suma el nombre d'atributs numèrics, booleans i categòrics que coincideixen i el divideix pel nombre total d'atributs, que dóna un número entre 0 i 1 i representa el grau de semblança entre els dos ítems. Per tant, la distancia es 1 – aquest nombre. 0 representa que són dos ítems que tenen els atributs completament iguals i 1 que no tenen cap atribut en comú.

EstrategiaRecomanacio

Breu explicació: Es tracta d'una classe creada per tal de calcular els diferents algorismes recomanadors, com són el Collaborative Filtering, el Content Based Filtering i l'algorisme híbrid entre aquests dos.

Cardinalitat: Un conjunt de recomanacions es calculada només per un objecte EstrategiaRecomanacio, mentre que un mateix EstrategiaRecomanacio pot calcular més d'un conjunt de recomanacions (en concret 3, Collaborative, Content Based i l'híbrid).

Atributs:

```
private HashMap<Integer, HashMap<Integer, Double>> data;
```

Data emmagatzema les valoracions fetes per tots els clients fins al moment, és a dir, el primer enter representa l'identificador d'un client i el HashMap<Integer, Double>, representa el conjunt de valoracions fetes pel client, on la clau representa l'identificador de d'un ítem i el Double, la valoració sobre aquest ítem.

```
private ArrayList<Item> items_totals;
```

Vector de ítems que representa el conjunt de tots els ítems registrats al sistema.

```
private Integer clientID;
```

clientID, representa l'identificador del client al qui li volem fer el conjunt de recomanacions.

Tots tres atributs no poden ser estàtics, ja que els atributs poden variar, no sempre farem les recomanacions al mateix client, com tampoc tindrem sempre els mateixos ítems ni el mateix nombre de valoracions.

Mètodes:

```
public EstrategiaRecomanacio(HashMap<Integer, Set<Valoracio>> c, ArrayList<Item> i, Integer x);
```

Paràmetres:

- c: HashMap<Integer, Set<Valoracio>>
- i: ArrayList<Item>
- x: Integer

Resultat: EstrategiaRecomanador

Descripció: es tracta d'una constructora que pren com a paràmetres el conjunt de valoracions de tots els clients del sistema, el vector de tots els ítems del sistema i un enter x que representa l'identificador del client al qui li volem fer recomanacions.

```
private static Map<Integer, Double> sortByComparator(Map<Integer, Double> unsortMap, final boolean order)
```

Paràmetres:

- unsortMap: Map<Integer, Double>, map que es vol ordenar segons el valor
- order: boolean, true si es vol ordenar de menor a major i false de major a menor

Resultat: Map<Integer, Double>

Descripció: Mètode privat que donat un map, retorna el map ordenat segons el seu valor.

```
private Map<Integer, Double> kNN(ArrayList<Item> C, HashMap<Integer, Double> val)
```

Paràmetres:

- C: ArrayList<Item>, conjunt d'ítems agradats pel client
- val: HashMap<Integer, Double>, valoracions del client

Resultat: Map<Integer, Double>

Descripció: Mètode privat que donat el conjunt d'ítems que li agraden al client i les valoracions fetes pel client, retorna el conjunt Map<Integer, Double> dels k ítems amb la seva valoració esperada que se li vol recomanar al client, trobats aplicant l'algorisme kNN.

```
public Map<Integer, Double> ContentBasedFiltering();
```

Paràmetres: cap

Resultat: Map<Integer, Double>, representa el conjunt d'ítems amb la seva valoració esperada.

Descripció: Mètode públic que filtre el conjunt d'ítems que li agraden al client (els de valoració major o igual que 0.7 * rating màxim), crida a la funció kNN i retorna el conjunt Map<Integer, Double> dels k ítems amb la seva valoració esperada que se li vol recomanar al client.

```
private List<HashMap<Integer, Double>> CentroidesRandom(List<Integer> clients, int k)
```

Paràmetres:

- clients : List<Integer>, llista de clients totals
- k: Integer

Resultat: List<HashMap<Integer, Double>>, que representa la llista de k centroides

Descripció: Mètode privat que genera k centroides de manera aleatòria. Aquests centroides no tenen perquè coincidir amb les valoracions d'un client en concret (és a dir, no són punts reals), sinó que són punts generats dins un domini definit, que es el rang entre la valoració màxima possible i la mínima possible.

```
private HashMap<Integer, Double> CentroideMesProper (Integer client, List<HashMap<Integer, Double>> centroids)
```

Paràmetres:

- client: Integer, identificador d'un client
- centroids: List<HashMap<Integer, Double>>, conjunt de k centroides

Resultat: HashMap<Integer, Double>, que representa el centroide més proper al client que tenim com a paràmetre d'entrada.

Descripció: Mètode privat que donat l'identificador d'un client qualsevol, trobem al centroide que es troba a menor distància.

private HashMap<HashMap<Integer, Double>, List<Integer>> assignarCluster(HashMap<HashMap<Integer, Double>, List<Integer>> clusters, Integer client, HashMap<Integer, Double> centroid)

Paràmetres:

- clusters: HashMap<HashMap<Integer, Double>, List<Integer>>, representa el conjunt de k clústers.
- client: Integer, identificador d'un client.
- centroid: HashMap<Integer, Double>, representa el centroide d'un clúster.

Resultat: HashMap<HashMap<Integer, Double>, List<Integer>>, representa el clúster actualitzat, és a dir, al clúster que té centroide el centroide del paràmetre d'entrada se li afegeix el client de la entrada i s'elimina el client del centroide que es trobava.

Descripció: Mètode privat que assigna al client d'entrada al clúster que forma part el centroide d'entrada, és a dir, del conjunt de clústers busquem el que pertany el centroide d'entrada, i a la llista de clients d'aquest clúster, afegim el client d'entrada i l'eliminem del clúster que hi estava.

private HashMap<Integer, Double> mitjana(HashMap<Integer, Double> centroid, List<Integer> clients)

Paràmetres:

- centroid: HashMap<Integer, Double>, representa el centroide d'un clúster.
- clients: List<Integer>, representa la llista de clients que forma part del clúster que té com a centroide centroid.

Resultat: HashMap<Integer, Double>, que representa el nou centroide que té com a coordenades la mitjana de les coordenades de tots els clients de la llista.
Descripció: Mètode privat que calcula la mitjana de tots els punts de la llista de clients, si es buida, només retorna el centroide actual (que es centroid).

private List<HashMap<Integer, Double>> recolocacioCentroides (HashMap <HashMap <Integer, Double>, List<Integer>> clusters)

Paràmetres:

- clusters: HashMap<HashMap<Integer, Double>, List<Integer>>, representa el conjunt de k clústers.

Resultat: List<HashMap<Integer, Double>>, que representa la llista dels nous centroides.

Descripció: Mètode privat que recalcula tots els centroides dels k clústers, fent crides al mètode privat mitjana.

private HashMap<HashMap<Integer, Double>, List<Integer>> kmeans(List<Integer> clients, int k)

Paràmetres:

- clients: List<Integer>, representa el conjunt de clients que aplicarem l'algorisme kmeans per tal de classificar-los en diferents clústers.
- k: Integer, representa els k clústers que es formarà.

Resultat: HashMap<HashMap<Integer, Double>, List<Integer>>, que representa el conjunt de k clústers, és a dir, es classifica tots els clients del conjunt en un d'aquests k clústers.

Descripció: Mètode privat que aplica l'algorisme de kmeans, primer genera k centroides aleatoris, a continuació, per cada client, troba el centroide més proper, i se li assigna al clúster format per aquell centroide. Tot seguit, recalcula tots els centroides dels k clústers, de manera que siguin la mitjana de tots els clients que formen part d'aquell clúster. Després de recalcular els centroides, es torna a buscar el centroide més proper per cada client i moure el client de clúster i així successivament fins que en alguna iteració, tots els clients s'estabilitzen, és a dir, ja no canvien de clúster.

private List<Integer> find_group(HashMap<HashMap<Integer, Double>, List<Integer>> kmean)

Paràmetres:

- kmean: HashMap<HashMap<Integer, Double>, List<Integer>>, representa els k clústers calculats amb el mètode kmeans.

Resultat: List<Integer>, representa el grup de clients on pertany el nostre client.

Descripció: Mètode privat que troba al grup (clúster) que pertany el nostre client dins el conjunt de k grups calculats amb el mètode kmeans.

private HashMap<Integer, Double> slopeOne(HashMap<Integer, HashMap<Integer, Double>> data, List<Integer> group)

Paràmetres:

- HashMap<Integer, HashMap<Integer, Double>>: data, representa les valoracions fetes pels clients que pertanyen a la llista group.
- List<Integer>: group, representa el grup de clients on pertany el nostre client després d'aplicar l'algorisme kmeans.

Resultat: HashMap<Integer, Double>, representa el conjunt de prediccions (recomanacions que se li vol fer al nostre client).

Descripció: Mètode privat que aplica l'algorisme slopeOne al grup de clients que on pertany el nostre client, que en teoria, son els que més s'assemblen al nostre client. Per aplicar SlopeOne, s'ha de construir dues matrius, la del promig de la diferencia en les valoracions entre 2 items i la de la freqüència dels clients que han valorat tots 2 items. Basant-se en aquestes 2 matrius, la predicció de la valoració d'un ítem per al nostre client, es el sumatori de les valoracions de tots els ítems que ha valorat el nostre client + el promig de la diferencia entre aquests ítems amb l'ítem que es vol predir * la freqüència dels usuaris que han valorat tots dos ítems i finalment es divideix aquest sumatori amb el sumatori de totes les freqüències entre l'ítem que volem predir amb els altres ítems.

```
public Map<Integer, Double> CollaborativeFiltering();
```

Paràmetres: cap

Resultat: Map<Integer, Double>, representa el conjunt d'ítems amb la seva valoració esperada.

Descripció: Mètode públic que inicialment aplica kmeans per tal de dividir el conjunt de clients totals en k clústers. Tot seguit, troba el grup (clúster) on pertany el nostre client i aplica SlopeOne sobre aquest grup de clients (que en teoria son els que es s'assemblen més al nostre client, basant-se en les valoracions sobre ítems fetes). Finalment, del conjunt de prediccions fetes per l'algorisme SlopeOne, es queda amb el top 10 dels ítems que han obtingut una valoració esperada més alta. I aquests 10 ítems seran els que li recomanarem al nostre client.

```
public Map<Integer, Double> Hibrid();
```

Paràmetres: cap

Resultat: Map<Integer, Double>, representa el conjunt d'ítems amb la seva valoració esperada.

Descripció: Mètode públic que executa els algorismes Collaborative Filtering i Content Based Filtering per obtenir les dues llistes de recomanacions calculades amb cadascun dels algorismes, llavors, el mètode híbrid fa una unió d'aquestes dues llistes i retorna una llista amb els 10 ítems que tenen la valoració esperada més alta. I aquests 10 ítems seran els que li recomanarem al nostre client. Com que mitjançant una sèrie d'experiments hem pogut observar que en la majoria de casos les valoracions esperades dels ítems recomanats pel Collaborative Filtering eren molt més alts que els del Content Based Filtering, hem decidit rebaixar les valoracions esperades dels ítems recomanats pel Collaborative Filtering, és a dir, multiplicar les valoracions esperades per un factor de 0.65 (aquest valor l'hem trobat mitjançant la experimentació). Aplicant aquest factor, el mètode híbrid agafa recomanacions fetes tant per l'algorisme Content Based Filtering com pel Collaborative Filtering.

Valoracio

Breu explicació: Aquesta classe representa una valoració, conté tots els atributs que defineixen la valoració.

Cardinalitat: Poden existir tantes instàncies com valoracions hagin realitzat els usuaris.

Atributs

Tots els atributs són exclusius de la classe.

```
private final Integer itemID;
```

Enter que conté l'identificador de l'item valorat.

```
private final Integer itemID;
```

Enter que conté l'identificador de l'usuari valorador.

```
private final Doble valoracio;
```

Double que conté el número amb què s'ha valorat l'item.

Mètodes:

```
public Valoracio(Integer userID, Integer itemID, Double rating)
```

Paràmetres:

- userID – identificador de l'usuari que realitza la valoració
- itemID – identificador de l'item valorat
- rating – valoració que ha obtingut l'item per part de l'usuari

Resultat: Valoracio

Classe creadora que inicialitza cada atribut amb els atributs passats per paràmetre.

```
public Integer getItemID()
```

Resultat: Valor de l'identificador de l'item valorat.

Aquest mètode retorna l'identificador de l'item valorat.

```
public Integer getUserID()
```

Resultat: Valor de l'identificador de l'usuari valorador.

Aquest mètode retorna l'identificador de l'usuari valorador.

public Double getValoracio()

Resultat: Valor amb que l'usuari ha valorat l'item.

Aquest mètode retorna el número amb que s'ha valorat l'item.

public void setValoracio(Double novaValoracio)

Paràmetres:

- novaValoracio - valor amb que l'usuari vol canviar la valoració de l'ítem

Aquest mètode modifica el número amb que s'ha valorat l'item.

public void imprimirValoracio()

Aquest mètode imprimeix els atributs de la valoracio.

CtrlValoracio

Breu explicació: CtrlValoracio és la classe encarregada de gestionar totes les operacions relacionades amb el control de les valoracions. És on s'hi emmagatzema el conjunt de valoracions realitzades, i té operacions per valorar i obtenir valoracions.

Cardinalitat: Només pot existir un sol objecte d'aquesta classe durant l'execució del programa.

Atributs

Tots els atributs són exclusius de la classe.

```
private static CtrlValoracio instance;
```

Atribut que conté la única instància de la classe que pot existir. Explicat amb més detall al mètode getInstance()

```
private final HashMap<Integer, Set<Valoracio>> valoracions;
```

Taula hash encarregada d'emmagatzemar les valoracions realitzades. L'enter representa l'identificador de l'usuari, i el Set totes les valoracions que ha realitzat.

Mètodes:

```
private CtrlValoracio()
```

Resultat: Creació d'un CtrlValoracio

Aquest mètode permet crear una instància del controlador de valoracions.

```
public static CtrlValoracio getInstance()
```

Resultat: Instància del controlador de valoracions.

Aquest mètode permet obtenir la instància del controlador de valoracions, per tal de poder-ne executar les operacions. Substitueix a la creadora, ja que així controla que només existeixi una única instància del controlador.

```
public void valorarItem(Integer userID, Integer itemID, Double valoracio)
```

Paràmetres:

- userID – identificador de l'usuari que realitza la valoració
- itemID – identificador de l'item valorat
- valoracio – valoració que ha obtingut l'item per part de l'usuari

Aquest mètode permet crear una nova valoració amb els atributs passat per paràmetre.

public Set<Valoracio> getValoracionsUsuari(Integer userID)

Paràmetres:

- userID – identificador de l'usuari que realitza la valoració

Resultat: Conjunt de valoracions realitzades per un usuari.

Aquest mètode permet obtenir totes les valoracions realitzades per l'usuari amb identificador userID. Si no n'ha realitzat cap, llença una excepció.

public HashMap<Integer, Set<Valoracio>> getValoracions()

Resultat: Conjunt de valoracions realitzades per tots els usuaris.

Aquest mètode permet obtenir totes les valoracions realitzades pels usuaris.

private void guardarValoracio(Valoracio v)

Paràmetres:

- v – instància Valoracio

Resultat: Conjunt de valoracions realitzades per un usuari.

Aquest mètode communica a la capa de persistència la creació d'una nova Valoracio, que ha de ser emmagatzemada.

public void carregarValoracions(List<List<String>> historicValoracions)

Paràmetres:

- historicValoracions – llista dels atributs de les valoracions emmagatzemades a la capa de dades

Aquest mètode permet carregar al sistema les valoracions emmagatzemades a la capa de persistència.

public void imprimirValoracionsUsuari(Integer userID)

Paràmetres:

- userID – identificador de l'usuari del qual es volen imprimir les recomanacions

Aquest mètode imprimeix totes les valoracions del sistema associades a l'usuari passat per paràmetre.

public void modificarValoracio(Integer userID, Integer itemID, Double valoracio)

Paràmetres:

- userID – identificador de l'usuari que realitza la valoració
- itemID – identificador de l'item valorat
- valoracio – nova valoració que ha obtingut l'item per part de l'usuari

Aquest mètode permet modificar la valoració de l'ítem indicat per part de l'usuari, donant-li el valor passat per paràmetre.

public void modificarValoracio(Integer userID, Integer itemID)

Paràmetres:

- userID – identificador de l'usuari que realitza la valoració
- itemID – identificador de l'item valorat

Aquest mètode permet eliminar la valoració de l'ítem indicat per part de l'usuari.

Recomanacio

Breu explicació: Aquesta classe representa una recomanacio, conté tots els atributs que defineixen la recomanacio.

Cardinalitat: Poden existir tantes instàncies com recomanacions creat el sistema recomanador.

Atributs

Tots els atributs són exclusius de la classe.

```
private final Integer itemID;
```

Enter que conté l'identificador de l'item recomanat.

```
private final Doble valoracio_esperada;
```

Double que conté el número amb què s'espera que un usuari valori l'item recomanat.

Mètodes:

```
public Recomanacio(Integer itemID, Double val)
```

Paràmetres:

- itemID – identificador de l'ítem recomanat
- val – valoració amb que s'espera que es valori l'ítem

Resultat: Recomanacio

Classe creadora que inicialitza cada atribut amb els atributs passats per paràmetre.

```
public Integer getItemID()
```

Resultat: Valor de l'identificador de l'item recomanat.

Aquest mètode retorna l'identificador de l'item recomanat.

```
public Double getValoracio()
```

Resultat: Valor s'espera que l'usuari valori l'item.

Aquest mètode retorna el número amb que s'espera que l'usuari valori l'ítem.

```
public void imprimirRecomanacio()
```

Aquest mètode imprimeix els atributs de la recomanació.

ConjuntRecomanacions

Breu explicació: Aquesta classe representa una recomanacio, conté tots els atributs que defineixen la recomanacio.

Cardinalitat: Poden existir tantes instàncies com vegades s'hagi executat el sistema recomanador.

Atributs

Tots els atributs són exclusius de la classe.

```
private final Integer userID;
```

Enter que conté l'usuari a qui es fan les recomanacions.

```
private final Date data;
```

Data en que s'han creat les recomanacions.

```
private final String estrategia;
```

String que conté la estratègia amb que s'han obtingut o s'obtindran les recomanacions.

```
private final ArrayList<Recomanacio> CjtRecomanacions;
```

Llista que conté totes les recomanacions obtingudes a través del sistema recomanador.

```
private final Map<Integer, ArrayList<Double>> test;
```

Mapa que conté les recomanacions de test, per a poder evaluar la qualitat de les recomanacions obtingudes.

Mètodes:

```
public ConjuntRecomanacions(Integer userID, Date data, String estrategia)
```

Paràmetres:

- userID – identificador de l'usuari a qui es recomana
- data – data de creació de les recomanacions
- estrategia – indica quin dels sistemes recomanador s'ha d'utilitzar per a obtenir les recomanacions

Resultat: ConjuntRecomanacions

Classe creadora que inicialitza cada atribut amb els atributs passats per paràmetre.

public void obtenirRecomanacions()

Aquest mètode omple l'atribut CjtRecomanacions amb totes les recomanacions que retorna el sistema recomanador, segons la estrategia escollida. Per tal d'obtenir-les, primer ha d'obtenir totes les valoracions i items del sistema, per a que es puguin executar els algorismes correctament.

public Double avaluarQualitat()

Resultat: Double del valor del guany acumulat amb descompte (DCG - Discounted Cumulative Gain)

Aquest mètode retorna el valor del DCG, que permet avaluar la qualitat de les recomanacions obtingudes per part del sistema. Aquest valor s'obté a partir d'executar un algorisme que utilitza les recomanacions obtingudes i les recomanacions de test, comparar-les. Com més gran és aquest valor, major és la qualitat de les recomanacions.

public void carregaTest(List<List<String>> stringTest)

Paràmetres:

- List<List<String>> stringTest – llista les recomanacions de test emmagatzemades a la capa de dades

Aquest mètode permet carregar al sistema les recomanacions de test emmagatzemades a la capa de persistència. S'emmagatzemaran a l'atribut test.

public Integer getUserID()

Resultat: Valor de l'identificador de l'usuari.

Aquest mètode retorna l'identificador de l'usuari a qui es recomana.

public Date getData()

Resultat: Data en que es van crear les recomanacions.

Aquest mètode retorna la data de creació de les recomanacions.

public String getEstrategia()

Resultat: Estrategia usada per a obtenir les recomanacions

Aquest mètode retorna l'estrategia usada per tal d'obtenir les recomanacions.

public void imprimirRecomanacions()

Aquest mètode imprimeix totes les recomanacions obtingudes.

CtrlRecomanacio

Breu explicació: CtrlRecomanacio és la classe encarregada de gestionar totes les operacions relacionades amb el control de les recomanacions. És on s'hi emmagatzemen tots els conjunt de recomanacions realitzades, i té operacions per obtenir-ne, avaluar-les i imprimir-les.

Cardinalitat: Només pot existir un sol objecte d'aquesta classe durant l'execució del programa.

Atributs

Tots els atributs són exclusius de la classe.

```
private static CtrlRecomanacio instance;
```

Atribut que conté la única instància de la classe que pot existir. Explicat amb més detall al mètode getInstance()

```
private final HashMap<Integer, ConjuntRecomanacions> llistaConjuntRecomanacions;
```

Taula hash encarregada d'emmagatzemar els conjunts de recomanacions obtinguts. L'enter representa l'identificador de l'usuari a qui es recomanen, i el ConjuntRecomanacions, l'últim conjunt de recomanacions que s'ha obtingut per part del sistema.

Mètodes:

```
private CtrlRecomanacio()
```

Resultat: Creació d'un CtrlRecomanacio

Aquest mètode permet crear una instància del controlador de recomanacions.

```
public static CtrlRecomanacio getInstance()
```

Resultat: Instància del controlador de recomanacions.

Aquest mètode permet obtenir la instància del controlador de recomanacioons, per tal de poder-ne executar les operacions. Substitueix a la creadora, ja que així controla que només existeixi una única instància del controlador.

```
public void obtenirRecomanacions(String estrategia, Integer userID)
```

Paràmetres:

- estrategia – indica quin dels sistemes recomanador s'ha d'utilitzar per a obtenir les recomanacions
- userID – identificador de l'usuari per a que s'han d'obtenir les recomanacions

Aquest mètode permet obtenir un nou conjunt de recomanacions per a l'usuari indicat, amb l'algorisme indicat al paràmetre estrategia. En cas que l'usuari ja tingui un conjunt de recomanacions, aquest es substitueix pel nou.

public void imprimirRecomanacions(Integer userID)

Paràmetres:

- userID – identificador de l'usuari

Aquest mètode permet imprimir les el conjunt de recomanacions de l'usuari indicat. En cas que no en tingui, es llença una excepció.

public Double avaluarQualitat(Integer userID, List<List<String>> test)

Paràmetres:

- userID – identificador de l'usuari
- test – llista de les recomanacions de test emmagatzemades a la capa de dades

Resultat: Double del valor del guany acumulat amb descompte (DCG - Discounted Cumulative Gain) del conjunt de recomanacions de l'usuari.

Aquest mètode retorna el valor del DCG, que permet avaluar la qualitat de les recomanacions obtingudes per part del sistema per a l'usuari indicat. Aquest valor s'obté a partir d'executar un algorisme que utilitza les recomanacions obtingudes i les recomanacions de test, comparar-les. Com més gran és aquest valor, major és la qualitat de les recomanacions. En cas que l'usuari no tingui cap conjunt de recomanacions, es llença una excepció.

3.2 Descripció breu de les classes pertanyents a la Capa de Persistència

FileIO

FileIO és la classe encarregada de realitzar les operacions de lectura/escriptura en els fitxers de persistència de dades. Ens permet llegir les dades i escriure-les mitjançant matrius d'strings, convertint alhora de format CSV al format esperat i viceversa.

CtrlPersistència

CtrlPersistencia és la controladora que conté els mètodes necessaris per escriure el contingut de les estructures de dades als fitxers de persistència. Comunica les altres capes amb els mètodes destinats a l'emmagatzematge de dades.

CsvUtils

CsvUtils és la classe que conté mètodes estàtics destinats a la traducció i interpretació de text en format CSV.

3.3 Descripció breu de les classes pertanyents a la Capa de Presentació

SistemaRecomanadorPROP

Aquesta classe és l'encarregada de contenir el main() que inicialitzarà tot el sistema, cridant el mètode CtrlPresentacio.inicialitzarPresentacio(), que també inicialitza tots els controladors de la capa de domini i de la capa de persistència.

CtrlPresentacio

Aquesta classe té diverses funcions, la més important, inicialitzar tots els controladors i vistes que contindran el JFrame principal del programa. També és l'encarregada d'inicialitzar aquest JFrame.

La classe serveix de pont per totes les vistes i les seves classes associades, per poder comunicar-se amb la capa de domini, i els seus respectius controladors.

VistaManagementItems

La vista que conté aquesta classe només es troba disponible per a usuaris amb permisos d'administrador, s'hi pot accedir des del menú principal on es troben totes les recomanacions, a la barra de menú superior.

Hi podem trobar quatre opcions per obrir vistes relacionades amb la gestió dels ítems: definir atributs, afegir ítems, modificar ítems i eliminar ítems. Cal destacar que les tres darreres opcions només estan disponibles un cop s'han definit els atributs per primer cop.

VistaDefinirAtributs

Aquesta classe permet inicialitzar el conjunt, definint els atributs que contindrà cada ítem segons el seu tipus. Un cop es confirmen els atributs, si abans existia un conjunt s'esborra i se substitueix per un conjunt buit preparat per contenir els nous atributs.

VistaAfegirItem

Com el seu nom indica, permet afegir ítems al conjunt. L'assignació d'identificadors és de forma incremental, començant pel 0. Permet afegir ítems amb atributs sense inicialitzar, que reben el nom de "NO DEFINIT".

VistaEditarItem

La classe permet editar atributs dels ítems que han estat afegits al conjunt prèviament. L'únic paràmetre que no es pot modificar és l'identificador de l'ítem.

VistaEliminarItem

Aquesta classe permet eliminar un ítem del conjunt. Només cal indicar l'identificador.

VistaUnItem

Aquesta classe permet visualitzar l'identificador de l'item i els seus atributs.

VistaIniciarSessio

La classe permet que un usuari inicïi sessió, indicant-ne el username i la contrasenya. En cas que sigui incorrecte, mostra un missatge.

VistaRegistrarUsuari

La classe permet que un usuari es registri al sistema, indicant-ne el username i la contrasenya.

VistaManagementUsuaris

La classe genera la vista perquè els usuaris administradors puguin administrar els usuaris del sistema.

VistaCarregaFitxers

Classe que dona les diferents opcions per a carregar els diferents arxius de dades al sistema. Dona opció a carregar els fitxers de manera individual o de carregar-los tots des d'una carpeta.

FilaLlistaItems

Classe que genera la vista d'una fila corresponent a un ítem a valorar a la pantalla principal.

VistaPrincipalUsuari

Classe que genera la vista principal de la sessió d'un client.

VistaPrincipalAdmin

Classe que genera la vista principal de la sessió d'un administrador.

VistaRecomanacionsUsuari

Classe que genera la vista de les recomanacions que obté cada usuari.

4. Estructures de dades i algorismes

Per a emmagatzemar el conjunt d'ítems hem decidit utilitzar un vector, ja que en la majoria de casos, l'accés a aquest conjunt serà seqüencial. També per a la gestió de cadascun dels atributs d'un ítem, hem decidit utilitzar tres diccionaris, cadascun corresponent a un tipus d'atribut. Això s'ha decidit així per a fer l'accés més fàcil quan s'executen diversos algorismes.

L'estructura de dades utilitzada per a guardar les valoracions de tots els clients és un HashMap<Integer, HashMap<Integer, Double>>, on el primer Integer representa l'identificador d'un client, i el Hashmap<Integer, Double> de tot seguit representa el conjunt de parells Integer, Double que representa els ítems que ha valorat aquell client (l'Integer d'aquest segon hashmap representa l'identificador d'un ítem i el Double representa la valoració).

Algorisme Content Based Filtering: El primer pas de l'algoritme és decidir els ítems que li agraden al nostre client, per tant, dintre de les seves valoracions, ens quedarem amb el conjunt d'ítems que ha valorat amb una puntuació igual o major que 70% de la puntuació màxima (per exemple, en el cas que les valoracions siguin de 0 a 10, el 70% de la puntuació màxima és 7, llavors considerem que els ítems valorats amb un valor major o igual que 7 seran els considerats com a agradats per part de l'usuari). El següent pas és aplicar l'algoritme kNN per cadascun d'aquests ítems per decidir quins són els k ítems que es troben a menor distància i, per tant, tenen més probabilitat de ser acceptats pel client. Evidentment, es descarta els ítems que el client ja ha valorat. Finalment, les valoracions esperades d'aquests ítems es calcula multiplicant la distància per la valoració de l'ítem agradat, és a dir, si l'usuari ha valorat un ítem amb un 7 sobre 10, considerem que li agrada, ja que està dintre del 70%, i llavors, es calcula les distàncies entre aquest ítem que li ha agradat a l'usuari i tots els altres ítems del sistema, tot seguit ens quedem amb el top 10 dels ítems que s'assemblen més a aquest ítem agradat (els que tenen la distància més petita) i per calcular la valoració esperada, primer es resta 1 - distància i finalment es multiplica per 7 (la valoració que ha fet l'usuari a l'ítem agradat).

Collaborative Filtering: Aquest algorisme comença aplicant l'algorisme de kmeans, que classifica els diferents clients en grups (clústers) diferents segons les valoracions que sobre ítems que ha fet. L'algorisme consisteix a seleccionar k centroides inicials de manera aleatòria, escollint punts a l'atzar dins el rang de valoracions màxima i mínima (no tenen per què coincidir amb valoracions dels clients), i a continuació per cada client, calcular el centroide que es troba més proper a ell i assignar-li al clúster d'aquell centroide, un cop assignats tots els clients a tots els clústers, es recalcula el centroide fent la mitjana de les valoracions dels clients que formen part d'aquell clúster. Un cop recalculats els centroides, es tornen a calcular per cada client el centroide més proper i es tornen a col·locar al clúster del centroide més proper i així successivament. L'algoritme acaba quan s'estabilitza la recol·locació dels clients als clústers, és a dir, si tots els clients en fer una recol·locació es tornen a col·locar al mateix clúster. Un cop tenim calculats els diferents grups, hem de trobar el grup on pertany el nostre client, i sobre aquest grup de clients aplicar l'algoritme de Slope One, que intenta predir la valoració sobre un ítem mitjançant les valoracions que han fet altres usuaris sobre aquest ítem. Per aplicar l'algoritme, es calculen matrius de diferència i freqüència per tal de calcular el valor mitjà de les diferències entre les valoracions de 2 ítems. Per tal de fer les prediccions, s'utilitza els ítems ja valorats per l'usuari amb els altres ítems. Es fa el sumatori de la valoració de l'ítem ja valorat més el promig de la diferència en les valoracions d'aquest ítem i els altres, multiplicat per la

freqüència dels usuaris que han valorat tots els ítems. Finalment, d'entre totes les prediccions de valoracions, es fa un filtratge del top 10 de millors valoracions i es recomana al client.

Hibrid: Aquest algorisme és un híbrid dels dos algorismes anteriors, és a dir, primer obté el conjunt de recomanacions aplicant l'algorisme Content Based Filtering, tot seguit obté el conjunt de recomanacions aplicant l'algorisme Collaborative Filtering, un cop obtinguts els dos conjunts, fa una unió dels dos conjunts i es queda amb els 10 ítems amb la valoració esperada més alta. Nosaltres inicialment el vam plantejar així, però tot seguit mitjançant diversos experiments, vam poder comprovar que sempre escull les recomanacions fetes per l'algorisme Collaborative Filtering, ja que les valoracions esperades són bastant més alts que els del Content Based Filtering i, per tant, per equilibrar-ho, hem multiplicat les valoracions esperades dels ítems recomanats pel Collaborative Filtering per un factor de 0.65, de tal manera que les valoracions esperades calculades pels tots dos algorismes no tinguin molta diferència. I així l'algorisme híbrid agafa recomanacions fetes tant pel Collaborative Filtering com pel Content Based Filtering.