# PROJECTE DE PROGRAMACIÓ

## *- Sistemes Recomanadors -*

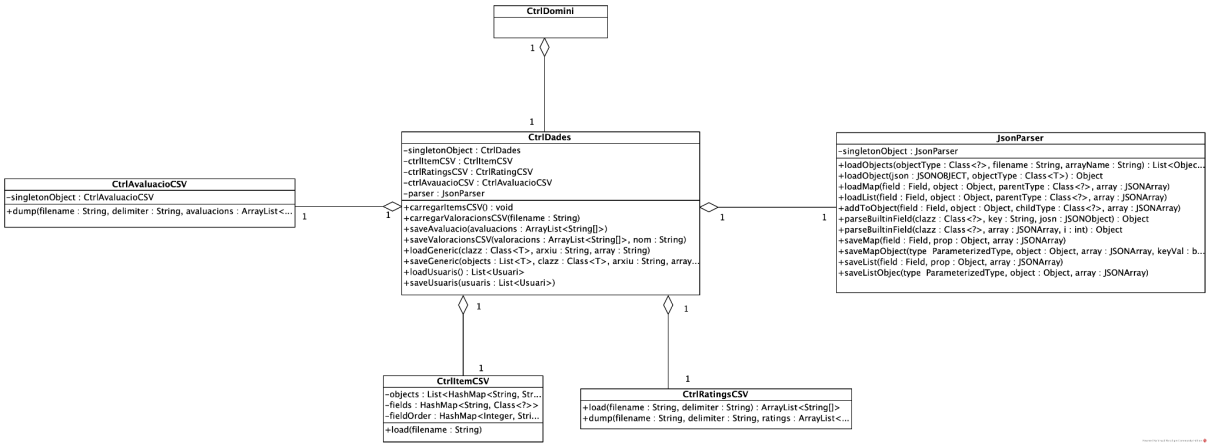
*Pau Dorca - pau.dorca*
*Jia Long Ji - jia.long.ji*
*Roberto Navarro - roberto.navarro*
*You Wu - you.wu*

**Grup 11.4: 2a Entrega - Versió 1.0**

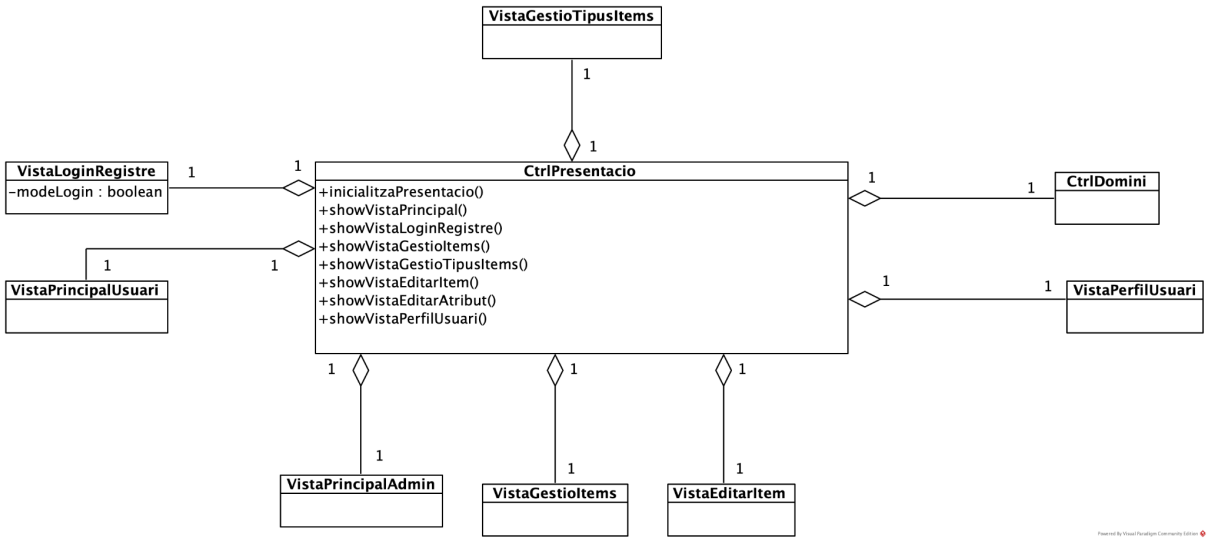
# ÍNDEX

# 1. Model Conceptual

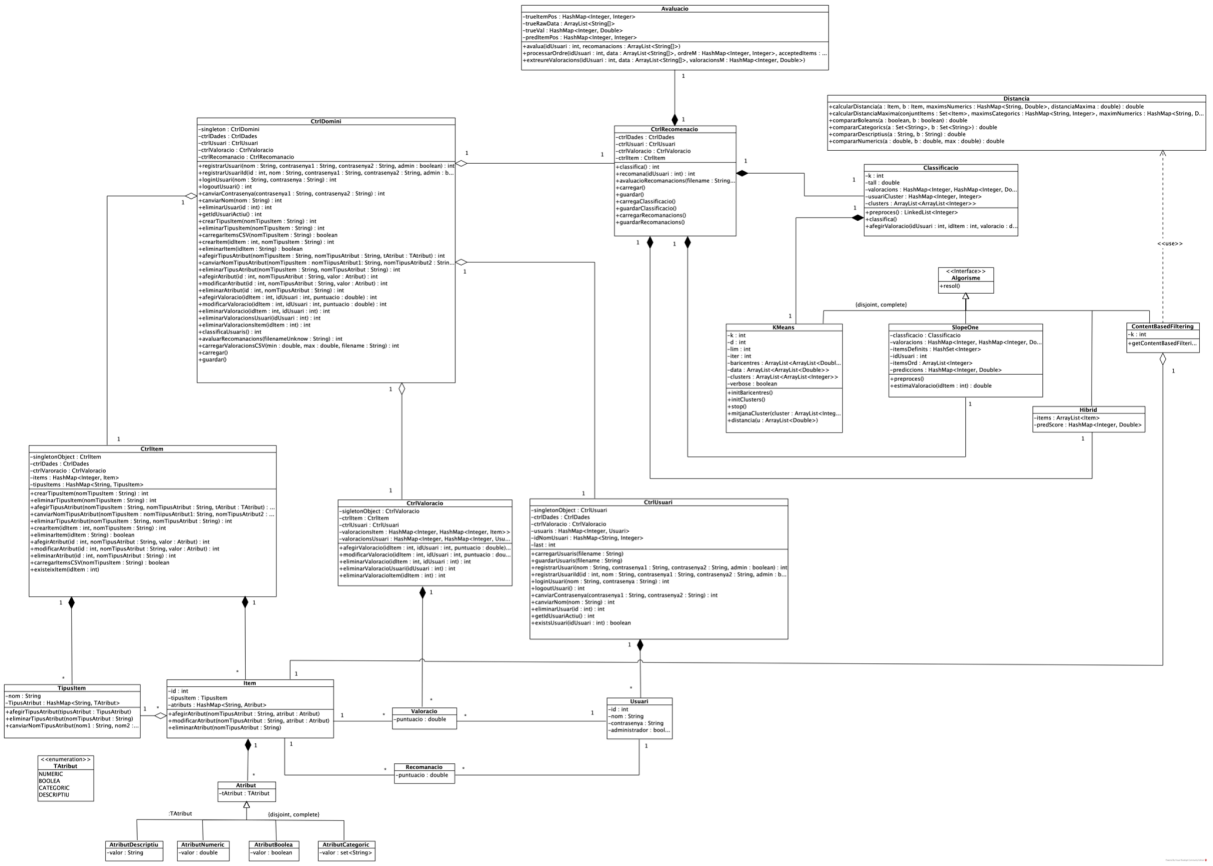
## 1.1 Capa de dades




## 1.2 Capa de presentació



# 1.3 Capa de domini




Per a veure els diagrames en més detall veure la carpeta docs.

# 2. Descripció de les classes

## 1. Classe Usuari

**Descripció:** Representa l'estructura d'un Usuari, tant l'id com el nom identifiquen un usuari.
**Atributs**:

- **id:** identificador de l'usuari.
- **nom:** nom de l'usuari.
- **contrasenya:** contrasenya de l'usuari.
- **admin:** indica si l'usuari és administrador (true) o no (false).

**Mètodes**:

- Funcionalitats bàsiques (*getters i setters*)

## 2. Classe Item

**Descripció**: Representa un l'estructura d'un item.
**Atributs**:

- **id:** identificador de l'ítem.
- **tipusItem:** el tipus d'ítem.
- **atributs:** atributs de l'ítem (compoxt per nomTipusAtributs i atributs).

**Mètodes**:

- Funcionalitats bàsiques (*getters i setters*)
- *afegirAtribut(String nomAtribut, Atribut atribut)*: afegeix un atribut amb el nom indicat.
- *modificarAtribut(String nomAtribut, Atribut atribut)*: modifica el valor de l'atribut amb el nom indicat.
- *eliminarAtribut(String nomAtribut*): elimina l'atribut amb el nom indicat.

## 3. Classe Atribut

**Descripció**: Representa un atribut d'un ítem. És una superclase abstracta que pot ser implementada per les quatre subclasses AtributNumeric, AtributBoolea, AtributCategoric i AtributDescriptiu.
**Atributs**:

- **tAtribut:** tipus d'atribut que pertany l'atribut, pot ser NUMERIC, BOOLEA, CATEGÒRIC, DESCRIPTIU.

**Mètodes**:

- Funcionalitats bàsiques(*getters*).

## 4. Classe AtributDescriptiu

**Descripció**: Representa un atribut descriptiu.
**Atributs:**

- **valor**: valor descriptiu (String) de l'atribut.

**Mètodes:**

- Funcionalitats bàsiques (implementació dels getters de la superclasse).

## 5. Classe AtributNumeric

**Descripció**: Representa un atribut numèric.

**Atributs:**

- **valor**: valor numèric (double) de l'atribut.

**Mètodes:**

- Funcionalitats bàsiques (implementació dels getters de la superclasse).

## 6. Classe AtributBoolea

**Descripció**: Representa un atribut boolea.
**Atributs:**

- **valor**: valor boolea (boolean) de l'atribut.

**Mètodes:**

- Funcionalitats bàsiques (implementació dels getters de la superclasse).

## 7. Classe AtributCategoric

**Descripció**: Representa un atribut categòric.
**Atributs:**

- **valor**: valors categòrics (Set<String>) de l'atribut.

**Mètodes**:

- Funcionalitats bàsiques (implementació dels getters de la superclasse).

## 8. Classe Valoracio

**Descripció**: Representa la valoració d'un ítem per part d'un usuari.
**Atributs**:

- **idItem:** identificador de l'ítem del qual s'ha valorat.
- **idUsuari:** identificador de l'usuari que ha realitzat la valoració.
- **puntuacio:** puntuació de l'usuari sobre l'ítem.

**Mètodes**:

- Funcionalitats bàsiques(*getters i setters*)

## 9. Classe Recomanacio

**Descripció**: Representa una recomanació que s'ha fet a un usuari.
**Atributs**:

- **idItem:** identificador de l'ítem del qual s'ha valorat.
- **idUsuari:** identificador de l'usuari que ha realitzat la valoració.
- **puntuacio:** puntuació de l'usuari sobre l'ítem.

**Mètodes**:

- Funcionalitats bàsiques(*getters i setters*)

## 10. Classe TipusItem

**Descripció:** Representa una estructura del tipus d'ítem.
**Atributs:**

- **nom:** el nom del tipus d'ítem.
- **tipusAtributs:** tipusAtribut que forma part del tipus d'ítem, compost per nom tipus d'atributs i TAtributs(NUMERIC, BOOLEA, CATEGORIC, DESCRIPTIU).

**Mètodes**:

- Funcionalitats bàsiques (*getters i setters*)
- Funcionalitats bàsiques(*getters i setters*)
- *afegirTipusAtribut(String nom, TAtribut tipus)*: afegeix un tipus d'atribut especificant el nom de l'atribut i el seu tipus.
- *canviarNomTipusAtribut(String nom1, String nom2)*: Canviar el nom del tipus d'atribut.
- *eliminarTipusAtribut(String nom)*: Eliminar el tipus d'atribut amb el nom indicat.

## 11. Classe Algorisme

**Descripció**: Classe abstracta, composta per dues subclases que són els dos tipus d'algorismes a utilitzar (K-means i content-based filtering)
**Atributs**: No en té
**Mètodes**:

- **resol() :** únic mètode, segons l'algoritme a utilitzar, tindrà la implantació del content-based filtering o del k-means.

## 12. Classe KMeans

**Descripció**: K-means és un algorisme de classificació no supervisada (clusterització) que agrupa objectes en k grups basant-se en les seves característiques. L'agrupament es realitza minimitzant la suma de distàncies entre cada objecte i el centroide del grup o clúster.
**Atributs**:

- **k:** nombre de baricentres.
- **n:** dimensió dels vectors proporcionats.
- **lim:** límit d'iteracions.
- **iter:** iteració actual.
- **baricentres:** els baricentres en la iteració i.
- **prevBaricentres:** baricentres de la iteració i-1.
- **data:** n vectors de d-dimensions que defineixen el conjunt de punts.
- **clusters:** [0, k) clústers, cada clúster té una llista on cada valor es correspon a data[i].

**Mètodes**:

- Funcionalitats bàsiques(*getters i setters*)
- *initBaricentres()*: inicialitza els baricentres de forma aleatòria.
- *initClusters()*: es creen els k-clusters
- *stop()*: es verifica si l'algorisme, retorna true si s'ha d'aturar, false altrament.
- *mitjanaCluster(ArrayList<Integer> cluster)*: calcula la mitjana d'un clúster.
- *distancia(ArrayList<Double> u, ArrayList<Double> v)*: calcula la distància entre dos vectors.

- *resol():* resol l'algorisme.

## 13. Classe SlopeOne

**Descripció**: classe amb una implementació de l'algorisme slope one per a predir valoracions d'ítems a partir de les valoracions d'usuaris d'altres ítems.
**Atributs**:

- **classificacio**: objecte on es pot obtenir les agrupacions dels usuaris.
- **valoracions:** valoracions dels usuaris als diferents ítems.
- **itemsDefinits:** conjunt dels ítems valorats per l'usuari actual.
- **idUsuari:** identificador de l'usuari actual.
- **prediccions:** valoració estimada per a cada ítem.
- **itemsOrd:** conjunt dels ítems ordenats d'acord amb la seva valoració

**Mètodes**:

- *preproces():* precomputa certes estructures per a agilitzar el procés de resolució.
- *estimaValoracio(int idItem):* estima una valoració per a un ítem determinat.

## 14. Classe ContentBasedFiltering

**Descripció:** L'algorisme fa una cerca seqüencial sobre tot el conjunt d'ítems C, guardant els k ítems més semblants a l'ítem donat d'entre els processats fins al moment.
**Atributs:**

- **k:** número d'ítems que guardarà el resultat.
- **itemsValorats:** conjunt dels ítems valorats per l'usuari.
- **conjuntItems:** conjunt dels ítems.
- **result:** resultat d'aplicar l'algoritme.

**Mètodes:**

- *getContentBasedFiltering()*: deixa a la variable result els ítems semblants que li han agradat a l'usuari.
- Funcionalitats bàsiques(*getter*).
- *resol()*: funció abstracta declara a la classe algoritme però aquesta, s'implementa a les subclases.

## 15. Classe Distancia

**Descripció**: Troba la distància entre dos atributs o dos ítems.
**Atributs**: No en té
**Mètodes**:

- *compararCategorics*(Set<String> a, Set<String> b) troba el grau de coincidència entre dos atributs categòrics.
- *compararNumerics*(double a, double b, double max) troba el grau de coincidència entre dos atributs numèrics.
- *compararBooleans*(boolean a, boolean b) troba el grau de coincidència entre dos atributs booleans.
- *compararDescriptius*(String a, String b) troba el grau de coincidència entre dos atributs descriptius.
- *calcularDistanciaMaxima(Set<Item> conjuntItems, HashMap<String, Integer> maximsCategorics, HashMap<String, Double> maximsNumerics)*: calcula la distància màxima que es pot assolir dins un conjunt d'items. Guarda els màxims categòrics i numèrics corresponents.

- *calcularDistancia*(Item a, Item b, HashMap<String, Double> maximsNumerics, double distanciaMaxima) troba la distancia entre dos ítems respecte un conjunt, a partir del precàlcul dels màxims numèrics i de la distància màxima.

## 16. Classe Classificacio

**Descripció**: classe que gestiona la classificació dels usuaris usant l'algorisme k-means. Aquesta classe s'encarrega d'adequar el valor k depenent del nombre d'ítems o del nombre d'usuaris per clúster. També s'encarrega de decidir quin és el percentatge d'ítems que es consideren representatius, això ho fa a partir del nombre de valoracions.

**Atributs**:

- ***Valoracions:*** valoracions dels usuaris als diferents ítems.
- ***usuariCluster:*** per a cada usuari, a quin clúster pertany.
- ***clusters:*** tots els clústers amb els seus usuaris.
- ***tall:*** el percentatge dels ítems que es consideraran representatius, és a dir, els XX % més valorats.
- ***k:*** grups inicials, l'algorisme el pot variar depenent de les dades i de resultats d'execucions diferents.

**Mètodes**:

- *preproces():* adequació de les dades per a poder executar l'algorisme.
- *classifica():* fer una nova classificació de tots els usuaris.
- *afegirValoracio(int idUsuar, int idItem, double valoracio):* afegir una nova valoració.

## 17. Classe Avaluacio

**Descripció**: classe per a avaluar la qualitat de les recomanacions amb la mètrica cdg.

**Atributs**:

- ***trueRawData:*** dades esperades en el format idUsuari, idItem, valoracio.
- ***trueItemPos:*** posició esperada d'un ítem.
- ***trueVal:*** valoració esperada d'un ítem.
- ***predItemPos:*** posició en la que es recomanaria un ítem.

**Mètodes**:

- *avalua(int idUsuari, Array<String[]> recomanacions):* obté la valoració de les recomanacions per un usuari concret.
- *processarOrdre(int idUsuari, ArrayList<String[]> data, HashMap<Integer, Integer> ordreM, Set<Integer> acceptedItems): determina l'ordre que ocupa un cert item si les prediccions per un usuari estiguessin ordenades.*

## 18. Classe CtrlDomini

**Descripció**: Representa el controlador del domini, conjunts de funcionalitats principals de les classes necessàries, aquest està format per controladors del domini secundaris i controlador de les dades.

**Atributs:**

- **singletonObject**: Controlador del domini
- **ctrlDades:** controlador de dades
- **ctrlUsuari:** controlador d'usuari
- **ctrlItem:** controlador d'ítem
- **ctrlValoracio:** controlador de valoració
- **ctrlRecomanacio**: controlador de recomanació.

**Mètodes:**

- Funcionalitats bàsiques(*Getters i setters*)
- Totes les funcions que té els controladors del domini secundaris.

## 19. Classe CtrlValoracio

**Descripció:** representa el controlador del domini secundari en el qual controla les valoracions.

**Atributs:**

- **singletonObject:** controlador de la valoració.
- **ctrlItem:** controlador dels ítems
- **ctrlUsuari:** controlador dels usuaris
- **valoracionsItem:** valoracions sobre l'ítems.
- **valoracionsUsuari:** valoracions fetes per usuaris.

**Mètodes:**

- Funcionalitats bàsiques(*Getters i setters*)
- *afegirValoracio(int idItem, int idUsuari, double puntuacio)*: afegeix una valoració nova.
- *modificarValoracio(int idItem, int idUsuari, double puntuacio):* modifica la puntuació d'una valoració.
- *eliminarValoracio(int idItem, int idUsuari):* elimina una valoració.
- *eliminarValoracionsUsuari(int idUsuari):* elimina totes les valoracions fetes per l'usuari.
- *eliminarValoracionsItem(int idItem):* elimina totes les valoracions sobre l'ítem.

## 20. Classe CtrlUsuari

**Descripció:** representa el controlador del domini secundari en el qual controla els usuaris.

**Atributs:**

- ***singletonObject*:** controlador dels usuaris.
- **ctrlDades:** controlador de les dades.
- **ctrlValoracio:** controlador de les valoracions.
- **usuaris:** tots els usuaris de l'aplicatiu identificat per id.
- **idNomUsuari:** els ids correponents als noms dels usuaris.
- **idUsuariActiu:** id d'usuari que s'ha iniciat la sessió.

**Mètodes:**

- Funcionalitats bàsiques(*Getters i setters*)
- *carregarUsuaris():* carregar els usuaris des del fitxer usuaris.json
- *guardarUsuaris():* guardar els usuaris en el fitxer usuaris.json
- *registrarUsuari(String nom, String contrasenya1, String contrasenya2, boolean admin)*: registra un usuari nou.
- *loginUsuari(String nom, String contrasenya):* inicia la sessió d'un usuari.
- *logoutUsuari():* usuari tanca la sessió.
- *canviarContrasenya(String contrasenya1, String contrasenya2)*: canviar la contrasenya de l'usuari actiu.
- *canviarNom(String nom):* canviar el nom de l'usuari actiu.
- *eliminarUsuari(int id):* eliminar un usuari.
- *existsUsuari(int idUsuari):* indica si existeix usuari amb idUsuari.

## 21. Classe CtrlItem

**Descripció:** representa el controlador del domini secundari en el qual controla els ítems.
**Atributs:**

- ***singletonObject*:** controlador dels ítems.
- **ctrlDades:** controlador de les dades.
- **ctrlValoracio:** controlador de les valoracions.
- **ítems:** el conjunt d'ítems que conté la dataset.
- **tipusItems:** tots els tipus d'ítems que conté la dataset.

**Mètodes:**

- Funcionalitats bàsiques(*Getters i setters*)
- *crearTipusItem(String nomTipusItem):* crea un tipus d'ítem nou.
- *eliminarTipusItem(String nom):* elimina un tipus d'ítem.
- *afegirTipusAtribut(String nomTipusItem, String nomTipusAtribut, TipusItem.TAtribut tAtribut):* afegeix un tipus d'atribut en un tipus d'ítem.
- *canviarNomTipusAtribut(String nomTipusItem, Strong nomTipusAtribut1, StringNomTipusAtribut2):* canviar el nom del tipus atribut.
- *eliminarTipusAtribut(String nomTipusItem, String nomTipusAtribut):* eliminar tipus d'atribut del tipus d'item especificat.
- *crearItem(int id, String nomTipusItem):* crear un ítem nou.
- *eliminarItem(int id):* eliminar un ítem.
- *afegirAtribut(int id, String nomTipusAtribut, Atribut valor):* afegir un atribut en l'ítem especificat.
- *modificarAtribut(int id, String nomTipusAtribut, Atribut valor):* modificar el valor de l'atribut de l'ítem proporcionat.
- *eliminarAtribut(int id, String nomTipusAtribut):* eliminar l'atribut de l'ítem indicat.
- *carregarItemsCSV(String nomTipusItem):* carregar un tipus d'ítem i el ítems que pertany aquest tipus des del fitxer items.csv.
- *existsItem(int idItem):* comprova si existeix l'ítem l'idItem proporcionat.

## 22. Classe CtrlRecomenacio

**Descripció**: Controlador que s'encarrega de gestionar tot allò relacionat amb les recomanacions dels usuaris.
**Atributs**:

- ***classificacio:*** objecte per a gestionar l'agrupació dels usuaris.
- ***slopeOne:*** objecte per a estimar valoracions i inferir un ordre en les recomanacions a partir d'ítems valorats per altres usuaris del mateix grup.
- ***recomanacions:*** recomanacions per a cada usuari.

**Mètodes**:

- *classifica():* fer una nova classificació dels usuaris disponibles.
- *recomana(int idUsuari):* fer noves recomanacions a l'usuari.
- *avaluacioRecomanacions(String filename):* avaluar totes les possibles recomanacions.
- *carregar():* carregar tot el necessari de disc, per tal que la classe sigui operativa.
- *guardar():* guardar tot el necessari a disc per a poder recuperar l'estat més endavant.
- *getRecomanacions(int idUsuari, int k)*: obtenir les k millors recomanacions per l'usuari donat.
- getItemsSimilars(int idUsuari, int k, String tipusItem): troba els k ítems similars als ítems valorats positivament per l'usuari del tipus especificat.

## 23. Classe CtrlDades

**Descripció**: Representa el controlador de les dades.
**Atributs**:

- **CtrlDades singletonObject**
- **CtrlItemCSV ctrlItemCSV**
- **JsonParser parser**

**Mètodes**:

- Funcionalitats bàsiques(*Getters i setters*)
- *carregarItemCSV():* Carregar els ítems d'un fitxer.
- *loadUsuaris():* Carregar els usuaris d'un fitxer.
- *saveUsuaris(List<Usuari> usuaris):* Guardar els usuaris en el fitxer.

## 24. Classe CtrlItemCSV

**Descripció**: Controlador que s'encarrega de carregar els datasets en format csv inferint els atributs i el seu tipus.
**Atributs:**

- ***objects:*** taula clau valor amb les propietats i els seus valors.
- ***fields:*** propietats trobades amb el seu tipus.
- ***fieldOrder:*** ordre en la qual s'han trobat els atributs al fitxer.

**Mètodes:**

- *load(String filename)*: carregar la dataset del fitxer especificat.

## 25. Classe CtrlRatingsCSV

**Descripció**: Controlador que s'encarrega de carregar els arxius amb les tripletes usuari, ítem i valoració.

**Mètodes:**

- *load(String filename, String delimiter):* carregar les tripletes del fitxer especificat amb el delimitador donat.
- *dump(String filename, String delimiter, ArrayList<String[]> ratings):* bolcar la llista de tripletes al fitxer especificat amb el delimitador donat.

## 26. Classe CtrlAvaluacioCSV

**Descripció**: Controlador que s'encarrega de guardar les valoracions de les recomanacions.
**Mètodes:**

- *dump(String filename, String delimiter, ArrayList<String[]> ratings):* bolcar la llista de tripletes al fitxer especificat amb el delimitador donat.

## 27. Classe JsonParser

**Descripció**: Classe que s'encarrega de guardar i llegir objectes del domini en format json fent servir reflection.
**Mètodes:**

- *loadObjects(Class<?> objectType, String filename, String arrayName):* carregar un conjunt d'objectes del mateix tipus del fitxer donat.
- *saveObjects(List<T> objects, Class<T> objectType, String filename, String arrayName):* guardar un conjunt d'objectes del mateix tipus en el fitxer json especificat.

## 28. Classe Hibrid

**Descripció**: Classe que s'encarrega de l'aplicació de les tècniques de recomanació híbrides combinant els resultats dels algorismes de content-based i el collaborative.
**Atributs:**

- ***slopeOne:*** instància de l'algorisme slope one.
- ***tipusItems:*** els tipus d'ítems dels quals es volen obtenir les recomanacions.
- ***idUsuari:*** usuari sobre el que es volen obtenir les recomanacions.
- ***itemsValorats:*** conjunt d'ítems valorats per l'usuari actual.
- ***conjuntItems:*** conjunt de tots els ítems classificats pel seu tipus.
- ***items:*** resultat de l'algorisme híbrid, és a dir, els ítems ordenats descendentment per la seva valoració estimada.
- ***predScore:*** puntuació estimada per a cada ítem.
- ***idToItem:*** correspondència id-ítem.

**Mètodes:**

- *resol()*: aplicar l'algorisme híbrid sobre l'usuari i els ítems seleccionats.

## 29. Classe CtrlPresentacio
Classe controlador que relaciona totes les vistes de la capa Presentació entre elles i amb la resta de capes. Està connectada amb el CtrlDomini per intercanviar informació en el Model Vista Controlador. Per implementar totes aquestes classes d'aquesta capa hem utilitzat la llibreria swing i AWT.

## 30. Classe VistaLoginRegistre
Vista que ens trobem només iniciem el programa, on l'usuari validarà les seves credencials i passarà a la Vista Principal o bé, si encara no té un compte, tindrà l'opció de registrar-se, que ho farà donant un nom d'usuari, que haurà de ser únic i una contrasenya.

## 31. Classe VistaPrincipalUsuari
Vista principal de l'usuari, és la vista que trobem pels usuaris que no són administradors.
Aquesta vista serveix de punt d'entrada per les accions que pot realitzar un usuari com modificar el perfil, valorar items, cercar etc.

## 32. Classe VistaPrincipalAdmin
És la vista que veu l'administrador un cop entra al aplicatiu després de la VistaLogin. És el punt d'accés a les accions que només pot realitzar un administrador, com ara afegir un ítem, definir un tipus de ítem nou etc.

## 33. Classe VistaPerfilUsuari
Vista que esdevé de triar el l'opció Modificar Perfil en la vista principal del usuari. En aquesta vista es poden modificar els atributs d'un usuari com ara el seu nom o la contrasenya.

## 34. Classe VistaGestioItems
Aquesta vista proporciona a un usuari administrador un CRUD sobre els ítems d'un tipus determinat que estaran llistats.

## 35. Classe VistaEditarItem
Aquesta classe permet editar els atributs d'un ítem determinat. Aquesta classe es pot cridar quan es crea un ítem nou, amb tots els atributs per assignar, o quan s'edita un ítem, amb els atributs ja carregats.

## 36. Classe VistaGestioTipusItem
Vista que proporciona a un usuari administrador un CRUD sobre els tipus de ítems que estaran llistats.

# 3. Estructures de dades i algorismes utilitzats

## 1. Estructures de dades

Per a la majoria d'implementacions hem fet servir estructures de taules de hash (HashMap) per tenir l'accés constant a les dades necessàries.

## 2. Algorismes de recomanació d'ítems

Les dues tècniques per elaborar la recomanació d'ítems a un conjunt d'usuaris eren el Collaborative filtering i el Content-based filtering.

### 2.1. Collaborative filtering

S'utilitzen dos algoritmes que son el **k-means** y el **Slope one**.

**K-means**
Començant amb una partició inicial a l'atzar de centroides de k clústers, explota la idea de canviar la partició actual a una altra que disminueixi la suma de quadrats de distàncies de les observacions als centroides dels clústers.

El K-means permet dues assignacions dels centroides: aleatòria o fixant-los prèviament.

L'estructura de dades que s'ha utilitzat per tractar totes les dades d'aquest algoritme han sigut el **doble ArrayList**. Per guardar el conjunt de clústers, els baricentres, els baricentres a la iteració i-1, i per guardar els vectors que defineixen el conjunt de punts.

**Slope One**
Aquest algorisme bàsic consisteix en estimar/predir la valoració que donaria l'usuari actiu a un ítem donat, a partir de les valoracions d'altres usuaris

Les estructures de dades que s'han utilitzat en aquest algorisme han sigut:

- Una instància de **classificació**, per obtenir els usuaris similars a un usuari i per obtenir les valoracions del conjunt d'usuaris.
- un hashmap doble per les **valoracions**
- un HashSet pels **items** definits.
- un HashMap per les **prediccions** que consisteix en la predicció d'una valoració d'un ítem per part d'un usuari i una llista dels items ordenats.

### 2.2. Content-based filtering

Són estratègies que es basen en recomanar a l'usuari actiu els ítems semblants als que li han agradat a ell. Utilitza una variació de l'algorisme k-NN.

Té dues estructures de dades que són dos **Sets**, un pels **ítems valorats**, i un altre Set que es pel **conjunt d'ítems**. I a més a té un **ArrayList per guardar el resultat**, els ítems més semblants als que li agraden a un usuari.
Aquest algorisme consisteix en, trobar per cada ítem valorat dins el conjunt d'ítems valorats, els k elements més propers a aquests dins d'un conjunt total d'ítems. D'entre tots els k elements més propers de cada ítem valorat, s'agafen els k ítems (sense repeticions) que més propers han sigut a algun dels ítems valorats. Aquest últim grup d'ítems són els destinats a ser recomanats.

Per implementar aquest algorisme, s'ha fet servir una classe d'utilitat *Pair*, on l'atribut *first* és un valor *double*, que serveix per guardar distàncies, i l'atribut *second* és l'ítem. Aquesta classe facilita l'ordenació d'ítems segons la distància que s'ha calculat respecte a un altre ítem, establint prioritat per aquells *Pair* que tinguin una menor distància. Això ens permet poder implementar una cua de prioritat destinada a guardar els ítems comparats de manera ordenada, prioritzant aquells que tinguin menor distància. D'aquesta manera, quan s'acaben totes les comparacions entre ítems, només caldria agafar els k primers de la cua.

### 2.3. Híbrid

L'algorisme híbrid s'encarrega de combinar les recomanacions de l'estratègia collaborative filtering amb la de content-based filtering. Per a fer la combinació segueix una sèrie de passos.

1. Obté les puntuacions del collaborative normalitzades entre 0 i 1.
2. Obté les distàncies del content-based normalitzades entre 0 i 1.
3. Pels ítems comuns als 2 obté una puntuació calculada com el punt mig entre les dues puntuacions. Pels que no siguin comuns conserva la puntuació obtinguda per l'algorisme corresponent.

Per a guardar el resultat far servir un **ArrayList** amb els ítems ordenats per la seva puntuació estimada. Paral·lelament, manté un **HashMap** on per a cada ítem es té el valor de la puntuació.

## 3. Distància

La distància entre ítems sempre es calcula respecte a un conjunt total d'ítems.

Per calcular la distància entre dos items, tenim en compte una distància màxima. Aquesta distància màxima es calcula com el sumatori de:
- Número d'atributs descriptius
- Número d'atributs numèrics
- Numero d'atributs booleans
- En el cas dels atributs categòrics, per cada atribut categòric es considera la quantitat més gran de categories que hi ha dins un conjunt d'ítems

Si per exemple tenim un conjunt de dos ítems (han de ser del mateix tipus), amb un atribut de cada tipus (1 descriptiu, 1 numèric, 1 boolea, 1 categòric), i l'ítem 1 te 5 categories al conjunt de categories i l'item 2 te 3, la distancia màxima serà 1+1+1+max(5,3)=8.
Per tant la distància serà, entre cada ítem del conjunt:

[distància(ítem1, ítem2)] = [distància màxima(conjunt)] - [grau de coincidència total(ítem1, ítem2)]

Aquest grau de coincidències es calcula comparant un per un els atributs del mateix tipus (és a dir, amb el mateix nom d'atribut), per això és important que siguin del mateix tipus d'ítem:
- Si un dels dos ítems no tenen un atribut determinat (null), no incrementa el grau de coincidència
- Atributs descriptius: si les dues descripcions són iguals, el grau de coincidència incrementa 1
- Atributs numèrics: el grau de coincidència s'incrementa 1-diferència de valors normalitzat respecte el valor màxim que hi ha d'aquest atribut dins el conjunt d'ítems
- Atributs booleans: si els dos booleans tenen el mateix valor, el grau de coincidència incrementa 1
- Atributs categòrics: incrementa 1 per cada coincidència en categories

# 4. Capa de presentació.

Per a la capa de presentació hem treballat amb les llibreries AWT i SWING.