

PROP

| Sistema Recomanador

SEGONA ENTREGA
Disseny

Casau Pueyo, Sergi
El Attar Yalaoui, Abdelali
García Soler, Miguel
Guixaró Tranco, Ricard

sergi.casau@estudiantat.upc.edu
abdelali.el.attar@estudiantat.upc.edu
miguel.garcia.i@estudiantat.upc.edu
ricard.guixaro@estudiantat.upc.edu

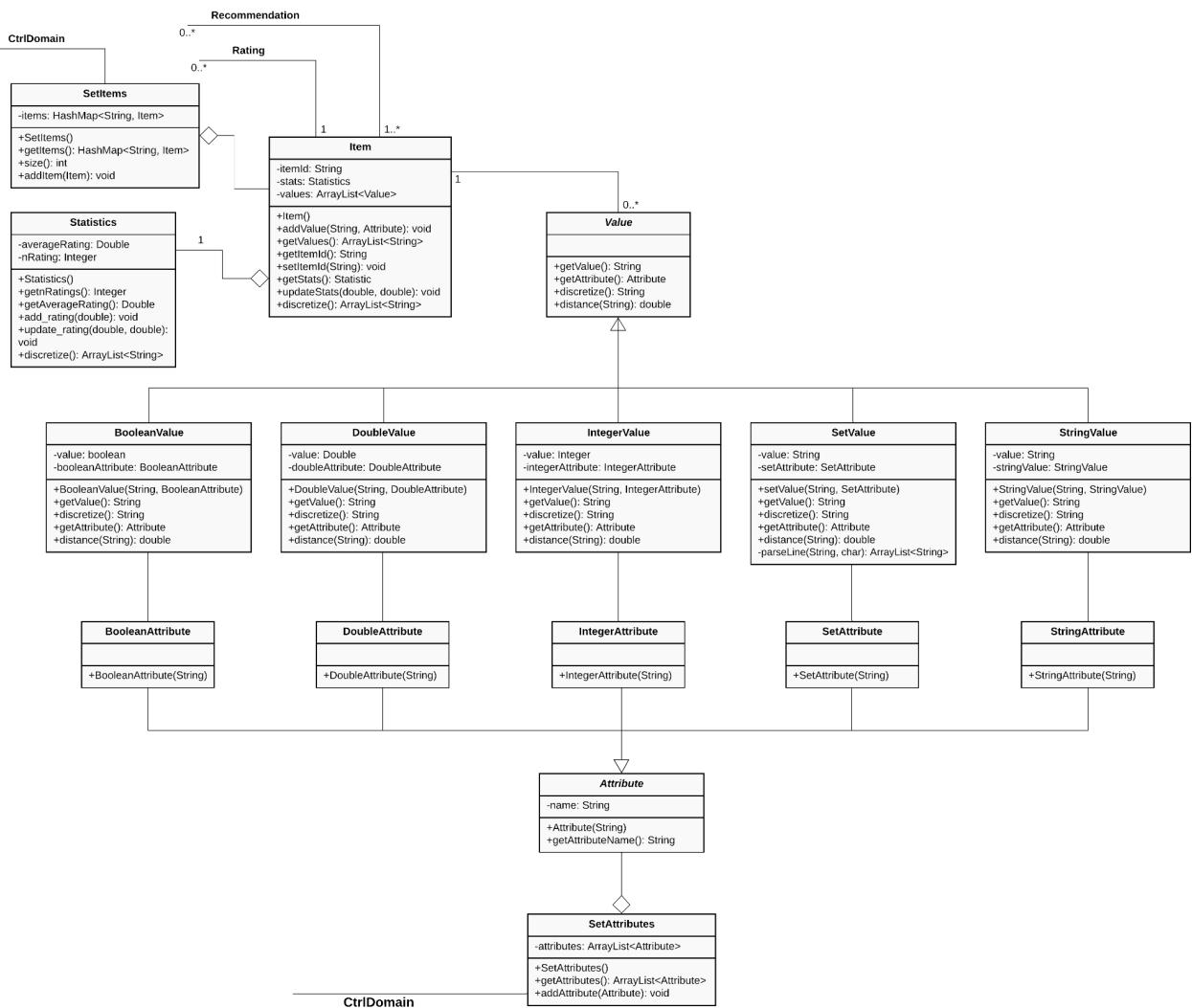
ÍNDEX

ÍNDEX	1
DIAGRAMA CAPA DOMINI - Esquema	2
DIAGRAMA CAPA DOMINI - Explicació	6
DIAGRAMA CAPA PRESENTACIÓ - Esquema	20
Diagrama UML	20
Diagrama de pantalles navegabilitat	21
DIAGRAMA CAPA PRESENTACIÓ - Explicació	27
DIAGRAMA CAPA PERSISTÈNCIA - Esquema	30
DIAGRAMA CAPA PERSISTÈNCIA - Explicació	31
DESCRIPCIÓ ESTRUCTURES DE DADES	34
DESCRIPCIÓ FUNCIONALITATS PRINCIPALS	35
Collaborative Filtering	35
K-Means:	35
SlopeOne:	35
Content Based Filtering	36
Hybrid Approaches	37
Avaluació Recomanacions	38
REPARTICIÓ CLASSES	39
ANNEX	41

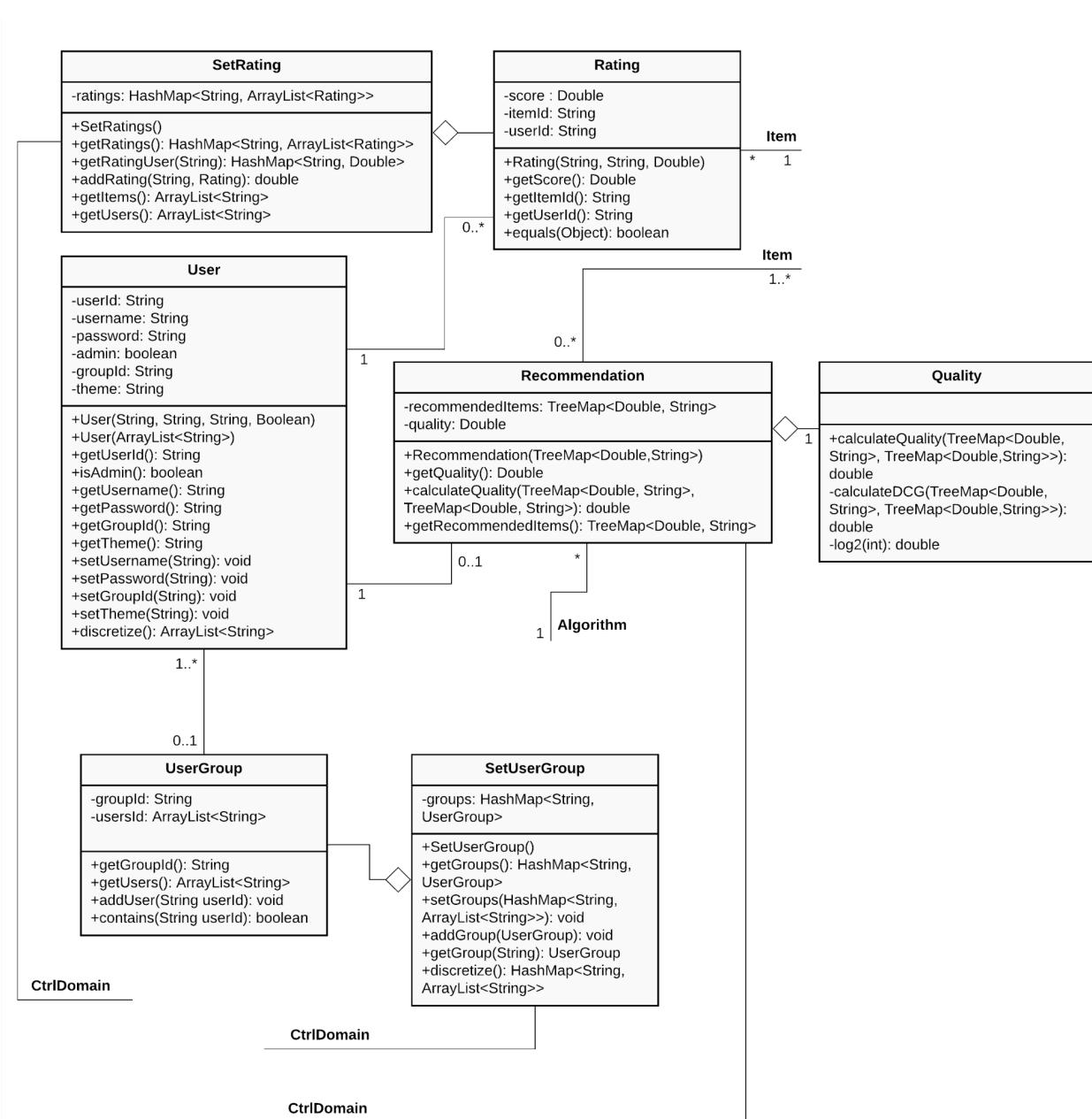
1. DIAGRAMA CAPA DOMINI - Esquema

Al tractar-se d'un esquema què no cap en una única pàgina, l'hem fragmentat en diverses parts. A l'Annex, però, hi ha la versió completa.

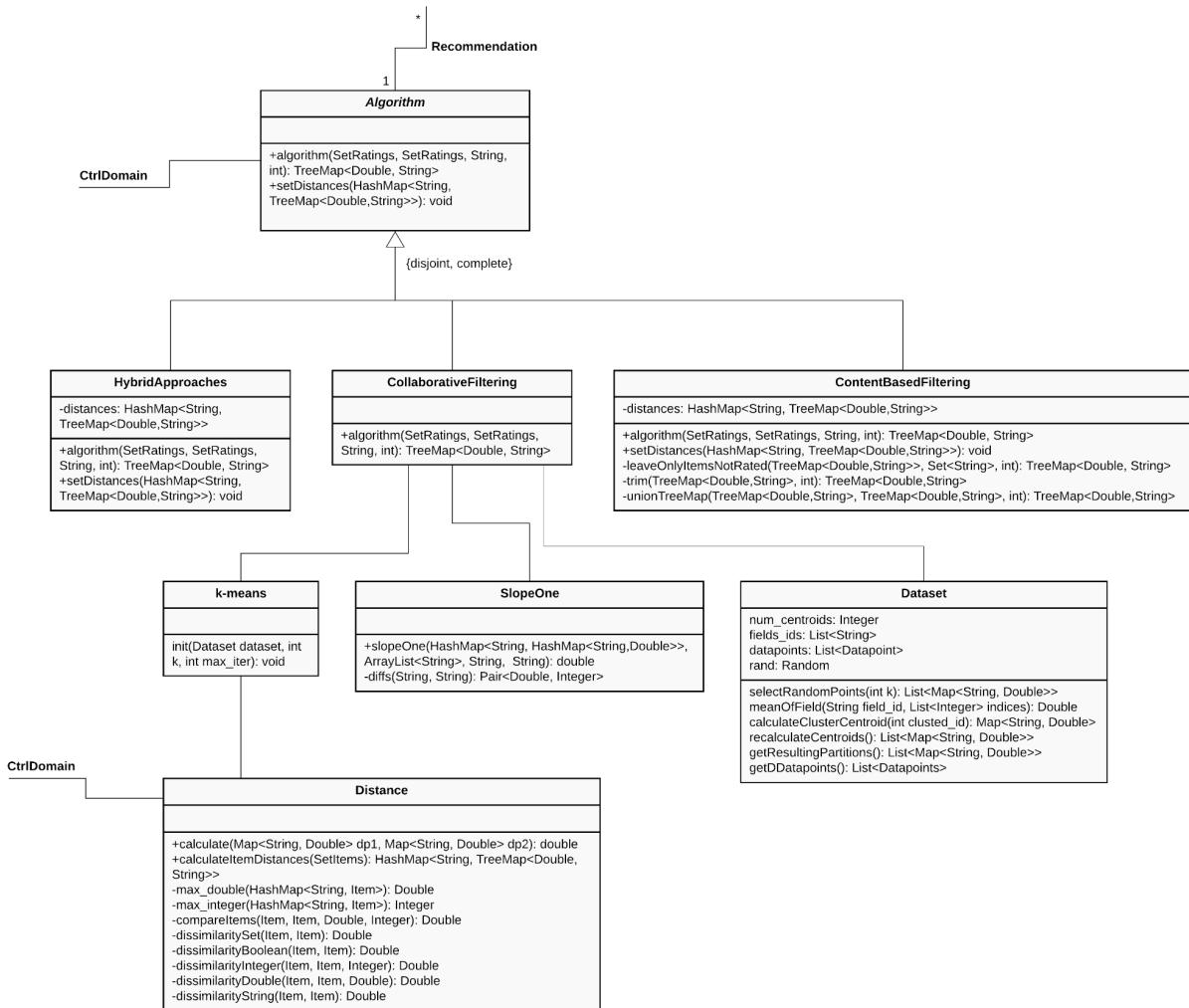
Classes de la gestió dels ítem i els seus atributs i valors.



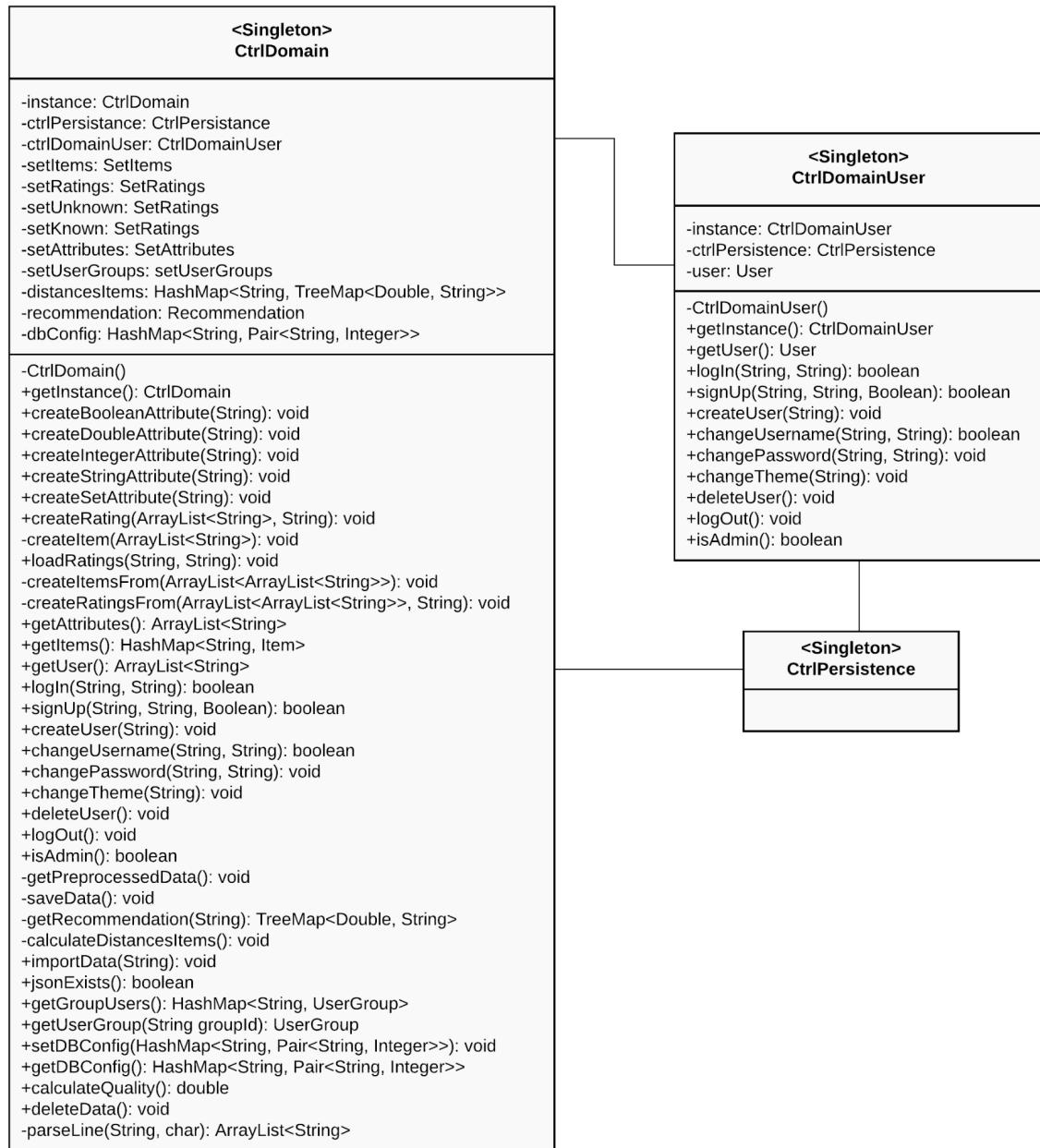
Classes de la gestió de les recomanacions, valoracions i usuaris:



Classes de la gestió d'algorismes i distàncies:



UML dels controladors de domini:



2. DIAGRAMA CAPA DOMINI - Explicació

- **Rating:**

Un *Rating* fa referència a un ítem *itemId* i a un usuari *userId*. Conté la valoració en forma de *Double* que ha fet un usuari a un determinat ítem.

- Atributs:

```
private Double score; Puntuació de la valoració
private String userID; Usuari que fa la valoració
private String itemID; Item que es valora
```

- Mètodes:

- `public Rating(String userID, String itemID, Double score)`

És la creadora amb valors de la classe.

- `public boolean equals(Object b)`

Retorna un booleà que indica si el Rating actual i el passat com a paràmetre són iguals.

- **SetRatings:**

Classe que emmagatzema un conjunt de Ratings en un `HashMap<String, ArrayList<Rating>>`.

- Atributs:

- `private HashMap<String, ArrayList<Rating>> ratings;`

L'únic atribut que té és un mapa on es guarden tots els ratings en conjunt.

- Mètodes:

- `public HashMap<String, Double> getRatingUser(String user)`

Retorna els ratings que ha fet l'usuari donat.

- `public double addRating(String user, Rating rating)`

Afegeix un Rating al conjunt de ratings si no hi era anteriorment i el modifica si ja existia.

- `public ArrayList<String> getItems()`

Retorna el conjunt d'ítems del set de ratings.

- `public ArrayList<String> getUsers()`

Retorna el conjunt d'usuaris del set de ratings.

- **User:**

Un *User* està identificat per *userId*. Conté informació de l'usuari (si és administrador del sistema o és un usuari corrent, el seu *username*, la seva *password* i el grup d'usuaris al qual pertany).

- Atributs:

- `private String userId;` L'identificador de l'usuari
- `private String username;` Username de l'usuari.
- `private String password;` Password de l'usuari.
- `private boolean admin;` Booleà que indica si l'usuari és administrador.
- `private String groupId;` Grup al qual pertany l'usuari.
- `private String theme;` Tema de l'aplicació de l'usuari.

- Mètodes:

- `public User(String userId, String username, String password, Boolean admin)`

Constructora de la classe *User*, crea un usuari amb les dades introduïdes com a paràmetre.

- `public User(ArrayList<String> data)`

Constructora de la classe *User*, crea un usuari amb les dades introduïdes com a paràmetre.

- `public ArrayList<String> discretize()`

La funció retorna una llista que conté totes les dades pròpies de la classe *usuari*.

- **UserGroup:**

Un *UserGroup* està identificat per *groupId*. Conté els identificadors d'aquells usuaris que pertanyen al grup, permet consultar si un *User* pertany al grup o no.

- Atributs:

- `private String groupId;` Identificador del grup d'usuaris.
- `private ArrayList<String> usersId;` Identificadors dels usuaris que pertanyen al grup.

- Mètodes:

- `public UserGroup(String groupId)`

Constructora d'un grup d'usuaris amb l'identificador.

- `public UserGroup(String groupId, ArrayList<String> users)`

Constructora d'un grup d'usuaris amb identificador i llista d'usuaris.

- `public boolean contains(String userId)`

Operació que retorna si un usuari pertany al grup.

- **SetUserGroup:**

Aquesta classe emmagatzema els grups d'usuaris.

- Atributs:

- `private HashMap<String, UserGroup> groups;` Guarda el conjunt de grups d'usuaris amb el seu identificador.

- Mètodes:

- `public SetUserGroup()`

Creadora buida del conjunt de grups d'usuaris.

- `public void addGroup(UserGroup group)`

Afegeix el grup especificat al paràmetre al conjunt.

- `public UserGroup getGroup(String groupId)`

Retorna el grup d'usuaris identificat per groupId.

- `public HashMap<String, ArrayList<String>> discretize()`

Retorna els grups d'usuaris en un format estàndard.

- **Item:**

Aquesta classe conté la informació d'un ítem juntament amb tots els seus conjunts d'atributs (de tots els tipus; String, Integer, Double, Boolean i Set<String>). Un Item està identificat pel seu identificador, *itemId*. També té associada informació sobre les seves estadístiques.

- Atributs:

- `private String itemId;` Identificador de l'ítem
- `private Statistics stats;` Estadístiques de l'ítem
- `private ArrayList<Value> values;` Valors dels atributs de l'ítem

- Mètodes:

- `public Item()`

Constructora de la classe Item buida, sense dades.

- `public void addValue(String value, Attribute attribute)`

Afegeix un valor de l'atribut amb el seu tipus corresponent ítem.

- `public void updateStats(double rating, double oldRating)`

Modifica les estadístiques associades a l'ítem després d'afegir el rating passat com a primer paràmetre.

- `public ArrayList<String> discretize()`

Retorna l'ítem amb els seus valors en format d'un array de strings, és a dir, retorna l'ítem de forma genèrica en format String.

- **SetItems:**

Aquesta classe conté el conjunt d'ítems de la base de dades.

- Atributs:

- `private HashMap<String, Item> items;` HashMap que emmagatzema els ítems.

- Mètodes:

- `public SetItems()`

Constructora buida d'un set d'ítems.

- `public void addItem(Item item)`

Afegeix un ítem al conjunt total d'ítems.

- **Statistics:**

És una classe auxiliar d'*Item* que emmagatzema les estadístiques associades als ítems. Està identificada per l'*Item* i permet ser actualitzada cada vegada que s'afegeix un nou *Rating*.

- Atributs:

- `private Integer nRatings;` Enter que indica el nombre de ratings efectuats.
- `private Double averageRating;` Double que indica la valoració (o rating) mitjana.

- Mètodes:

- `public Statistics()`

Constructora buida sense dades de la classe.

- `public void add_rating(double rating)`

Afegeix un rating a les estadístiques (el passat com a paràmetre) i actualitza les dades.

- `public void update_rating(double new_rating, double old_rating)`

Es modifica un rating si aquest ha canviat de valor (`new_rating != old_rating`) i s'actualitza la valoració mitjana.

- `public ArrayList<String> discretize()`

Retorna la instància de la classe en qüestió en un format estàndard, on tots els atributs de la classe es troben en un array de strings.

- **Quality:**

És una classe auxiliar de *Recommendation* que calcula la qualitat d'una recomanació. La funció principal calcula el *Discounted Cumulative Gain* de la *Recommendation* calculada en funció d'una *Recommendation* real.

- Mètodes:

- `public static double calculateQuality(TreeMap<Double, String> predictedData, Set<String> setUnknown, String user)`

Operació pública per calcular la qualitat.

- `private static double calculateDCG(TreeMap<Double, String> predictedData, HashMap<String, Double> ratingsUser)`

Calcula el Discounted Cumulative Gain tenint en compte la recomanació que s'ha predit, i la del fitxer unknown.

- **Recommendation:**

Aquesta classe emmagatzema un conjunt ordenat d'itemId's i preferències. També guarda la qualitat d'una *Recommendation*, que es calcula mitjançant la funció: `private void public void calculateQuality(TreeMap<Double, String> realRecommendation)`. A més a més, crea la classe *Algorithm* en funció d'un atribut.

- Atributs:

- `private Double quality;` És la qualitat de la recomanació

- `private TreeMap<Double, String> recommendedItems;` Emmagatzema els ítems recomanats decreixentment per la similitud.
- Mètodes:
 - `public Recommendation(TreeMap<Double, String> recommendedItems)`
Constructora de la classe Recommendation amb el seu l'identificador (primer paràmetre) i el tipus d'algorisme usat per recomanar (segon paràmetre).
 - `public double calculateQuality(SetRatings setUnknown, String user)`
Calcula la qualitat d'una recomanació real que és la passada com a paràmetre.

- Algorithm:

Classe abstracta des d'on es calculen les recomanacions d'ítems.

- Mètodes abstractes:
 - `public abstract TreeMap<Double, String> algorithm(SetRatings known, SetRatings setUnknown, String user, int k);`
Calcula la recomanació i la retorna en un TreeMap ordenat per similitud.

- Collaborative Filtering:

Classe on es crea una *Recommendation* seguint l'estrategia Collaborative Filtering, combinant el Kmeans amb el SlopeOne.

- Mètodes:
 - `public TreeMap<Double, String> algorithm(SetRatings setKnown, SetRatings setUnknown, String user, int k)`
Calcula la recomanació en funció de l'algorisme Collaborative i la retorna en un TreeMap ordenat per similitud.

- K-Means:

Classe que s'encarrega de fer el clustering d'un dataset representat per un Array de HashMap de String i Double. És independent del concepte de *user* i *items*.

- Atributs:
 - `private List<Integer> cluster_IDs;`
S'utilitza per fer el seguiment dels clústers als que es va assignant cada datapoint.

- `private static final Random rand;`

Serveix per generar valors aleatoris.

- Mètodes:

- `private List<Map<String, Double>> selectRandomPoints(List<Map<String, Double>> dataset, int k)`

Genera els centroides inicials seleccionant K *datapoints* aleatoris del dataset.

- `public Map<Map<String, Double>, List<Map<String, Double>>> compute(List<Map<String, Double>> dataset, int k, int max_iter, Distance distance);`

Funció principal de la classe que s'encarrega d'executar l'algorisme i retorna la llista de centroides associada amb el conjunt de *datapoints* de cada cluster.

- `Map<String, Double> nearestCentroid(Map<String, Double> datapoint, List<Map<String, Double>> centroids, Distance distance);`

Donat un *datapoint* i una llista de centroides retorna el centroide més proper al *datapoint* segons la funció de distància del interface *distance*.

- `Map<String, Double> averageCentroid(Map<String, Double> centroid, List<Map<String, Double>> datapoints);`

Retorna el centroide que representa la mitjana del conjunt de *datapoints* d'un cluster. Accepta com a paràmetre el centroide "antic" per si el conjunt de *datapoints* és buit, en altres paraules, hi ha un cluster que no té assignat cap *datapoint* i en aquest cas no es modifica el centroide.

- `private List<Map<String, Double>> recalculateCentroids(int k);`

Aplica el mètode *averageCentroid(...)* per cadascun dels K centroides amb els quals està treballant l'algorisme.

- **SlopeOne:**

Classe que prediu la valoració que faria l'usuari actiu a un determinat *Item*, en funció dels *Ratings* que han fet usuaris del seu *UserGroup* a aquell *Item*.

- Mètodes:

- `public static double slopeOne(HashMap<String, HashMap<String, Double>> input, ArrayList<String> usersGroup, String userId, String itemId)`

Fa la predicció que faria l'usuari a l'ítem tenint en compte les valoracions dels usuaris del seu mateix grup.

```
- private static Pair<Double, Integer> diffs(HashMap<String, HashMap<String, Double>> input, String itemId1, String itemId2, ArrayList<String> usersGroup)
```

Calcula la desviació entre dos ítems.

- **Content-based Filtering:**

Classe on es crea una *Recommendation* seguint l'estrategia Content-based Filtering. Implementa un algorisme *k*-nearest neighbours (*k*-NN).

- Atributs:

```
- private HashMap<String, TreeMap<Double, String>> distances;
```

Emmagatzema les distàncies entre els ítems.

- Mètodes:

```
- public TreeMap<Double, String> algorithm(SetRatings setKnown, SetRatings setUnknown, String user, int k)
```

Calcula la recomanació en funció de l'algorisme KnN mitjançant les distàncies entre els ítems. La retorna en un TreeMap ordenat per similitud.

```
- private TreeMap<Double, String> leaveOnlyItemsNotRated(TreeMap<Double, String> similarItems, Set<String> itemsNotRated, int k)
```

Treu de la recomanació aquells ítems que l'usuari ja ha valorat.

```
- private TreeMap<Double, String> trim(TreeMap<Double, String> recommendation, int k)
```

Retallar el TreeMap per deixar únicament els *k* ítems més similars.

```
- private TreeMap<Double, String> unionTreeMap(TreeMap<Double, String> t1, TreeMap<Double, String> t2, int k)
```

Fusiona dos TreeMap's en un de sol.

- **Hybrid Algorithm:**

Classe on es crea una *Recommendation* seguint una estratègia que combina tècniques basades en les dues aproximacions anteriors.

- Atributs:

- `private HashMap<String, TreeMap<Double, String>> distances;`
Emmagatzema les distàncies entre els ítems.

- Mètodes:

- `public TreeMap<Double, String> algorithm(SetRatings setKnown, SetRatings setUnknown, String user, int k)`

Calcula primer la recomanació següent l'algorithm Collaborative Filtering, i posteriorment, amb el Content Based. Fusiona les dues recomanacions i retorna la definitiva en un TreeMap ordenat per similitud.

- **Distance:**

Classe que calcula la distància entre dos usuaris basant-se en els Rating's de tots els Item's que han valorat en comú. A més, també permet calcular la distància entre dos ítems.

- Mètodes:

- `public static double calculate(Map<String, Double> dp1, Map<String, Double> dp2)`

Calcula la distància entre datapoints (generalització dels usuaris i la valoració dels items).

- `public static HashMap<String, TreeMap<Double, String>> calculateItemDistances(SetItems setItems)`

Calcula les distàncies entre els ítems del set. Retorna un HashMap amb l'identificador de cada ítem, i un TreeMap amb les distàncies entre l'ítem i els altres, ordenat decreixentment.

- `private static Double max_double(HashMap<String, Item> items)`

Busca el double més gran dintre dels valors de tots els ítems. Necessari per calcular la similitud posteriorment.

- `private static Integer max_integer(HashMap<String, Item> items)`

Busca el integer més gran dintre dels valors de tots els ítems. Necessari per calcular la similitud posteriorment.

- `private static Double compareItems(Item item1, Item item2, Double maxDouble, Integer maxInteger)`

Compara els dos ítems donats i retorna la seva distància.¹

- `private static Double dissimilaritySet(Item item1, Item item2)`

Calcula la dissimilitud dels atributs de tipus Set de dos items.

- `private static Double dissimilarityBoolean(Item item1, Item item2)`

Calcula la dissimilitud dels atributs de tipus Boolean de dos items.

- `private static Double dissimilarityInteger(Item item1, Item item2, Integer maxInteger)`

Calcula la dissimilitud dels atributs de tipus Integer de dos items.

- `private static Double dissimilarityDouble(Item item1, Item item2, Double maxDouble)`

Calcula la dissimilitud dels atributs de tipus Double de dos items.

- `private static Double dissimilarityString(Item item1, Item item2)`

Calcula la dissimilitud dels atributs de tipus String de dos items.

- Attribute:

Classe abstracta que identifica un atribut.

- Atributs:

- `private String attributeName;` Guarda el nom de l'atribut.

- Mètodes:

- `public Attribute(String name)`

Creador d'un atribut amb el nom.

- BooleanAttribute:

Classe que hereta d'Attribute. Identifica un atribut de tipus booleà.

- Mètodes:

- `public BooleanAttribute(String name)`

Creador d'un atribut booleà amb nom.

- DoubleAttribute:

Classe que hereta d'Attribute. Identifica un atribut de tipus double.

¹ La fórmula que defineix la distància entre dos ítems està explicada a l'apartat 8.2 Content Based Filtering

- Mètodes:
 - `public DoubleAttribute(String name)`

Creador d'un atribut double amb nom.

- **IntegerAttribute:**

Classe que hereta d'Attribute. Identifica un atribut de tipus integer.

- Mètodes:
 - `public IntegerAttribute(String name)`

Creador d'un atribut integer amb nom.

- **SetAttribute:**

Classe que hereta d'Attribute. Identifica un atribut de tipus set.

- Mètodes:
 - `public SetAttribute(String name)`

Creador d'un atribut set amb nom.

- **StringAttribute:**

Classe que hereta d'Attribute. Identifica un atribut de tipus string.

- Mètodes:
 - `public StringAttribute(String name)`

Creador d'un atribut string amb nom.

- **SetAttributes:**

Classe que emmagatzema el conjunt d'Atributs.

- Atributs:
 - `private ArrayList<Attribute> attributes;` Emmagatzema el conjunt d'atributs.

- Mètodes:
 - `public void addAttribute(Attribute attribute)`

Afegeix un Atribut al conjunt.

- `public ArrayList<String> getAttributesNames()`

Retorna el nom de tots els atributs del conjunt.

- **Value:**

Classe abstracta que representa el valor d'un atribut.

- Mètodes abstractes:

- `public abstract String discretize();`

Retorna el valor en format String.

- `public abstract double distance(String value);`

Retorna la distància entre dos valors del mateix tipus.

- **BooleanValue:**

Classe que hereta de Value. Representa un valor de tipus Boolean.

- Atributs:

- `private boolean value;` És el valor.

- `private BooleanAttribute booleanAttribute;` És l'atribut de tipus Boolean al qual pertany el valor.

- Mètodes:

- `public BooleanValue(String value, BooleanAttribute booleanAttribute)`

Creadora amb valor i atribut.

- `public String discretize();`

Retorna el valor en format String.

- `public double distance(String value);`

Retorna la distància entre dos valors de tipus Boolean.

- **DoubleValue:**

Classe que hereta de Value. Representa un valor de tipus Double.

- Atributs:

- `private Double value;` És el valor.

- `private DoubleAttribute doubleAttribute;` És l'atribut de tipus Double al qual pertany el valor.

- Mètodes:

- `public DoubleValue(String value, DoubleAttribute doubleAttribute)`

Creadora amb nom i atribut.

- `public String discretize();`

Retorna el valor en format String.

- `public double distance(String value);`

Retorna la distància entre dos valors de tipus Double.

- **IntegerValue:**

Classe que hereta de Value. Representa un valor de tipus Integer.

- Atributs:

- `private Integer value;` És el valor.
- `private IntegerAttribute integerAttribute;` És l'atribut de tipus Integer al qual pertany el valor.

- Mètodes:

- `public IntegerValue(String value, IntegerAttribute integerAttribute)`

Creadora amb valor i atribut.

- `public String discretize();`

Retorna el valor en format String.

- `public double distance(String value);`

Retorna la distància entre dos valors de tipus Integer.

- **SetValue:**

Classe que hereta de Value. Representa un valor Set.

- Atributs:

- `private String value;` És el valor.
- `private SetAttribute setAttribute;` És l'atribut de tipus Set al qual pertany el valor.

- Mètodes:

- `public SetValue(String value, SetAttribute setAttribute)`

Creadora amb valor i atribut.

- `public String discretize();`

Retorna el valor en format String.

- `public double distance(String value);`

Retorna la distància entre dos valors de tipus Set.

- `private static ArrayList<String> parseLine(String line, char delimiter)`

Operació privada que transforma un String delimitat per *delimiter* en un array d'Strings.

- **StringValue:**

Classe que hereta de Value. Representa un valor de tipus String.

- Atributs:

- `private String value;` És el valor.
- `private StringAttribute stringAttribute;` És l'atribut de tipus String al qual pertany el valor.

- Mètodes:

- `public StringValue(String value, StringAttribute stringAttribute)`

Creadora amb valor i atribut.

- `public String discretize();`

Retorna el valor en format String.

- `public double distance(String value);`

Retorna la distància entre dos valors de tipus String.

3. DIAGRAMA CAPA PRESENTACIÓ - Esquema

Diagrama UML

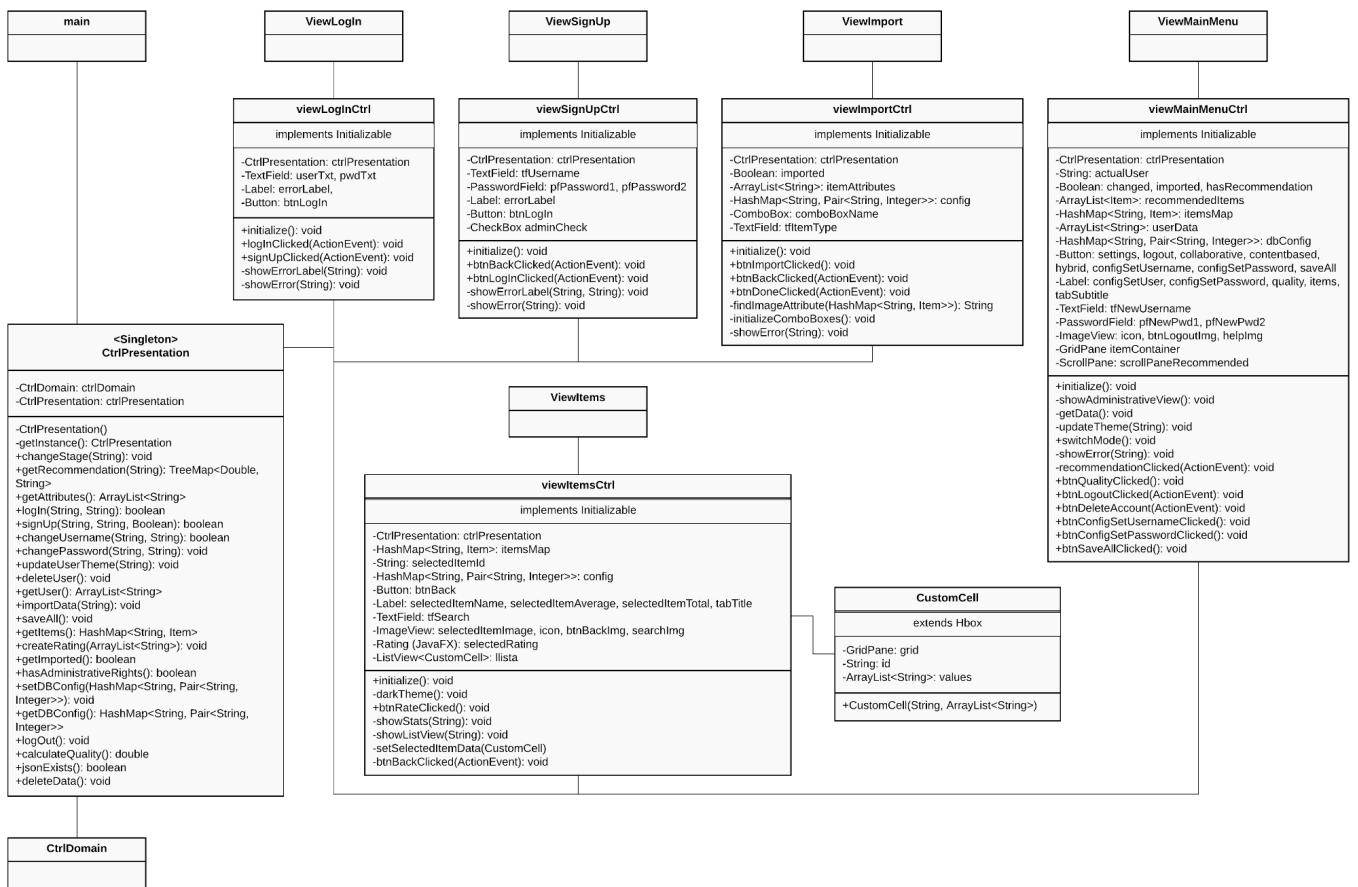


Diagrama de pantalles navegabilitat

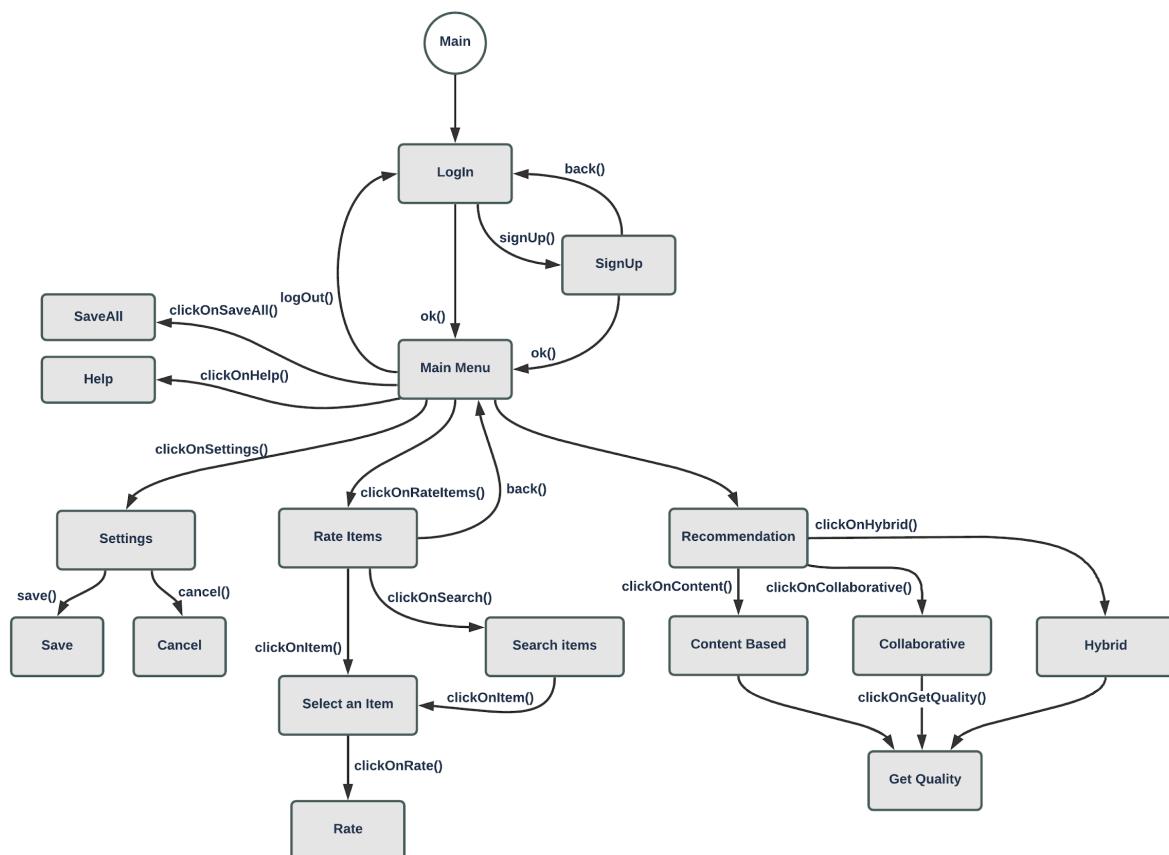
Aquest és el diagrama de navegabilitats del nostre sistema.

Des de la pantalla inicial de *Login*, podem anar a *SignUp* o al *MainMenu*.

Tota vista té implementada la funcionalitat de tornar a la pantalla anterior (excepte *Login* obviament).

Un cop al *MainMenu*, podem accedir al menú *Settings* i a la vista *Rate Items*.

Les funcionalitats de cada vista estan explicades de forma més detallada a continuació.

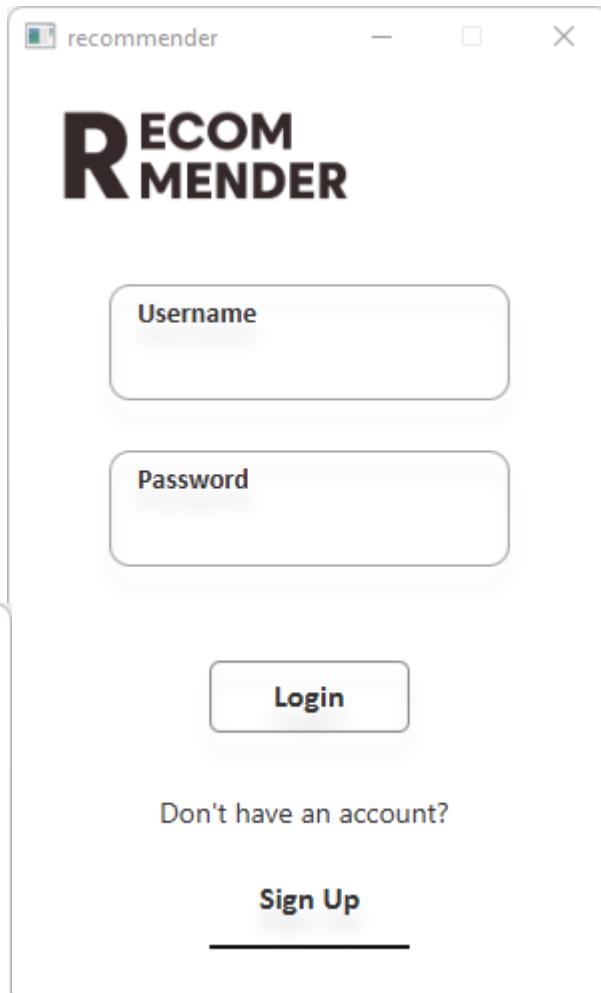
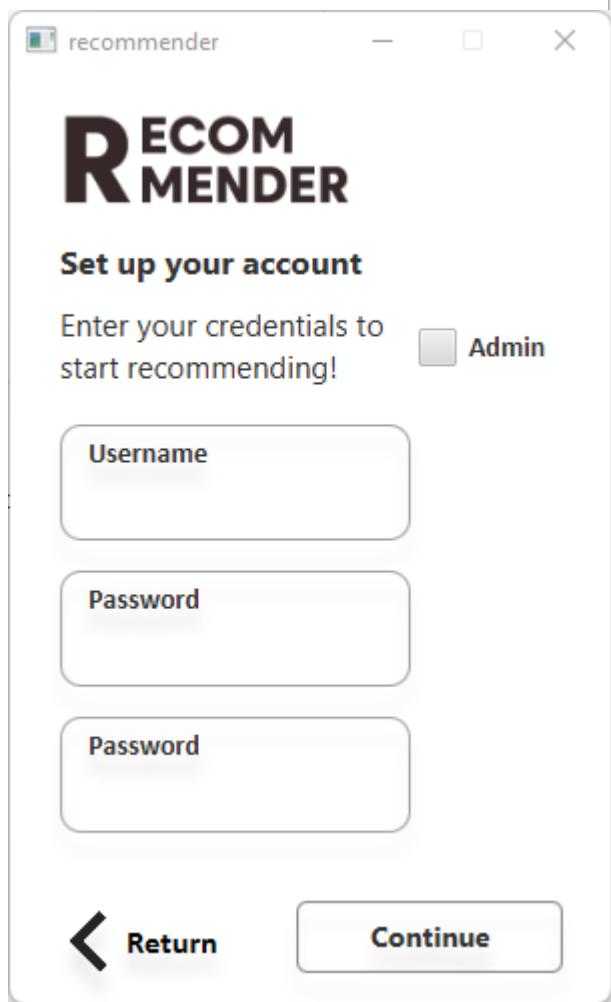


- Login:

Permet a l'usuari accedir al seu compte mitjançant l'usuari i la contrasenya.

Si no en té, pot demanar al sistema crear-ne un de nou mitjançant l'opció “Sign Up”.

En cas que les credencials proporcionades per l'usuari no siguin vàlides, el sistema remarca els camps incorrectes (poden ser-ho tots dos) i mostra el missatge d'error.



- SignUp:

El nou usuari emplena els camps necessaris per registrar el compte; el nom d'usuari, la contrasenya (amb una confirmació), i si té permisos d'administrador.

De nou, si es prem el botó “Continue” i algun d'aquests camps és incorrecte o no compleix amb les condicions del camp, el sistema ho fa saber a l'usuari mostrant un error i remarcant-ho.

- **Import:**

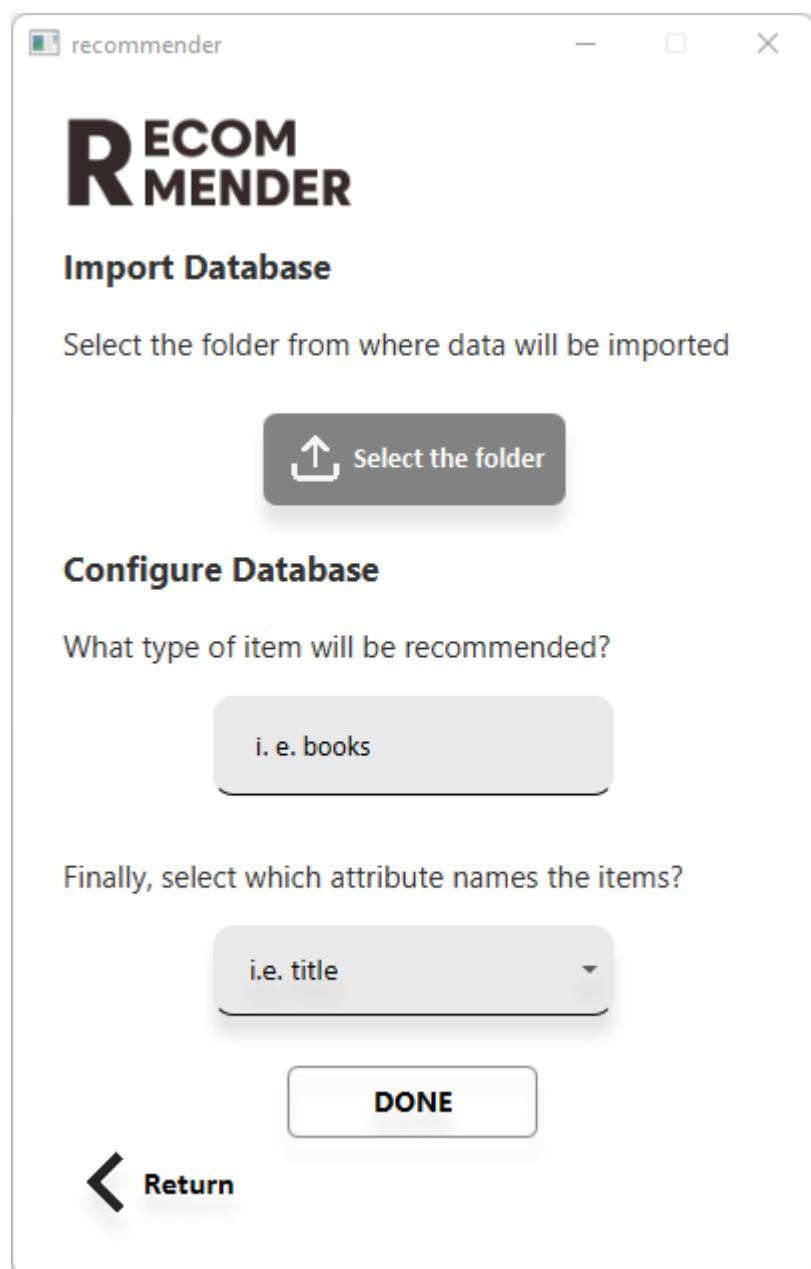
Aquesta vista només es mostra als usuaris administradors del sistema.

Quan s'ha de carregar la base de dades la primera vegada, aquesta pantalla es mostra a l'usuari administrador per tal de fer-ho.

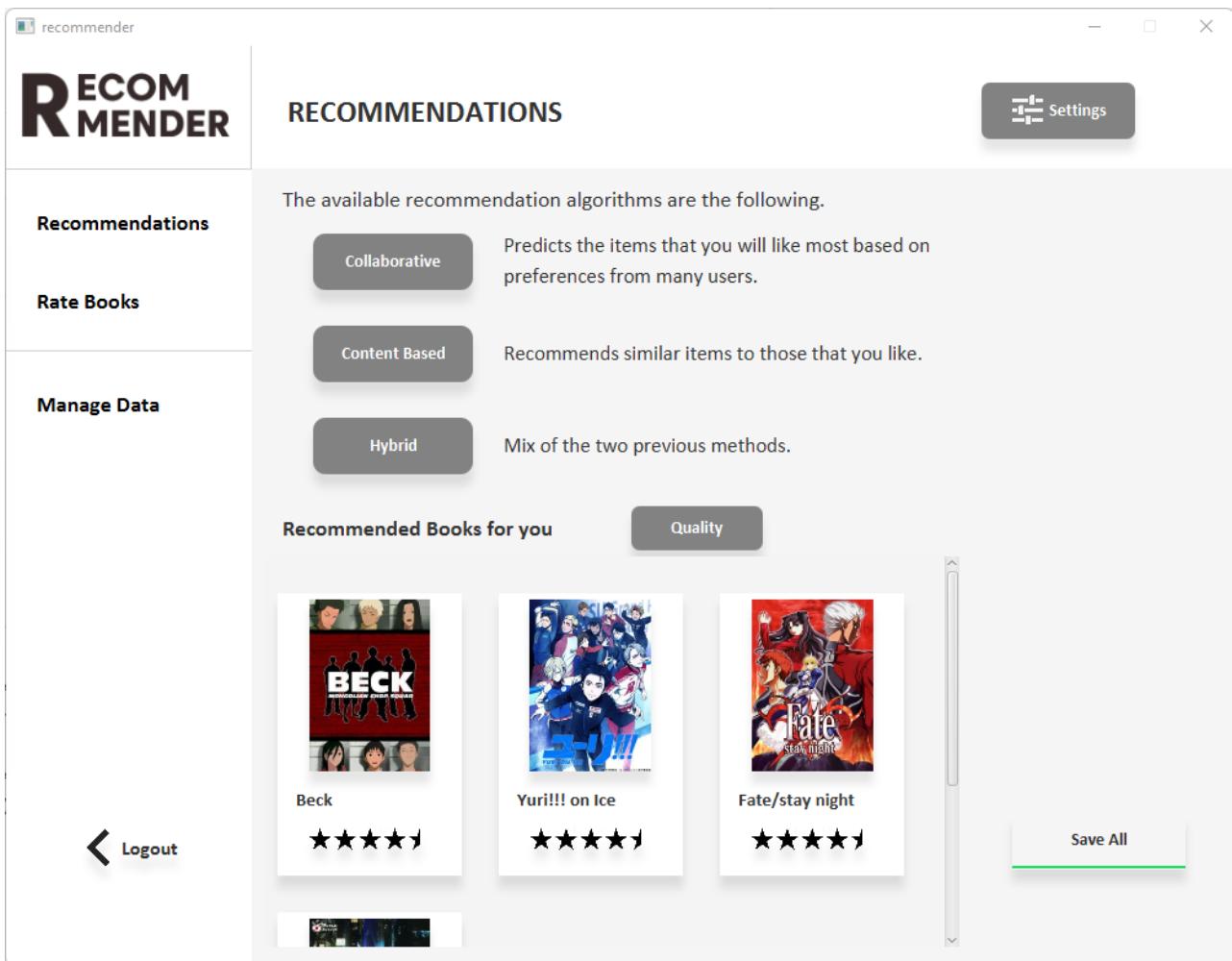
Un cop dins l'aplicació, es pot tornar a accedir a aquest menú per tal de modificar la base de dades o bé, carregar-ne una de nova.

Permet seleccionar el directori des d'on s'agafaran els fitxers.

A més, també es demana a l'usuari que introdueixi el nom del tipus d'ítem (i.e. series) i seleccioni, de tots els atributs dels ítems, quin és el que identifica l'ítem (i.e. títol sèrie).



- **MainMenu:**



Aquesta és la vista principal de l'aplicació (en aquest cas ja s'ha demanat una recomanació i la seva qualitat).

A la barra lateral de l'esquerra hi ha diverses opcions que porten a noves pantalles (la vista per defecte és “Recommendations”):

- Rate Items: Permet a l'usuari fer valoracions dels ítems.
- Manage Data: Aquesta opció només apareix als usuaris amb permisos d'administrador. Permet modificar la configuració de la base de dades.
- Logout: Permet a l'usuari sortir de la seva sessió.

A la part central es mostren les opcions per demanar una recomanació dependent de l'algorisme desitjat (*Collaborative, Content Based i Hybrid*).

Un cop s'ha demanat una recomanació, aquesta apareix a la part inferior de la pantalla, mostrant el títol o nom identificador de l'ítem, la imatge (si en té) i la valoració mitjana dels ratings que ha rebut l'ítem (en format estrelles).

Finalment, a la dreta hi ha dues opcions més:

- Settings: Obre un nou menú que permet a l'usuari canviar determinades característiques del seu perfil.
- SaveAll: Demana al sistema que guardi tots els canvis fets (valoracions, canvis a la base de dades i canvis a la configuració de l'usuari).
- **Rate Items:**

The screenshot shows the RECOMENDER application interface. On the left, there is a card for the movie "Zankyou no Terror" with a thumbnail, an average rating of 8.15, total ratings of 52, and a "RATE" button. On the right, there is a search bar labeled "Search an item" and a table listing various items with columns for id, title, synopsis, and genres. The table includes titles like "Katanagatarri", "Seishun Buta Yarou wa Bu...", "Rainbow: Nisha Rokubou ...", "Akame ga Kill!", "Gekkan Shoujo Nozaki-kun", "Psycho-Pass 2", "Zankyou no Terror", "Fate/stay night", "Code Geass: Hangyaku no...", "Koukyoushien Eureka Se...", "NHK ni Youkoso!", "Ore no Imouto ga Konnan...", "Baka to Test to Shoukanjuu", "Kami nomi zo Shiru Sekai", "Sora yori mo Too! Basho", "Fate/Zero 2nd Season", "Shokugeki no Souma", "Gintama", "Gosick", "Mahou Shoujo Madoka★...", and "Haikyuu!!!".

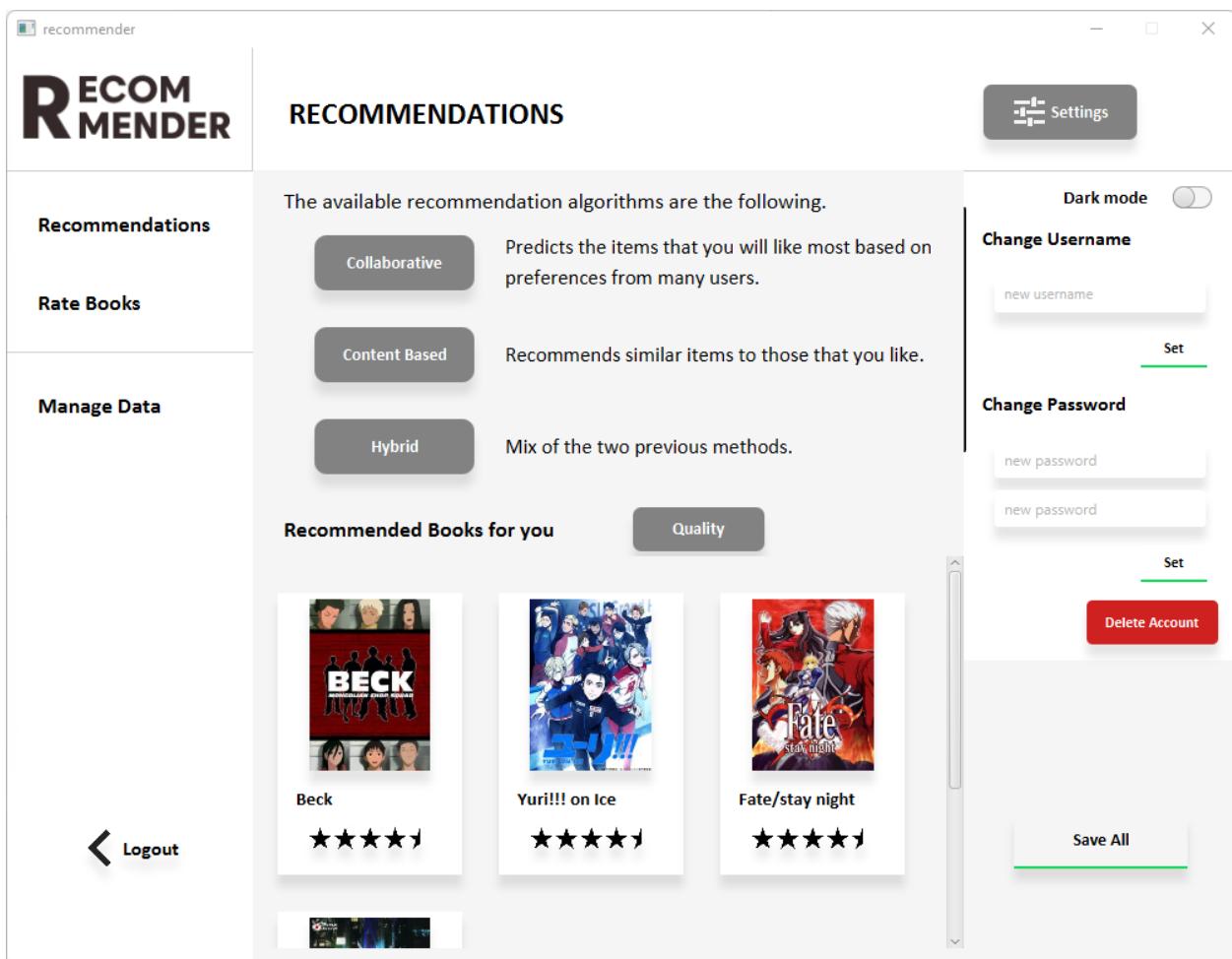
id	title	synopsis	genres
6594	Katanagatarri	In an Edo-era Japan lush ...	Action;Adventure
37450	Seishun Buta Yarou wa Bu...	The rare and inexplicable ...	Comedy;Romance
6114	Rainbow: Nisha Rokubou ...	Japan 1955: Mario Minaka...	Drama;Historical
22199	Akame ga Kill!	Night Raid is the covert as...	Action;Adventure
23289	Gekkan Shoujo Nozaki-kun	Chiyo Sakura is a cheerful ...	Comedy;Romance
23281	Psycho-Pass 2	A year and a half after the...	Action;Sci-Fi;Poli...
23283	Zankyou no Terror	Painted in red the word V...	Mystery;Psycholog...
356	Fate/stay night	After a mysterious inferno...	Action;Supernatu...
1575	Code Geass: Hangyaku no...	In the year 2010 the Holy ...	Action;Military;Sci...
237	Koukyoushien Eureka Se...	In the backwater town of ...	Adventure;Drama
1210	NHK ni Youkoso!	Twenty-two-year-old colle...	Comedy;Psycholog...
8769	Ore no Imouto ga Konnan...	Kirino Kousaka embodies ...	Slice of Life;Com...
6347	Baka to Test to Shoukanjuu	Fumizuki Academy isn't a ...	Comedy;Romance
8525	Kami nomi zo Shiru Sekai	Keima Katsuragi known o...	Comedy;Harem;f...
35839	Sora yori mo Too! Basho	Filled with an overwhelmi...	Adventure;Come...
11741	Fate/Zero 2nd Season	As the Fourth Holy Grail ...	Action;Supernatu...
28171	Shokugeki no Souma	Ever since he was a child fi...	Ecchi;School;Sho...
918	Gintama	The Amanto aliens from o...	Action;Comedy;H...
8425	Gosick	Kazuya Kujou is a foreign ...	Mystery;Historical
9756	Mahou Shoujo Madoka★...	Madoka Kaname and Saya...	Psychological;Dr...
20583	Haikyuu!!!	Inspired after watching a v...	Comedy;Sports;F...

Aquesta vista mostra a l'usuari la llista d'ítems de la base de dades. L'usuari té l'opció de buscar ítems tant per l'id com per l'atribut identificador (en aquest cas, l'administrador al configurar la base de dades ha dit que era *title*).

L'usuari pot clicar a l'ítem desitjat, i a l'esquerra, apareixerà un quadre amb el nom de l'ítem, la imatge, les estadístiques associades a l'ítem (valoració mitjana i nombre total de valoracions) i unes estrelles que permeten valorar l'ítem en una escala de 0 a 5 (la valoració s'escala a 10 per seguir amb els rangs de valoracions definits per a la base de dades).

A la part inferior esquerra de la pantalla hi ha l'opció de tornar enrere (MainMenu).

- **Settings:**



Si l'usuari fa clic al botó “Settings” de la part superior dreta, un nou menú es desplegarà des del costat dret, mostrant diverses opcions:

- Dark Mode²: Permet a l'usuari seleccionar si vol el tema fosc i clar.
- Change Username: L'usuari pot canviar el seu nom d'usuari des d'aquí.
- Change Password: L'usuari pot canviar la seva contrasenya (introduint també la contrasenya actual).

² A l'Annex (punt 9), hi ha les vistes en mode fosc.

- Delete Account: Si l'usuari prem el botó, apareix un diàleg perquè l'usuari confirmi l'operació. Si ho confirma de nou, el compte s'eliminarà (no les seves valoracions) i tornarà al menú inicial (login).

Per tancar el menú, simplement s'ha de fer clic fora del quadre de les opcions.

4. DIAGRAMA CAPA PRESENTACIÓ - Explicació

Per programar la interfície gràfica s'ha decidit utilitzar les llibreries JavaFX i ControllerFX, basades en Swing.

- **CtrlPresentation:**

Aquest controlador gestiona tot la capa de presentació.

- `public void changeStage(String path)`

Canvia el fitxer .fxml que es mostra en pantalla per l'identificat per *path*.

- `public TreeMap<Double, String> getRecommendation(String algorithmName, int k)`

Demana al controlador de domini la recomanació.

- `public ArrayList<String> getAttributes()`

Demana al controlador de domini els atributs dels ítems.

- `public boolean logIn(String username, String pwd)`

Demana al controlador de domini que iniciï la sessió de l'usuari identificat per les credencials donades (si aquestes pertanyen a un usuari i són vàlides).

- `public boolean signUp(String username, String pwd, Boolean admin)`

Demana al controlador de domini que creï un nou usuari amb les dades donades.

- `public boolean changeUsername(String newUsername, String oldUsername)`

Demana al controlador de domini que canviï el *username* de l'usuari actiu per un de nou.

- `public void changePassword(String newPassword, String username)`

Demana al controlador de domini que canviï el *password* de l'usuari actiu per un de nou.

- `public void deleteUser()`

Demana al controlador de domini que elimini l'usuari actiu dels registres (els seus ratings es mantenen).

- `public ArrayList<String> getUser()`

Demana al controlador de domini les dades de l'usuari actiu en format array.

- `public void importData(String path)`

Demana al controlador de domini que importi les dades des del directori indicat per *path*.

- `public void saveAll()`

Demana al controlador de domini que desi tots els canvis realitzats per l'usuari.

- `public HashMap<String, Item> getItems()`

Demana al controlador de domini el conjunt d'ítems.

- `public void updateUserTheme(String theme)`

Demana al controlador de domini que actualitzi el theme de l'usuari actiu per *theme*.

- `public void createRating(ArrayList<String> rating)`

Demana al controlador de domini que creï una nova valoració amb l'array donat.

- `public boolean getImported()`

Demana al controlador de domini si les dades han estat importades.

- `public boolean hasAdministrativeRights()`

Demana al controlador de domini si l'usuari actiu té permisos d'administrador (necessaris per modificar la configuració de la base de dades).

- `public void setDBConfig(HashMap<String, Pair<String, Integer>> config)`

Demana al controlador de domini que actualitzi el fitxer de configuració de la base de dades.

- `public HashMap<String, Pair<String, Integer>> getDBConfig()`

Demana al controlador de domini la configuració de la base de dades.

- `public void logOut()`

Demana al controlador de domini que tanqui la sessió de l'usuari actiu.

- `public double calculateQuality()`

Demana al controlador de domini que calculi la qualitat d'una recomanació (només obtenen un resultat real els usuaris definits a ratings.test.unknown.csv).

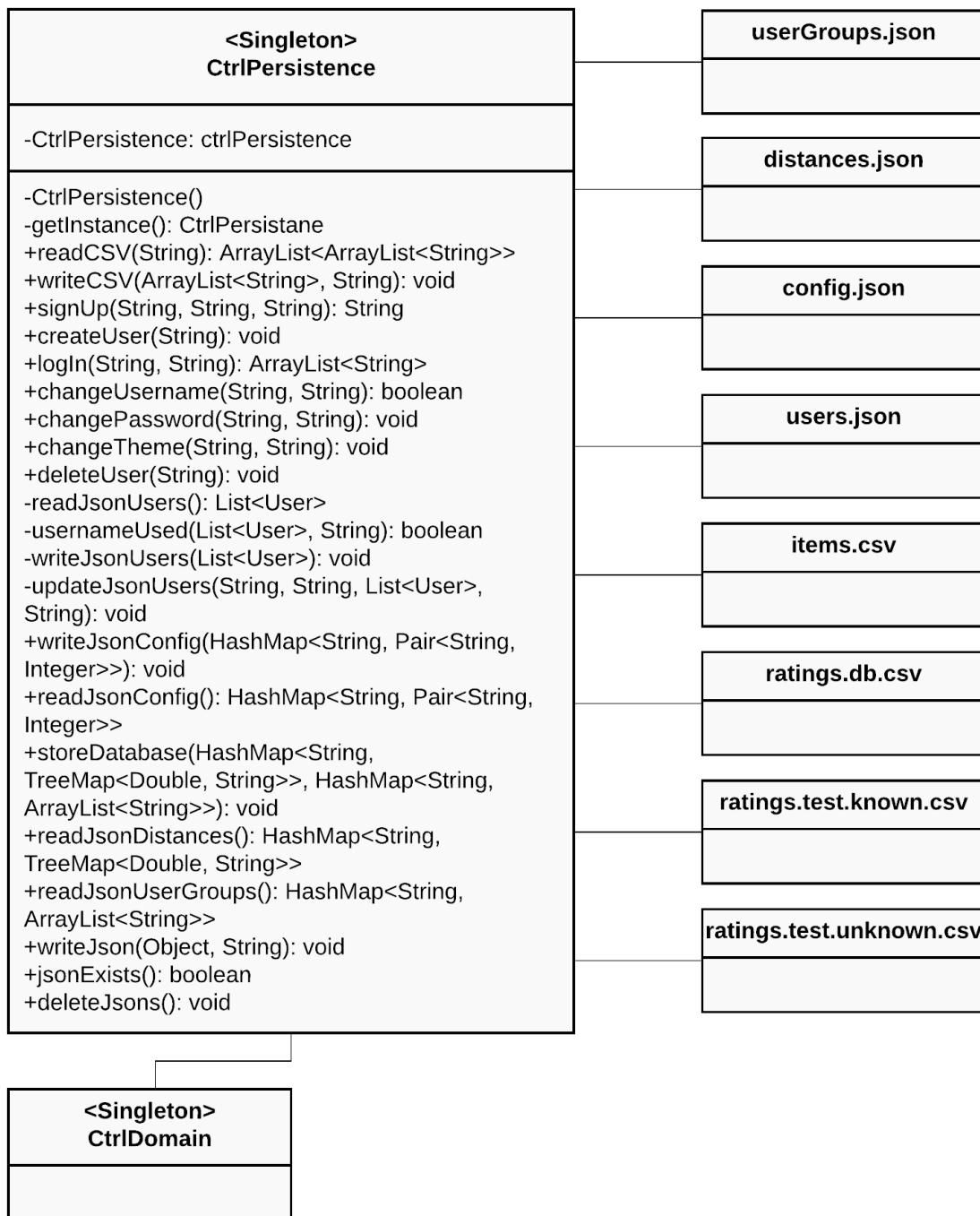
- `public boolean jsonExists()`

Demana al controlador de domini si existeixen els fitxers .json amb les dades preprocessades (necessari per saber si s'ha carregat la base de dades).

- `public void deleteData()`

Demana al controlador de domini que elimini els fitxers amb les dades preprocessades (config.json, distances.json i usergroups.json).

5. DIAGRAMA CAPA PERSISTÈNCIA - Esquema



6. DIAGRAMA CAPA PERSISTÈNCIA - Explicació

Per emmagatzemar dades en determinats fitxers .json s'ha fet servir la llibreria Gson.

- **CtrlPersistence:**

Hem optat per fer un únic controlador que gestió l'entrada i sortida de dades de tota la base de dades.

Per emmagatzemar les dades, s'utilitzen diversos fitxers:

- distances.json: Guarda les distàncies entre els ítems, de tal forma que aquestes només es calculen quan es carrega la base de dades per primera vegada.
- usergroups.json: Guarda els grups d'usuaris calculats. S'ha d'anar actualitzant.
- users.json: Emmagatzema els usuaris i les seves dades (id, username, password, admin?, theme i userGroup).
- config.json: Guarda la configuració de la base de dades que l'administrador assigna a l'importar les dades.

Els mètodes del controlador són els següents:

- `public ArrayList<ArrayList<String>> readCSV(String filename)`

Llegeix el fitxer .csv especificat per *filename* i retorna la matriu amb les dades.

- `public static void writeCSV(ArrayList<String> row, String filename)`

Afegeix al fitxer .csv especificat per *filename* un fila amb les dades donades.

- `public String signUp(String username, String pwd, Boolean admin)`

Comprova si al fitxer users.json existeix un usuari amb aquest nom, si no ho fa, busca un identificador vàlid i li assigna. Crea l'usuari amb els paràmetres donats i retorna l'identificador.

- `public void createUser(String id)`

Crea i escriu al fitxer users.json un usuari amb *username* i *id* iguals i *password* "1234". Aquest usuari és dels que es llegeixen de ratings.db.csv i

ratings.test.known.csv, i és per això que no té ni nom usuari ni contrasenya “definits”.

- `public ArrayList<String> logIn(String username, String pwd)`

Comprova si al fitxer users.json existeix un usuari amb les credencials donades, si ho fa, retorna les seves dades.

- `public boolean changeUsername(String newUsername, String oldUsername)`

Actualitza el nom d’usuari de l’usuari donat i crida updateJsonUsers amb els paràmetres adequats.

- `public void changePassword(String newPassword, String username)`

Actualitza la contrasenya de l’usuari donat i crida updateJsonUsers amb els paràmetres adequats.

- `public void changeTheme(String username, String newTheme)`

Actualitza el tema de l’usuari donat i crida updateJsonUsers amb els paràmetres adequats.

- `public void deleteUser(String username)`

Elimina l’usuari identificat pel *username* donat del fitxer users.json.

- `private static List<User> readJsonUsers()`

Llegeix el fitxer users.json i el retorna en una llista d’usuaris.

- `private static boolean usernameUsed(List<User> users, String username)`

Comprova si ja existeix un usuari amb el *username* donat. Retorna el resultat.

- `private static void writeJsonUsers(List<User> users)`

Escriu el fitxer users.json amb la llista d’usuaris passada per paràmetre.

- `private static void updateJsonUsers(String newValue, String username, List<User> users, String field)`

Actualitza l’atribut (*field*) de l’usuari identificat per *username* amb un nou valor.

- `public void writeJsonConfig(HashMap<String, Pair<String, Integer>> config)`

Escriu el fitxer config.json amb les dades passades pel paràmetre.

- `public HashMap<String, Pair<String, Integer>> readJsonConfig()`

Llegeix el fitxer config.json i el retorna en un `HashMap` amb l'identificador de cada valor, i un parell amb les dades.

- `public void storeDatabase(HashMap<String, TreeMap<Double, String>> distances, HashMap<String, ArrayList<String>> groupUsers)`

Escriu els fitxers distances.json i usergroups.json amb els paràmetres donats.

- `public HashMap<String, TreeMap<Double, String>> readJsonDistances()`

Llegeix el fitxer distances.json i el retorna en un `TreeMap` amb l'identificador de cada ítem, i les distàncies amb la resta dels ítems ordenades de forma decreixent.

- `public HashMap<String, ArrayList<String>> readJsonUserGroups()`

Llegeix el fitxer usergroups.json i el retorna en un `HashMap` amb l'identificador del grup, i els seus integrants (en un array).

- `private static void writeJson(Object obj, String filename)`

Escriu el fitxer .json especificat pel paràmetre `filename` amb l'objecte donat.

- `public boolean jsonExists()`

Comprova si els fitxers .json existeixen i retorna el resultat.

- `public void deleteJsons()`

Elimina els fitxers .json emmagatzemats.

7. DESCRIPCIÓ ESTRUCTURES DE DADES

Estructures principals utilitzades:

- Recomanació:

L'estructura de dades escollida per emmagatzemar una **recomanació** és un TreeMap en ordre invers (decreixent) on la 'key' és el valor de la valoració i el 'value' és l'identificador de l'ítem.

```
TreeMap<Double, String> recommendation;
```

Aquesta implementació ens evita haver d'ordenar el conjunt d'ítems recomanats constantment. Això fa que en termes d'eficiència, el TreeMap es quedi enrere en comparació amb HashMap's i LinkedHashMap's, però com tampoc haurà d'emmagatzemar quantitats immenses de dades, la pèrdua en eficiència compensa l'ordenació automàtica.

- Conjunt de valoracions:

Per tal de guardar tots els ratings de forma que aquests després siguin fàcilment accessibles, s'ha decidit implementar el conjunt de la següent manera:

```
HashMap<String, ArrayList<Rating>> ratings;
```

De forma que cada usuari té un *ArrayList<Rating>*. Això ens permet buscar fàcilment els ratings associats a un usuari, condició capital per a poder executar els algorismes de recomanació eficientment.

8. DESCRIPCIÓ FUNCIONALITATS PRINCIPALS

8.1. Collaborative Filtering

L'estratègia del Collaborative Filtering es basa a recomanar a l'usuari aquells ítems que han agradat a usuaris semblants. Està implementada en dues parts:

a. K-Means:

L'algoritme necessita els següents paràmetres:

- El nombre de centroides inicial.
- El nombre màxim d'iteracions que pot realitzar l'algoritme.
- El *dataset* sobre el qual aplicar l'algoritme.

Primerament, se seleccionen els centroides iniciais escollint aleatoriament K usuaris, això garanteix que inicialment no hi hagi cap partició buida. Aquesta part de l'algoritme té un cost $O(k)$.

A partir d'aquí se segueixen els mateixos passos fins que l'algoritme finalitza (arribem al màxim d'iteracions o les particions s'estabilitzen). Per cada *datapoint* (usuari) es calcula de quin dels K centroides es troba més a prop fins que tots es troben assignats a un clúster. Això té un cost $O(K * n * m)$, on n és el número de datapoints i m el nombre d'atributs (ítems valorats) per usuari.

A continuació es recalculen els centroides fent la mitjana de tots els *datapoints* de cada clúster i repetim fins a acabar l'algoritme. Aquest últim pas té un cost proporcional a $O(n * m)$.

b. SlopeOne:

La funció rep diversos paràmetres:

- L'identificador de l'usuari que vol rebre una recomanació.
- L'identificador de l'ítem pel qual es vol fer la predicción.

A més a més, també s'accedeix al conjunt de valoracions a través del Controlador de Domini.

El primer pas és recórrer aquells ítems que ha valorat l'usuari actiu i, si no són l'ítem pel qual s'està fent la predicción, calculem la desviació mitjana i la quantitat de valoracions. Per fer-ho, s'accedeix a les valoracions d'aquells usuaris que pertanyen al grup d'usuaris de l'usuari actiu i que hagin valorat tots dos ítems.

Per cada desviació, aquesta es divideix entre el nombre de valoracions, se suma a la valoració que ha fet l'usuari actiu a aquest segon ítem i es multiplica per n.

Finalment, les desviacions acumulades es divideixen entre les ns acumulades, obtenint així la predicción que faria l'usuari actiu a un determinat ítem.

8.2. Content Based Filtering

L'estrategia implementada està basada en l'algorisme “k-nearest neighbours (k-NN)”. Bàsicament, estudia quins són els ítems més semblants als que el nostre User actiu ha valorat positivament. Després d'haver recorregut tot el conjunt d'ítems, només ens quedarem amb els k ítems més similars.

Per tant, el primer que fem és, treballar amb dos *HashMaps*, un que representa el conjunt d'ítems que li agraden a l'usuari actiu i l'altre que representa el conjunt d'ítems no valorats per l'usuari.

La funció, per tant, es basa a fer un doble bucle, on comparem cada ítem que li agrada a l'usuari per cada ítem de la base de dades no valorat. En aquesta comparació ens quedem amb tots els ítems i la seva puntuació, la qual indica més probabilitat de ser recomanat com més propera a 0. Les següents iteracions fem el mateix, però de manera que actualitzem les puntuacions dels ítems. Aquesta puntuació es calcula a partir de la distància entre els atributs de l'ítem i de la valoració mitjana complementaria.

Pondera per ser concrets un 60% la distància i un 40% la valoració mitjana complementària. El càlcul de la valoració mitjana complementària es fa de la següent manera: agafem la valoració mitjana de l'ítem i la restem del valor màxim que pot tenir una valoració (que és 5 en el nostre cas). Ens interessa més recomanar l'ítem com més petit sigui el valor que estem calculant. És per això que en comptes d'agafar la valoració mitjana original, agafem la complementària (valoració mitjana complementària = valoració mitjana màxima (5) - valoració mitjana)

El càlcul de la distància (entre atributs dels ítems) es fa de la següent manera: nosaltres sabem que els atributs d'un ítem poden ser de 5 tipus diferents

(Boolean, String, Integer, Double i Set). El que fem quan comparem la distància de dos items és calcular la distància dels valors d'aquest ítem. Els valors booleans i strings, els comparem de forma que si són diferents incrementa la distància en una unitat i si són iguals no toquem la distància. Pel que respecta als valors enters i double, el que fem és la diferència entre els 2 valors que es comparen, si són iguals, la diferència és 0 i, per tant, la distància no incrementa. En canvi, a mesura que incrementa la seva diferència ho fa de mateixa manera la distància. Fem servir posteriorment, el valor màxim que pot tenir aquesta diferència per normalitzar aquest valor i que ens quedí sobre 1. Això ho fem amb el càlcul de totes les distàncies de tots els atributs, els normalitzem. La distància dels valors tipus set, es calculen de la mateixa manera que els valors tipus string però comparant un per un cada element dels dos sets, i al final es normalitza el valor sabent el nombre total de strings que conté el set.

Finalment, cada distància calculada de cada tipus d'atribut pondera un $\frac{1}{6}$ sobre 1. Es sumen les distàncies amb la seva ponderació i ja tenim la distància entre els atributs de dos ítems. Aquest valor calculat, reiterem que pondera un 60% i la valoració mitjana complementària explicada anteriorment un 40%. Així és com obtenim el valor per cada ítem no valorat per l'usuari, i com més petit sigui més ens interessa recomanar-li.

Les següents iteracions fem el mateix, però de manera que actualitzem les puntuacions dels ítems.

Tots els ítems que hem introduït a la llista ara tenen una puntuació associada. I, com a resultat, l'únic que hem de fer és quedar-nos amb els k elements de la llista amb menys puntuació. Que seran els més similars i amb valoració mitjana més elevada.

8.3. Hybrid Approaches

L'estratègia del Hybrid Approaches és simple, recomanem els millors ítems segons l'algorisme Collaborative i els millors segons el Content Based. Després fem la unió d'aquestes dues recomanacions en funció de la rellevància dels ítems recomanats.

8.4. Avaluació Recomanacions

La funció rep dos paràmetres:

- Una llista amb la recomanació calculada.
- Una llista amb la recomanació real.

L'algorisme recorre els ítems de la recomanació calculada, i per a cada un d'ells busca la posició que aquest ocupa en la recomanació real.

Si aquest ítem pertany a la recomanació real, s'obté la valoració associada, altrament, la rellevància pren com a valor 0.

A partir d'aquí, s'aplica la fórmula del Discounted Cumulative Gain tenint en compte que, si dos ítems tenen la mateixa valoració, tots dos ocupen la mateixa posició dintre de la recomanació, i el següent ítem de la llista també estarà en una posició inferior (més rellevància).

El nostre sistema no calcula el Normalized Discounted Cumulative Gain.

9. REPARTICIÓ CLASSES

Capa de Domini:

- Algorithm: Abdelali El Attar Yalaoui
- Attribute: Abdelali El Attar Yalaoui
- SetAttributes: Miguel García Soler
- BooleanAttribute: Miguel García Soler
- DoubleAttribute: Abdelali El Attar Yalaoui
- IntegerAttribute: Sergi Casau Pueyo
- StringAttribute: Sergi Casaú Pueyo
- SetAttribute: Sergi Casau Pueyo
- Value: Sergi Casau Pueyo
- BooleanValue: Abdelali El Attar Yalaoui
- DoubleValue: Abdelali El Attar Yalaoui
- IntegerValue: Abdelali El Attar Yalaoui
- StringValue: Abdelali El Attar Yalaoui
- SetValue: Miguel García Soler
- Collaborative Filtering: Miguel García Soler
- KMeans: Miguel García Soler
- SlopeOne: Ricard Guixaró Tranco
- Content Based: Sergi Casau Pueyo
- Hybrid Approaches: Miguel García Soler
- Item: Ricard Guixaró Tranco
- SetItems: Ricard Guixaró Tranco
- User: Ricard Guixaró Tranco
- UserGroup: Ricard Guixaró Tranco
- SetUserGroups: Miguel García Soler
- Quality: Ricard Guixaró Tranco
- Rating: Sergi Casau Pueyo
- SetRatings: Sergi Casau Pueyo
- Statistics: Sergi Casau Pueyo
- Recommendation: Abdelali El Attar Yalaoui

- CtrlDomain: Ricard Guixaró Tranco
- CtrlDomainUser: Miguel García Soler

Capa de Presentació:

- CtrlPresentation: Ricard Guixaró Tranco
- main: Sergi Casau Pueyo
- ItemController: Miguel García Soler
- ViewImportCtrl: Ricard Guixaró Tranco
- ViewItemsCtrl: Ricard Guixaró Tranco
- ViewLoginCtrl: Ricard Guixaró Tranco
- View Main Menu Ctrl: Ricard Guixaró Tranco

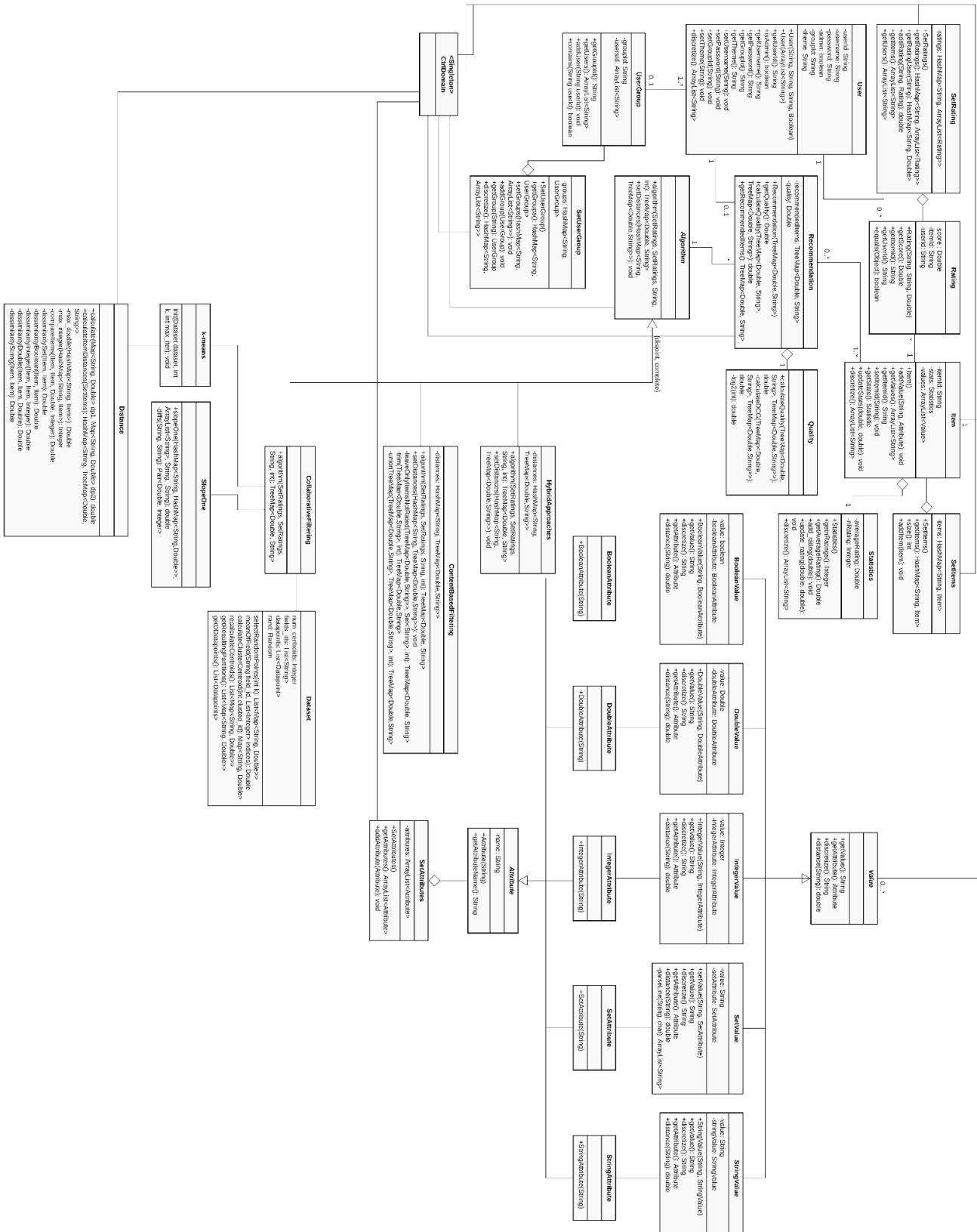
- ViewSignUpCtrl: Ricard Guixaró Trancho

Capa de Persistència:

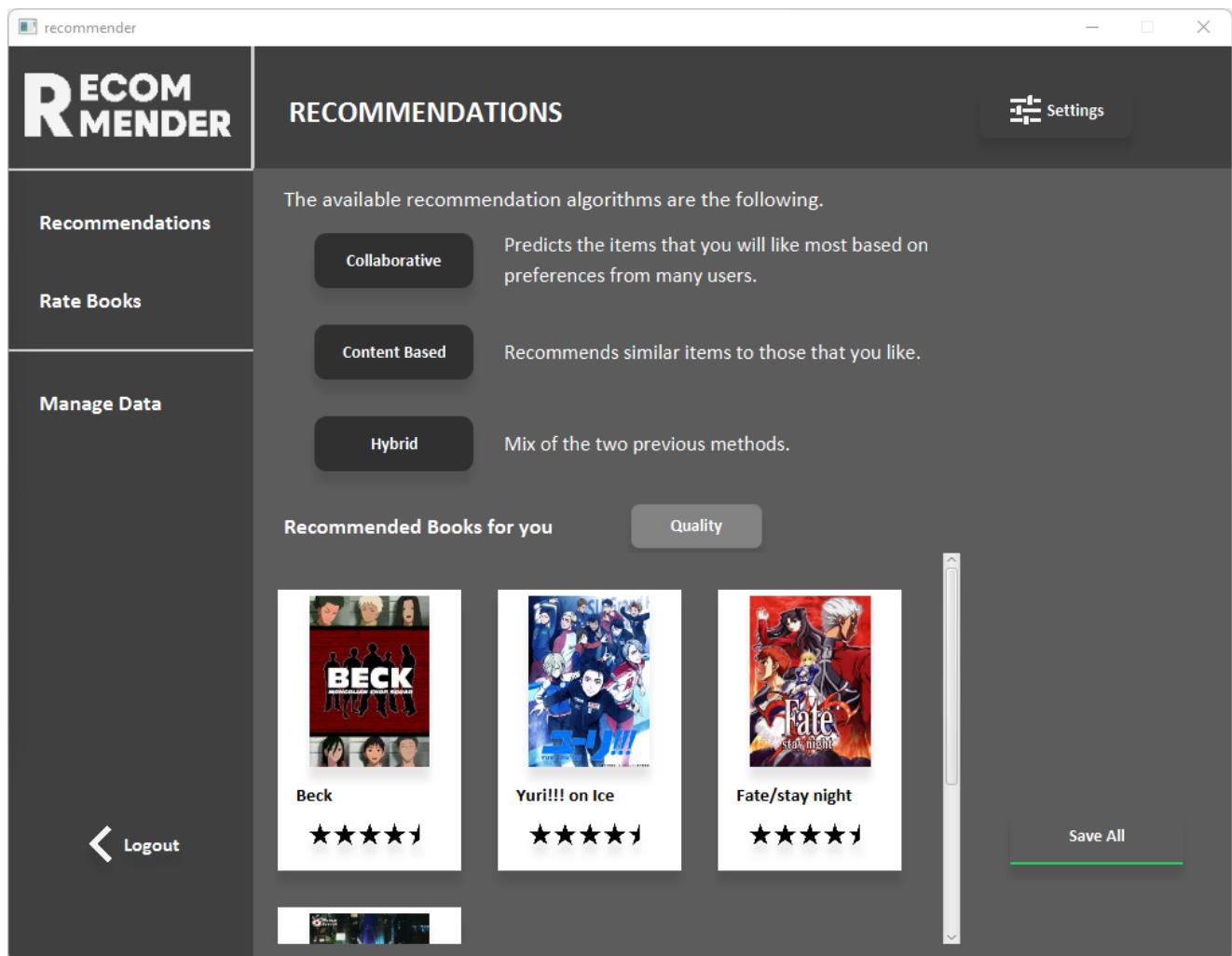
- CtrlPersistence: Ricard Guixaró Trancho

10. ANNEX

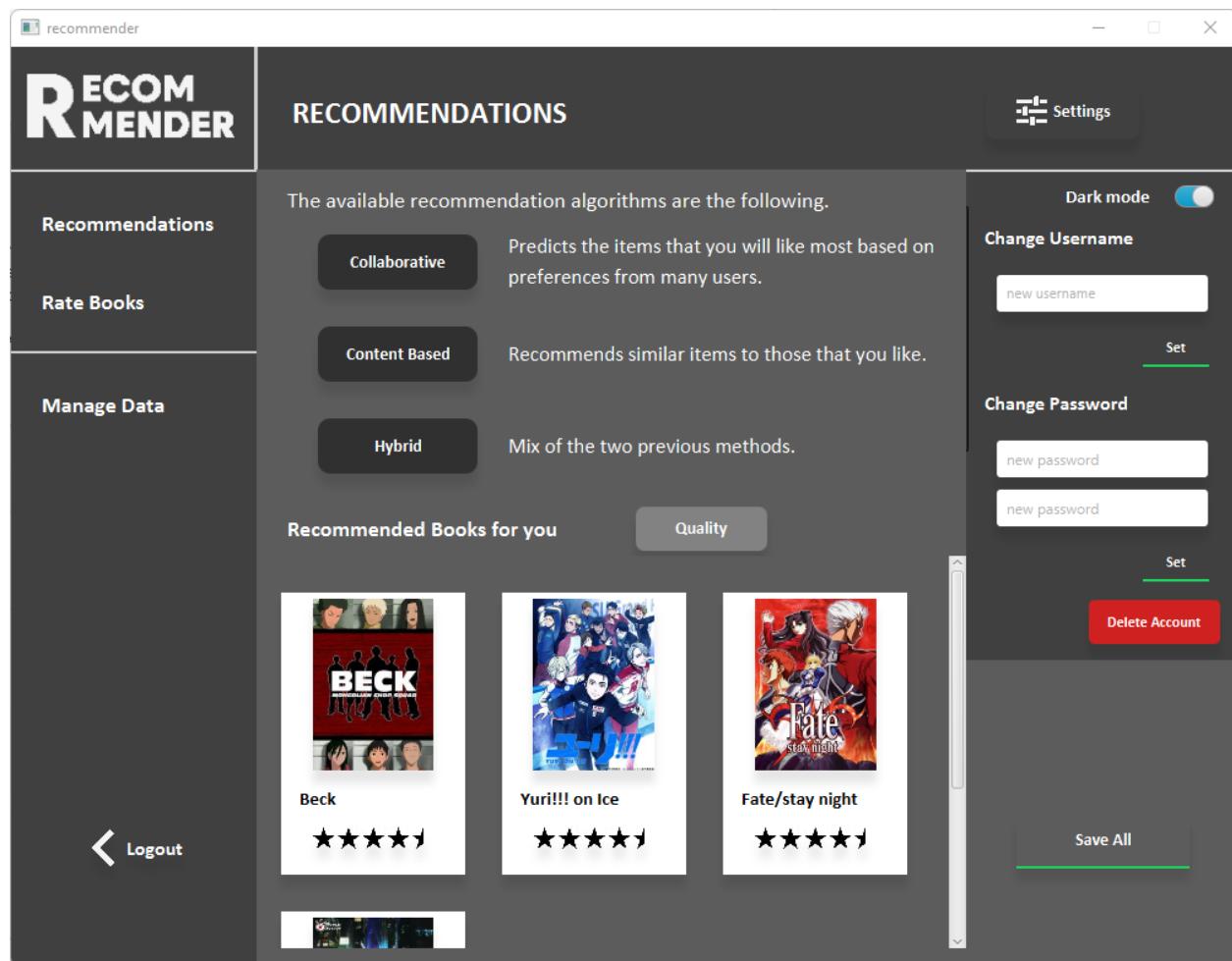
- Diagrama de classe de la capa de Domini:



- Vista en mode fosc del MainMenu de l'aplicació.



- Vista en mode fosc de les opcions de configuració:



- Vista en mode fosc de la pantalla de valoració d'ítems,

The screenshot shows a desktop application window titled "recommender". On the left, there's a dark-themed panel for "Sword Art Online II" with a thumbnail, average rating (6.67), total ratings (57), and a star rating UI. A "RATE" button is at the bottom. On the right, a table lists books with columns for ID, Title, Description, and Genres. A search bar is at the top of the table.

	Title	Description	Genres
30015	ReLIFE	Dismissed as a hopeless lo...	Slice of Life;Rom...
11759	Accel World	Haruyuki Arita is an overw...	Action/Game;Sci-
245	Great Teacher Onizuka	Onizuka is a reformed bik...	Comedy;Drama;S...
6	Trigun	Vash the Stampede is the ...	Action;Sci-Fi;Adv...
249	InuYasha (TV)	Based on the Shogakukan...	Action/Adventure
15315	Mondaiji-tachi ga Isekai k...	Izayoi Sakamaki Asuka Ku...	Action/Comedy;F...
14467	K	Kings are individuals who ...	Action;Super Pov...
16524	Suisei no Gargantia	In the distant future a maj...	Action;Sci-Fi;Adv...
9989	Ano Hi Mita Hana no Nam...	Jinta Yadomi is peacefully ...	Slice of Life;Sup...
34618	Blend S	Wishing to be independe...	Slice of Life;Com...
14345	Btooom!	Ryouta Sakamoto is unem...	Action;Psycholog...
32438	Mayoiga	A bus full of eccentric indi...	Mystery;Comedy
20	Naruto	Moments prior to Naruto ...	Action/Adventure
21	One Piece	Gol D. Roger was known a...	Action/Adventure
8795	Panty & Stocking with Gar...	The Anarchy Sisters Panty ...	Action;Comedy;P...
15085	Amnesia	After fainting at work a yo...	Harem;Mystery;R...
21881	Sword Art Online II	A year after escaping Swo...	Action/Game;Adv...
11285	Black★Rock Shooter (TV)	On the first day of junior h...	Action;Drama;Sci...
17265	Log Horizon	In the blink of an eye thirt...	Action/Game;Adv...
22297	Fate/stay night: Unlimited ...	The Holy Grail War is a ba...	Action;Fantasy;M...
34280	Gamers!	Keita Amano is a typical hi...	Comedy;Romance