Projecte de programació - Entrega 2.0

Codi del projecte: 12.2
Codi del clúster: 2.2

Arnau Badía Sampera (*arnau.badia.sampera@est.fib.upc.edu*)
Carla Claverol González (*carla.claverol@est.fib.upc.edu*)
Carlos Lázaro Costa (*carlos.lazaro.costa@est.fib.upc.edu*)
Guillem Castro Olivares (*guillem.castro@est.fib.upc.edu*)

Índex

Part comuna

Diagrama de classes compartides

Enllaç: http://i.imgur.com/VyiomEP.png

Especificació detallada de classes compartides

Node

*classe **abstracta**, implements **Serializable***
Defineix un node del graf.

Atributs:

- *protected* **nom** : String
- *protected* **id** : Integer
- *protected* **label** : String

Mètodes:

- **Node**(): Constructor per defecte.
- **Node**(int id, String nom): Constructor amb id i nom.
- **Node**(int id, String nom, String label): Constructor amb id, nom i label.

- void **setLabel**(String label): Modificador de label.
- void **setNom**(String nom): Modificador de nom.
- void **setId**(int id): Modificador de id.

- String **getLabel**(): Consultor de label.
- String **getNom**(): Consultor de nom.
- int **getId**(): Consultor d'id.

- String **toString**(): Retorna un String que representa aquest objecte.

Autor

extends ***Node***

Subclasse de node que especifica que el node és un autor.

Atributs: -

Mètodes:

- **Autor**(): Constructor per defecte.
- **Autor**(int id, String nom): Constructor amb id i nom.
- **Autor**(int id, String nom, String label): Constructor amb id, nom i label.

Paper

extends ***Node***

Subclasse de node que especifica que el node és un paper (article).

Atributs: -

Mètodes:

- **Paper**(): Constructor per defecte.
- **Paper**(int id, String nom): Constructor amb id i nom.
- **Paper**(int id, String nom, String label): Constructor amb id, nom i label.

Conferencia

extends ***Node***

Subclasse de node que especifica que el node és una conferència.

Atributs: -

Mètodes:

- **Conferencia**(): Constructor per defecte.
- **Conferencia**(int id, String nom): Constructor amb id i nom.
- **Conferencia**(int id, String nom, String label): Constructor amb id, nom i label.

Terme

extends ***Node***

Subclasse de node que especifica que el node és un terme.

Atributs: -

Mètodes:

- **Terme**(): Constructor per defecte.
- **Terme**(int id, String nom): Constructor amb id i nom.

Matriu<T extends Number>

implements ***Serializable***

Classe genèrica que representa una matriu amb elements de tipus T on totes les columnes tenen la mateixa mida.

Atributs:

- *private* ***cols*** *: int*
- *private* ***rows*** *: int*
- *private* **matriu** : ArrayList<HashMap<Integer, T>>
- private **rowSqrtNorm** : ArrayList<Double>
- private **colSqrtNorm** : ArrayList<Double>
- private **changes** = false : boolean
- private **type** : int
- private **elem** : T

Mètodes:

- **Matriu**(int files, int columnes, T elem): Constructor que crea una matriu amb un nombre determinat de files i columnes amb l'element elem en cada posició.

- T **get**(int fila, int columna): Retorna l'element a la posició (fila, columna).
- void **set**(int fila, int columna, T elem): Posa elem a la posició (fila, columna).

- T **getNormaFila**(int fila): Retorna la norma per una fila.
- T **getNormaColumna**(int columna): Retorna la norma per una columna.

- int **afegirFila**(T elem): Afegeix una fila a la matriu amb elem en cada posició i retorna l'índex de la nova fila.
- int **afegirColumna**(T elem): Afegeix una columna a la matriu amb elem en cada posició i retorna l'índex de la nova columna.

- ArrayList<T> **getFila**(int fila): Retorna una fila sencera tal com està la matriu.
- ArrayList<T> **getColumna**(int columna): Retorna una fila sencera tal com està la matriu.
- ArrayList<T> **getFilaNormalitzada**(int fila): Retorna una fila normalitzada.
- ArrayList<T> **getColumnaNormalitzada**(int columna): Retorna una columna normalitzada.

- boolean **esborrarFila**(int fila): Esborra una fila de la matriu i retorna si ha sigut possible.
- boolean **esborrarColumna**(int columna): Esborra una columna de la matriu i retorna si ha sigut possible.

- int **getFiles**(): Consulta el número de files de la matriu.
- int **getColumnes**(): Consulta el número de columnes de la matriu.

- Matriu<T> **multiplicar**(Matriu<T> m) throws *IllegalArgumentException*: Multiplica el paràmetre implícit amb m i ho retorna. Llença una excepció en cas de que m i el paràmetre implícit no siguin compatibles per la multiplicació (this.getColumnes() == m.getFiles()).
- Matriu<T> **transposada**(): Retorna la matriu transposada d'aquesta. El paràmetre implícit no queda modificat.
- void **transposar**(): Transposa el paràmetre implícit.
- Matriu<T> **normalitzadaPerFiles**(): Retorna la matriu normalitzada del paràmetre implícit per files. El paràmetre implícit no queda modificat.
- Matriu<T> **normalitzadaPerColumnes**(): Retorna la matriu normalitzada del paràmetre implícit per columnes. El paràmetre implícit no queda modificat.
- void **normalitzaPerFiles**(): Normalitza el paràmetre implícit per files.
- void **normalitzaPerColumnes**(): Normalitza el paràmetre implícit per columnes.

- ArrayList<Matriu<T>> **intermedia**(Matriu<T> m): Retorna una llista amb les matrius intermèdies entre el paràmetre implícit i m. Per exemple, donada una matriu AP retorna les matrius AE i EP, on E és la matriu intermèdia.
- String **toString**(): Retorna un String que representa aquest objecte.

Graf

*implements **Serializable***

Representa la base de dades del problema que ens ocupa.

Atributs:

- *private* **adjPapersAutors** : Matriu<Byte>
- *private* **adjPapersTermes** : Matriu<Byte>
- *private* **adjPapersConferencies** : Matriu<Byte>
- *private* **idToAutors** : HashMap<Integer, Autor>
- *private* **idToPapers** : HashMap<Integer, Paper>
- *private* **idToConferencies** : HashMap<Integer, Conferencia>
- *private* **idToTermes** : HashMap<Integer, Terme>
- *private* **nomToAutors** : HashMap<String, ArrayList<Autor>>
- *private* **nomToPapers** : HashMap<String, ArrayList<Paper>>
- *private* **nomToConferencies** : HashMap<String, ArrayList<Conferencia>>
- *private* **nomToTermes** : HashMap<String,ArrayList<Terme>>

Mètodes:

- **Graf**(): Constructor per defecte del graf.

- int **afegeix**(Autor autor): Afegeix un autor al graf sense cap relació i retorna el nou identificador intern.
- int **afegeix**(Paper paper): Afegeix un paper al graf sense cap relació i retorna el nou identificador intern.
- int **afegeix**(Conferencia conferencia): Afegeix una conferència al graf sense cap relació i retorna el nou identificador intern.
- int **afegeix**(Terme terme): Afegeix una Terme al graf sense cap relació i retorna el nou identificador intern.

- boolean **elimina**(Autor autor): Elimina un autor del graf i retorna si ha sigut possible.
- boolean **elimina**(Paper paper): Elimina un paper del graf i retorna si ha sigut possible.
- boolean **elimina**(Conferencia conferencia): Elimina una conferència del graf i retorna si ha sigut possible.
- boolean **elimina**(Terme terme): Elimina una Terme del graf i retorna si ha sigut possible.

- int **consultaMidaAutor**(): Consulta el nombre d'autors al graf.
- int **consultaMidaPaper**(): Consulta el nombre de papers al graf.
- int **consultaMidaConferencia**(): Consulta el nombre de conferències al graf.
- int **consultaMidaTerme**(): Consulta el nombre de termes / temàtiques al graf.

- boolean **afegirAdjacencia**(Paper paper, Autor autor): Afegeix la relació entre el paper i l'autor. Retorna si ha sigut possible.
- boolean **afegirAdjacencia**(Paper paper, Conferencia conferencia): Afegeix la relació entre el paper i la conferència. Retorna si ha sigut possible.
- boolean **afegirAdjacencia**(Paper paper, Terme tem): Afegeix la relació entre el paper i la Terme. Retorna si ha sigut possible.
- boolean **setAdjacencia**(Paper paper, Conferencia conferencia): Modifica la relació que tenia el paper amb l'anterior conferència i la canvia per la nova. Retorna si ha sigut possible.

- boolean **eliminarAdjacencia**(Paper paper, Autor autor): Elimina la relació entre el paper i l'autor.
- boolean **eliminarAdjacencia**(Paper paper, Conferencia conferencia): Elimina la relació entre el paper i la conferència. Retorna si ha sigut possible.
- boolean **eliminarAdjacencia**(Paper paper, Terme tem): Elimina la relació entre el paper i la Terme. Retorna si ha sigut possible.

- boolean **contains**(Autor autor): Consulta si l'autor és al graf.
- boolean **contains**(Paper paper): Consulta si el paper és al graf.
- boolean **contains**(Conferencia conferencia): Consulta si la conferència és al graf.
- boolean **contains**(Terme terme): Consulta si el terme és al graf.

- Autor **consultarAutor**(int idAutor): Retorna l'autor amb l'id corresponent o null si no existeix.
- List<Autor> **consultarAutor**(String nom): Retorna tots els autors amb el nom corresponent.
- Paper **consultarPaper**(int idPaper): Retorna el paper amb l'id corresponent o null si no existeix.
- List<Paper> **consultarPaper**(String nom): Retorna tots els papers amb el nom corresponent.
- Conferencia **consultarConferencia**(int idConferencia): Retorna la conferència amb l'id corresponent o null si no existeix.
- List<Conferencia> **consultarConferencia**(String nom): Retorna totes les conferències amb el nom corresponent.
- Terme **consultarTerme**(int idTerme): Retorna la Terme amb l'id corresponent o null si no existeix.
- List<Terme> **consultarTerme**(String nom): Retorna totes les temàtiques amb el nom corresponent.

- Matriu<Byte> **consultarMatriuPaperAutor**(): Retorna la matriu d'adjacència de papers i autors.
- Matriu<Byte> **consultarMatriuPaperConferencia**(): Retorna la matriu d'adjacència de papers i conferències.
- Matriu<Byte> **consultarMatriuPaperTerme**(): Retorna la matriu d'adjacència de papers i temàtiques / termes.

Path

*implements **Serializable***

Classe que representa un path / relació entre dos tipus de nodes.

Atributs:

- *private* **path** : String
- *private* **definicio** : String

Mètodes:

- **Path**(String path): Constructor amb path.
- **Path**(String path, String definicio): Constructor amb path i def.

- String **getPath**(): Retorna el path.
- String **getDefinicio**(): Retorna la definició o descripció.

- void **setDefinicio**(String definicio): Modifica la definició amb la nova.
- String **toString**(): Retorna un String que representa aquest objecte.

ControladorPersistencia

*classe **abstracta***

Permet intercanviar dades amb la memòria secundària.

Atributs: -

Mètodes:

- *static* ArrayList<String> **llegirFitxer**(String filesystem_path)
  throws IOException: Llegeix el fitxer corresponent al path donat i retorna totes les línies del fitxer. Llençarà una *FileNotFoundException* en cas de que no es trobi el fitxer o una altra *IOException* si no es pot llegir el fitxer.

- *static* void **guardarGraf**(String filesystem_path, Graf graf)
  throws IOException: Guarda el graf serialitzable a un fitxer corresponent al path donat, sobreescrivint-lo si ja existeix. Llençarà una excepció si no es pot escriure en el fitxer.

- *static* Graf **carregarGraf**(String filesystem_path) *throws IOException*: Llegeix un graf serialitzat del fitxer corresponent al path donat Llençarà una *FileNotFoundException* en cas de que no es trobi el fitxer o una altra *IOException* si no es pot llegir el fitxer o no hi ha cap objecte serialitzat amb format correcte.

- *static* void **guardarPaths**(String filesystem_path, Collection<Path> paths)
  throws IOException: Guarda una colecció de paths serialitzables a un fitxer

corresponent al path donat, sobrescrivint-lo si ja existeix. Llençarà una excepció si no es pot escriure en el fitxer.

- *static* Collection<Path> **carregarPaths**(String filesystem_path) *throws IOException*: Llegeix una colecció de paths serialitzats que representa una colecció de paths del fitxer corresponent al path donat. Llençarà una *FileNotFoundException* en cas de que no es trobi el fitxer o una altra *IOException* si no es pot llegir el fitxer o no hi ha cap objecte serialitzat amb format correcte.

- *static* void **guardarClausures**(String filesystem_path, Map<String, Matriu<Double>> clausures) *throws IOException*: Guarda un mapa de identificador-matriu a un fitxer corresponent al path donat, sobrescrivint-lo si ja existeix. Llençarà una excepció si no es pot escriure en el fitxer.

- *static* Map<String, Matriu<Double>> **carregarClausures**(String filesystem_path) *throws IOException*: Llegeix un objecte serialitzat que representa un mapa de identificador-matriu del fitxer corresponent al path donat. Llençarà una *FileNotFoundException* en cas de que no es trobi el fitxer o una altra *IOException* si no es pot llegir el fitxer o no hi ha cap objecte serialitzat amb format correcte.

Classe: ControladorPaths

*implements **Serializable***

Permet mantenir un conjunt de paths.

Atributs:

- *private* **paths** : Map<String, Path>

Mètodes:

- **ControladorPath**(ControladorGraf controladorGraf)

- boolean **afegir**(String path, String definicio): Afegeix un nou path i retorna si ha sigut possible (encara no hi era).
- boolean **afegir**(String path): Afegeix un nou path sense definició i retorna si ha sigut possible (encara no hi era).

- boolean **modificarDefinicio**(String path, String definicio): Modifica la definció d'un path i retorna si ha sigut possible (si existía).

- boolean **esborrar**(String path): Esborra un path i retorna si ha sigut possible (si existía).

- List<String> **consultarPaths**(): Retorna una llista amb tots els paths.
- String **consultarDefinicio**(String path): Retorna la definició d'un path.

- void **guardarPaths**(String filesystem_path) *throws IOException:* Guarda tots els paths en el fitxer donat. Si el fitxer existía es sobreescriu. Llença una excepció en cas de no poder-se escriure en el fitxer.

- void **carregarPaths**(String filesystem_path) *throws IOException:* Carrega tots els paths des d'un fitxer. Llençarà una *FileNotFoundException* en cas de que no es trobi el fitxer o una altra *IOException* si no es pot llegir el fitxer.

Classe: HeteSim

implements ***Serializable***
Controlador de l'algorisme.

Atributs:

- *protected* **graf** : Graf
- *private* **clausures** : Map<String, Matriu<Double>>

Mètodes:

- **HeteSim**(Graf graf)

- Graf **getGraf**(): Retorna el graf que s'utilitza per calcular l'algorisme.

- Matriu<Double> **clausura**(String path) throws *IllegalArgumentException*: Retorna la matriu de clausura d'un path donat. La matriu de clausura és la matriu on cada element (i, j) és la rellevància de l'element representat a la fila i amb l'element representat a la fila j. Llençarà una excepció en cas de que path sigui null.

- Matriu<Double> **clausura**(Matriu<Double> left, Matriu<Double> right, String path): Retorna la matriu de clausura a partir del resultat de multiplicar les matrius normalitzades per files fins l'element mitjà (left) i el resultat de multiplicar les matrius normalitzades per columnes fins l'element mitjà (right).

- double **heteSim**(Node a, Node b, String path) throws *IllegalArgumentException*: Retorna la rellevància entre dos nodes a i b donat un path. Llençarà una excepció en cas de que path sigui null o bé a no sigui del tipus indicat pel primer tipus de node del path o bé b no sigui del tipus indicat pel últim tipus de node del path que es passa com a paràmetre.

- ArrayList<Entry<Double, Integer>> **heteSimAmbIdentificadors**(Node n, String path) throws *IllegalArgumentException*: Retorna una llista ordenada per rellevàncies on cada element és la rellevància que té n amb un altre node representat pel seu identificador amb un path donat. Llençarà una excepció en cas de que path sigui null o bé que n no sigui del tipus indicat pel primer tipus de node del path.

- ArrayList<Entry<Double, String>> **heteSimAmbNoms**(Node n, String path) throws *IllegalArgumentException*: Retorna una llista ordenada per rellevàncies on cada element és la rellevància que té n amb un altre node representat pel seu nom amb un path donat. Llençarà una excepció en cas de que path sigui null o bé que n no sigui del tipus indicat pel primer tipus de node del path.

- void **guardarClausures**(String filesystem_path) *throws IOException*: Guarda les clausures a un fitxer corresponent al path donat, sobrescrivint-lo si ja existeix. Llençarà una excepció si no es pot escriure en el fitxer.

- void **carregarClausures**(String filesystem_path) *throws IOException*: Llegeix les clausures que representa una colecció de paths del fitxer corresponent al path donat. Llençarà una *FileNotFoundException* en cas de que no es trobi el fitxer o una altra *IOException* si no es pot llegir el fitxer o no hi ha cap objecte serialitzat amb format correcte.

Classe: ControladorNodes

Controlador amb funcions sobre un node.

Atributs:

- *private* **controladorGraf** : ControladorGraf

Mètodes:

- **ControladorNodes**(ControladorGraf controladorGraf)

- int **afegirAutor**(String nom): Afegeix un autor sense cap etiqueta al graf actual i retorna el seu identificador intern.
- int **afegirAutor**(String nom, String label): Afegeix un autor al graf actual i retorna el seu identificador intern.
- int **afegirLabel**(String label, Autor autor) throws *IllegalArgumentException*: Afegeix una etiqueta a un autor del graf i retorna el seu identificador intern. Llençarà una excepció en cas de que autor no existeixi en el graf actual.
- int **afegirConferencia**(String nom): Afegeix una conferència al graf actual sense cap etiqueta i retorna el seu identificador intern.
- int **afegirConferencia**(String nom, String label): Afegeix una conferència al graf actual i retorna el seu identificador intern.

- int **afegirLabel**(String label, Conferencia conferencia) throws *IllegalArgumentException*: Afegeix una etiqueta a una conferència del graf i retorna el seu identificador intern. Llençarà una excepció en cas de que conferencia no existeixi en el graf actual.

- int **afegirPaper**(String nom): Afegeix un paper al graf actual sense cap etiqueta i retorna el seu identificador intern.
- int **afegirPaper**(String nom, String label): Afegeix un paper al graf actual i retorna el seu identificador intern.
- int **afegirLabel**(String label, Paper paper) throws *IllegalArgumentException*: Afegeix una etiqueta a un paper del graf i retorna el seu identificador intern. Llençarà una excepció en cas de que paper no existeixi en el graf actual.
- int **afegirTerme**(String nom): Afegeix un terme al graf actual i retorna el seu identificador intern.

- boolean **modificarAutor**(String nouNom, int idAutor): Modifica el nom d'un autor. Retorna si ha sigut possible (si existeix).
- boolean **modificarPaper**(String nouNom, int idPaper): Modifica el nom d'un paper. Retorna si ha sigut possible (si existeix).
- boolean **modificarConferencia**(String nouNom, int idConferencia): Modifica el nom d'una conferència. Retorna si ha sigut possible (si existeix).
- boolean **modificarTerme**(String nouNom, int idTerme): Modifica el nom d'un terme. Retorna si ha sigut possible (si existeix).

- boolean **eliminarAutor**(int idAutor): Elimina un autor. Retorna si ha sigut possible (si existía).
- boolean **eliminarPaper**(int idPaper): Elimina un paper. Retorna si ha sigut possible (si existía).
- boolean **eliminarConferencia**(int idConferencia): Elimina una conferència. Retorna si ha sigut possible (si existía).
- boolean **eliminarTerme**(int idTerme): Elimina un terme. Retorna si ha sigut possible (si existía).
- String **consultarNomAutor**(int idAutor): Consulta el nom d'un autor.
- String **consultarNomPaper**(int idPaper): Consulta el nom d'un paper.
- String **consultarNomConferencia**(int idConferencia): Consulta el nom d'una conferència.
- String **consultarNomTerme**(int idTerme): Consulta el nom d'un terme.

Classe: ControladorRelacions

Controlador de les relacions en un graf.

Atributs:

- *private* **controladorGraf** : ControladorGraf

Mètodes:

- **ControladorRelacions**(ControladorGraf controladorGraf)

- boolean **afegirAdjacenciaPaperAutor**(int idPaper, int idAutor): Afegeix una adjacència al graf actual entre un paper i un autor. Retorna si ha sigut possible (si ambdós existeixen).
- boolean **afegirAdjacenciaPaperTerme**(int idPaper, int idTerme): Afegeix una adjacència al graf actual entre un paper i un terme. Retorna si ha sigut possible (si ambdós existeixen).

- boolean **setAdjacenciaPaperConferencia**(int idPaper, int idConferencia): Afegeix una adjacència al graf actual entre un paper i una conferència. En cas de que hi hagués prèviament una altra relació entre aquests dos nodes aquesta és eliminada ja que un paper només pot estar relacionat amb una conferència. Retorna si ha sigut possible (si ambdós existeixen).

- boolean **eliminarAdjacenciaPaperAutor**(int idPaper, int idAutor): Elimina la adjacència entre un paper i un autor. Retorna si ha sigut possible (si ambdós existeixen).
- boolean **eliminarAdjacenciaPaperTerme**(int idPaper, int idTerme): Elimina la adjacència entre un paper i un terme. Retorna si ha sigut possible (si ambdós existeixen).

- List<String> **consultarRelacionsConferencia**(int idConferencia): Consulta totes les relacions d'una conferència. La llista retornada conté els noms de tots els papers relacionats.
- List<String> **consultarRelacionsAutor**(int idAutor): Consulta totes les relacions d'un autor. La llista retornada conté els noms de tots els papers relacionats.
- List<String> **consultarRelacionsTerme**(int idTerme): Consulta totes les relacions d'un terme. La llista retornada conté els noms de tots els papers relacionats.

- List<String> **consultarRelacionsPaperAmbAutor**(int idPaper): Consulta totes les relacions d'un paper. La llista retornada conté els noms de tots els autors relacionats.
- List<String> **consultarRelacionsPaperAmbTerme**(int idPaper): Consulta totes les relacions d'un paper. La llista retornada conté els noms de tots els termes relacionats.
- List<String> **consultarRelacionsPaperAmbConferencia**(int idPaper): Consulta totes les relacions d'un paper. La llista retornada conté els noms de totes les conferències relacionades.

Classe: ControladorGraf

Controlador per mantenir l'estat del graf.

Atributs:

- *protected* **graf** : Graf

Mètodes:

- **ControladorGraf**()

- Graf **getGraf**(): Obté el graf actual.

- void **importar**(String directori) throws *FileNotFoundException, IOException:* Importa i crea el graf mitjançant els fitxers del directori que es passen com a paràmetre (autors, papers, conferències, labels i relacions). Llençarà una excepció de tipus *FileNotFoundException* en cas de que algun fitxer no es trobi o altre excepció de tipus *IOException* en cas de que algun fitxer no tingui el format correcte o bé no sigui possible llegir-lo.

- void **carregar**(String directori) throws *FileNotFoundException, IOException:* Carrega el graf del fitxer corresponent al path donat. Llençarà una *FileNotFoundException* en cas de que no es trobi el fitxer o una altra *IOException* si no es pot llegir el fitxer.

- void **guardar**(String directori) throws *IOException:* Guarda el graf actual en el fitxer donat. Si el fitxer existía es sobreescriu. Llença una excepció en cas de no poder-se escriure en el fitxer.

- List<Integer> **consultarPaper**(String nom): Retorna una llista amb els identificadors de tots els papers amb el nom del paràmetre.
- List<Integer> **consultarConferencia**(String nom): Retorna una llista amb els identificadors de totes les conferències amb el nom del paràmetre.
- List<Integer> **consultarTerme**(String nom): Retorna una llista amb els identificadors de tots els termes amb el nom del paràmetre.
- List<Integer> **consultarAutor**(String nom): Retorna una llista amb els identificadors de tots els autors amb el nom del paràmetre.

- int **consultarMidaAutors**(): Consulta quants autors hi ha en el graf actual.
- int **consultarMidaPapers**(): Consulta quants papers hi ha en el graf actual.
- int **consultarMidaTermes**(): Consulta quants termes / temàtiques hi ha en el graf actual.
- int **consultarMidaConferencies**(): Consulta quantes conferències hi ha en el graf actual.

Classe: ControladorDominiPersistencia

*implements **Serializable***

Atributs:

- *public static final* **DEFAULT_FILEPATH_GRAF** : String = “graf.dat”
- *public static final* **DEFAULT_FILEPATH_PATHS** : String= “paths.dat”
- *public static final* **DEFAULT_FILEPATH_CLAUSURES** : String= “clausures.dat”
- *protected* **controladorGraf** : ControladorGraf
- *protected* **controladorPaths** : ControladorPaths
- *private* **controladorHeteSim** : HeteSim

Mètodes:

- **ControladorDominiPersistencia**(ControladorGraf controladorGraf, ControladorPaths controladorPaths, HeteSim controladorHeteSim)

- **ControladorDominiPersistencia**(ControladorGraf controladorGraf, ControladorPaths controladorPaths)

- void **guardarDades**() *throws IOException:* Guarda totes les dades que es poden (fa una crida als mètodes de guardar graf, paths i clausures) als fitxers corresponents als paths per defecte. Si algun fitxer existía es sobreescriu. Llença una excepció en cas de no poder-se escriure en els fitxers.

- void **guardarDades**(String fitxerGraf, String fitxerPaths, String fitxerClausures) *throws IOException:* Guarda totes les dades (fa una crida als mètodes de carregar graf, paths i clausures) als fitxers corresponents als paths donats per paràmetre. Si algun fitxer existía es sobreescriu. Llença una excepció en cas de no poder-se escriure en els fitxers.

- void **guardarDades**(String fitxerGraf, String fitxerPaths) *throws IOException:* Guarda totes les dades (fa una crida als mètodes de guardar graf i paths) als fitxers corresponents als paths donats per paràmetre. Si algun fitxer existía es sobreescriu. Llença una excepció en cas de no poder-se escriure en els fitxers.

- void **carregarDades**() *throws IOException:* Carrega totes les dades (fa una crida als mètodes de guardar graf, paths i clausures) pels fitxers per defecte existents. Llença una excepció en cas de no poder llegir els fitxers.

- void **carregarDades**(String fitxerGraf, String fitxerPaths, String fitxerClausures) *throws IOException:* Carrega totes les dades (fa una crida als mètodes de guardar graf, paths i clausures) pels fitxers corresponents als paths donats per paràmetre. Llençarà una *FileNotFoundException* en cas de que no es trobi algun fitxer o una altra *IOException* si no es pot llegir el fitxer o no hi ha cap objecte serialitzat amb format correcte.

- void **carregarDades**(String fitxerGraf, String fitxerPaths) *throws IOException:* Carrega totes les dades (fa una crida als mètodes de guardar graf i paths)

dels fitxers corresponents als paths donats per paràmetre. Llençarà una *FileNotFoundException* en cas de que no es trobi algun fitxer o una altra *IOException* si no es pot llegir el fitxer o no hi ha cap objecte serialitzat amb format correcte.

Repartiment de responsabilitats

Grup 1:

- Node
- Autor
- Paper
- Conferència
- Terme
- Graf
- ControladorNodes
- ControladorGraf

Grup 2:

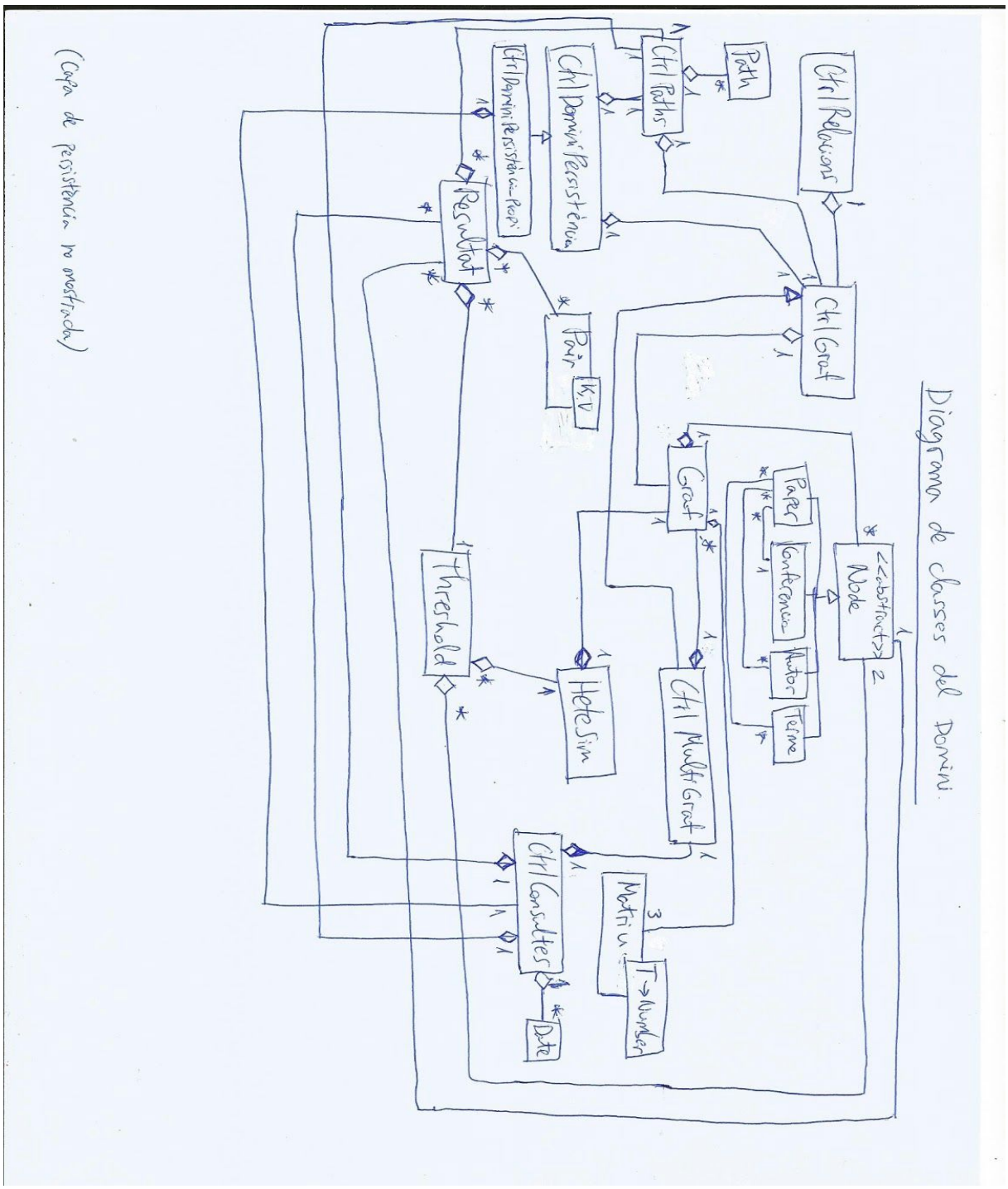
- ControladorPersistència
- ControladorDominiPersistència
- HeteSim

Grup 3:

- Path
- ControladorPaths
- ControladorRelacions
- Matriu<T>

Part pròpia del grup

Diagrama de classes



Especificació detallada de classes

ControladorMultigraf

extends ***ControladorGraf***

Atributs:

- *private* **idActual** : String
- *private* **grafs** : HashMap<String, Graf>
- *private* **controladors** : HashMap<String, HeteSim>

Mètodes:

- **ControladorMultigraf**(): Constructor per defecte, sense cap graf carregat

- String **getIdActual**(): Retorna l'IdActual.

- List<String> **getNomsGrafs**(): Retorna una llista amb els noms de tots els grafs carregats.

- HeteSim **getHeteSim**(): Retorna el controlador de l'algorisme HeteSim del graf actual.

- void **afegirGraf**(String nomGraf): Crea una nova entrada a grafs amb el nou graf de nom nomGraf i una altra a controladors (els dos seran buits). El graf actual i l'id actual passen a ser els d'aquest graf.

- void **importar**(String nomGraf, String directori) throws *IOException:* Importa i crea un graf que tindrà el nom donat mitjançant els fitxers del directori que es passen com a paràmetre (autors, papers, conferències, labels i relacions). Aquest graf s'estableix com l'actual. Llençarà una excepció de tipus *FileNotFoundException* en cas de que algun fitxer no es trobi o altre excepció de tipus *IOException* en cas de que algun fitxer no tingui el format correcte o bé no sigui possible llegir-lo. El graf i idActual passen a ser el graf importat.

- boolean **seleccionarGraf**(String nomGraf): Selecciona el graf amb nom nomGraf com a graf actual. Retorna si ha sigut possible (el graf existía).

- *@Override* void **carregar**(String directori) throws *IOException:* Carrega tots els grafs amb les seves clausures dels fitxers corresponents al directori

donat. Llençarà una *FileNotFoundException* en cas de que no es trobi algun fitxer o una altra *IOException* si no es pot llegir algun fitxer.

- *@Override* void **guardar**(String directori) throws *IOException:* Guarda tots els grafs i les seves clausures en el directori donat. Si algun fitxer existía es sobreescriu. Llença una excepció en cas de no poder-se escriure algun fitxer.

ControladorDominiPersistenciaPropi

extends ***ControladorDominiPersistencia***

Atributs:

- *public static final* String **DEFAULT_DIRECTORY_GRAFS** = “grafs/”
- *private* **controladorConsultes** : ControladorConsultes

Mètodes:

- **ControladorDominiPersistenciaPropi**(ControladorGraf ctrlGraf, ControladorPaths ctrlPaths, ControladorConsultes ctrlConsultes)

- *@Override* void **guardarDades**() *throws IOException:* Guarda totes les dades que es poden (fa una crida als mètodes de guardar grafs i clausures, paths i resultats) als fitxers corresponents als paths per defecte. Si algun fitxer existía es sobreescriu. Llença una excepció en cas de no poder-se escriure en els fitxers.

- *@Override* void **carregarDades**() *throws IOException:* Carrega totes les dades (fa una crida als mètodes de carregar grafs i clausures, paths i resultats) pels fitxers per defecte existents. Llença una excepció en cas de no poder llegir els fitxers.

Threshold

implements ***Serializable***

Atributs:

- *private* **rellevancia** : double
- *private* **a** : Node
- *private* **b** : Node
- *private* **path** : String
- *private* **hs**: HeteSim

Mètodes:

- **Threshold**(double rellevancia, Node a, Node b, String path, HeteSim hs)
- **Threshold**(Node a, Node b, String path, HeteSim hs) throws *IllegalArgumentException:* Constructor que calcula la rellevància entre els dos nodes segons el path fent servir el HeteSim. Llençarà una excepció en cas de que path sigui null o bé que el Node a no sigui del tipus indicat pel primer tipus de node de path o bé que el Node b no sigui del tipus indicat per l'últim tipus de node de path.

- ArrayList<Node> **getNodes**(): Retorna una llista de dos posicions amb els nodes que formen aquest Threshold.

- String **getPath**(): Retorna el nom del path que forma aquest Threshold.

- double **getRellevancia**(): Retorna la rellevància entre els nodes que formen aquest Threshold.

- void **setRellevancia**(double rellevancia): Modifica la rellevància entre els nodes que formen aquest Threshold.

- *@Override* String **toString**(): Retorna un String que representa aquest objecte.

Pair<K extends Comparable<K> & Serializable, V extends Serializable>

implements ***Entry<K, V>, Comparable<Pair<K, V>>, Serializable***

Atributs:

- *private* **key** : K
- *private* **value** : V

Mètodes:

- **Pair**(K key, V value)
- K **getKey**(): Retorna la clau del pair.
- V **getValue**(): Retorna el valor del pair.
- void **setKey**(K key): Canvia la clau del pair.
- V **setValue**(V value): Canvia el valor del pair i retorna el valor antic.

- int **compareTo**(Pair<K, V> p): Retorna un nombre positiu, 0 o un nombre negatiu segons si this és menor, igual o major a p, respectivament. La comparació es fa entre les claus dels dos pairs.

- String **toString**(): Retorna un String que representa aquest objecte.

Resultat

implements ***Serializable***

Atributs:

- *private* **dada** : Node
- private **controladorPaths** : ControladorPaths
- *private* **nomPath** : String
- *private* **nomGraf** : String
- *private* **resultats** : ArrayList<Pair<Double, Node>>
- *private* **threshold** : Threshold

Mètodes:

- **Resultat**(Node dada, String nomPath, ControladorPaths controladorPaths, String nomGraf, ArrayList<Pair<Double, Node>> resultats): Constructor sense threshold, el paràmetre resultats serà ordenat.
- **Resultat**(Node dada, String nomPath, ControladorPaths controladorPaths, String nomGraf, ArrayList<Pair<Double, Node>> resultats, Threshold threshold): Constructor complet, el paràmetre resultats serà ordenat.

- void **setNode**(Node node): Canvia el node d'aquest resultat.
- Node **getNode**(): Retorna el node d'aquest resultat.
- void **setPath**(String nomPath): Canvia el path d'aquest resultat.
- String **getPath**(): Retorna el path d'aquest resultat.
- void **setNomGraf**(String nomGraf) : Canvia el nom del graf d'aquest resultat.
- String **getNomGraf**(): Retorna el nom del graf d'aquest resultat.
- void **setResultats**(ArrayList<Pair<Double,Node>> resultats): Canvia els resultats d'aquest resultat.
- ArrayList<Pair<Double, Node>> **getResultats**(): Retorna la llista ordenada de resultats.
- void **setThreshold**(Threshold threshold): Canvia el threshold d'aquest resultat.
- Threshold **getThreshold**(): Retorna el threshold d'aquest resultat.

- int **size**(): Retorna el total de tuples de dada i rellevància que conté el resultat.
- boolean **isEmpty**(): Retorna si el resultat no conté cap tupla de dada i rellevància.

- Pair<Double, Node> **get**(int index) throws IndexOutOfBoundsException : Retorna el resultat en una posició. La primera posició és la 0.

- boolean **set**(int index, Pair<Double, Node> resultat): Canvia el resultat d'una posició. Retorna si 0 <= index < size(). S'ha de tenir en compte que després d'aquesta crida si el resultat és true llavors és probable que aquesta posició hagi canviat degut a l'ordre dels resultats.
- void **clear**(): Esborra totes les tuples de dada i rellevància de resultats.
- void **afegir**(Double rellevancia, Node dada): Afegeix un resultat.
- boolean **esborrar**(int index): Esborra un resultat i retorna si 0<= index< size().
- boolean **setRellevancia**(int index, Double rellevancia): Canvia la rellevància d'una posició dels resultats, retorna si ha sigut possible (0 <= index < size() i 0 <= rellevancia <= 1). S'ha de tenir en compte que després d'aquesta crida si el resultat és true llavors és probable que aquesta posició hagi canviat degut a l'ordre dels resultats.
- boolean **setDada**(int index, Node dada): Canvia la dada en una tupla dels resultats i retorna si 0 <= index <= size().

- void **filtrarElsPrimers**(int n): Elimina tots els resultats excepte els n primers.
- void **filtrarElsUltims**(int n): Elimina tots els resultats excepte els n últims.
- void **filtrarPetEtiqueta**(String label): Elimina tots els resultats menys els resultats que tenen el node amb l'etiqueta label.
- void **filtrarPerRellevancia**(double min, double max): Elimina tots els resultats menys els que tenen una rellevància entre min i max, ambdós inclosos.

- *@Override* String **toString**(): Retorna un String que representa aquest objecte.

ControladorPersistènciaPropi

Classe ***abstracta*** (només per evitar que s'instancii ja que els seus mètodes són estàtics), *extends* ***ControladorPersistencia***

Mètodes:

- *static* void **guardarResultats**(String filesystem_path, TreeMap<Date, Resultat> resultat) *throws IOException*: Guarda els resultats a un fitxer corresponent al path donat, sobrescrivint-lo si ja existeix. Llençarà una excepció si no es pot escriure en el fitxer.

- *static* TreeMap<Date, Resultat> **carregarResultats**(String filesystem_path) *throws IOException*: Llegeix els resultats del fitxer corresponent al path donat. Llençarà una *FileNotFoundException* en cas de que no es trobi el fitxer o una altra *IOException* si no es pot llegir el fitxer o no hi ha resultats amb format correcte.

ControladorExportacio

Classe ***abstracta*** (només per evitar que s'instancii ja que els seus mètodes són estàtics)

Mètodes:

- *static* void **exportar**(String filesystem_path, Date date, Resultat resultat) *throws IOException*: Exporta el resultat al fitxer donat (se sobreescriu si ja existía). Llencarà una excepció si no es pot crear o escriure en el fitxer.

ControladorConsultes

Atributs:

- *public static* **DEFAULT_PATH_RESULTATS** : String = "resultats.dat"
- *private* **resultats** : TreeMap<Data, Resultat>
- *private* **ultimaConsulta** : Date
- *private* **controladorMultigraf** : ControladorMultigraf
- *private* **controladorPaths** : ControladorPaths

Mètodes:

- **ControladorConsultes**(ControladorMultigraf controladorMultigraf, ControladorPaths controladorPaths)

- String **consultarResultat**() throws *IllegalArgumentException*: Consulta l'últim resultat consultat o llença una excepció en cas que no existeixi una última consulta.
- String **consultarResultat**(Date data) *throws IllegalArgumentException*: Consulta el resultat amb la data especificada o llença una excepció en cas de que no existeixi.

- String **consultarDates**(): Retorna les dates de totes les consultes.

- String **consulta**(String path, int idNode) *throws IllegalArgumentException*: Realitza una consulta de rellevàncies per un node amb un path determinat. Llençarà una excepció en cas de que el node o el path no existeixin.
- String **consulta**(String path, int idNode, int idNodeThreshold1, int idNodeThreshold2, String thresholdPath) *throws IllegalArgumentException*: Realitza una consulta de rellevàncies per un node amb un path determinat i

un threshold com a filtre. Llençarà una excepció en cas de que el node o el path no existeixin o bé algun dels nodes o el path del threshold no existeixin.

- boolean **esborrarConsulta**(Date data): Esborra la consulta que té la data especificada i retorna si ha sigut possible.

- void **filtrarElsPrimers**(int n) throws *IllegalArgumentException*: Elimina tots els resultats excepte els n primers de la última consulta o llença una excepció en cas que no existeixi una última consulta.
- void **filtrarElsUltims**(int n) throws *IllegalArgumentException*: Elimina tots els resultats excepte els n últims de la última consulta o llença una excepció en cas que no existeixi una última consulta.
- void **filtrarPetEtiqueta**(String label) throws *IllegalArgumentException*: Elimina tots els resultats de la última consulta menys els resultats que tenen el node amb l'etiqueta label o llença una excepció en cas que no existeixi una última consulta.
- void **filtrarPerRellevancia**(double min, double max) throws *IllegalArgumentException*: Elimina tots els resultats de la última consulta menys els que tenen una rellevància entre min i max, ambdós inclosos o llença una excepció en cas que no existeixi una última consulta.

- void **clear**() throws *IllegalArgumentException*: Esborra tots els resultats de la última consulta o llença una excepció en cas que no existeixi una última consulta.
- boolean **afegir**(double rellevancia, int idNode) throws *IllegalArgumentException*: Afegeix a la última consulta un resultat amb la rellevància indicada i el node amb l'id indicat o llença una excepció en cas que no existeixi una última consulta o no existeixi el node indicat.
- void **esborrar**(int index) throws *IllegalArgumentException*: Esborra de la última consulta el resultat de la posició indicada i retorna si s'ha pogut realitzar (index es troba dins d'un rang correcte) o llença una excepció en cas que no existeixi una última consulta.
- String **getTipusNode**() throws *IllegalArgumentException*: Retorna el tipus de nodes dels resultats de la última consulta: "Autor", "Paper", "Conferencia" o "Terme", o bé llença una excepció en cas que no existeixi una última consulta.
- boolean **setRellevancia**(int index, double rellevancia) throws *IllegalArgumentException*: Canvia la rellevància del resultat d'una posició de la última consulta i retorna si ha sigut possible (rellevancia és >= 0 i <= 1 i index es troba dins del rang correcte) o bé llença una excepció en cas que no existeixi una última consulta. S'ha de tenir en compte que després d'aquesta

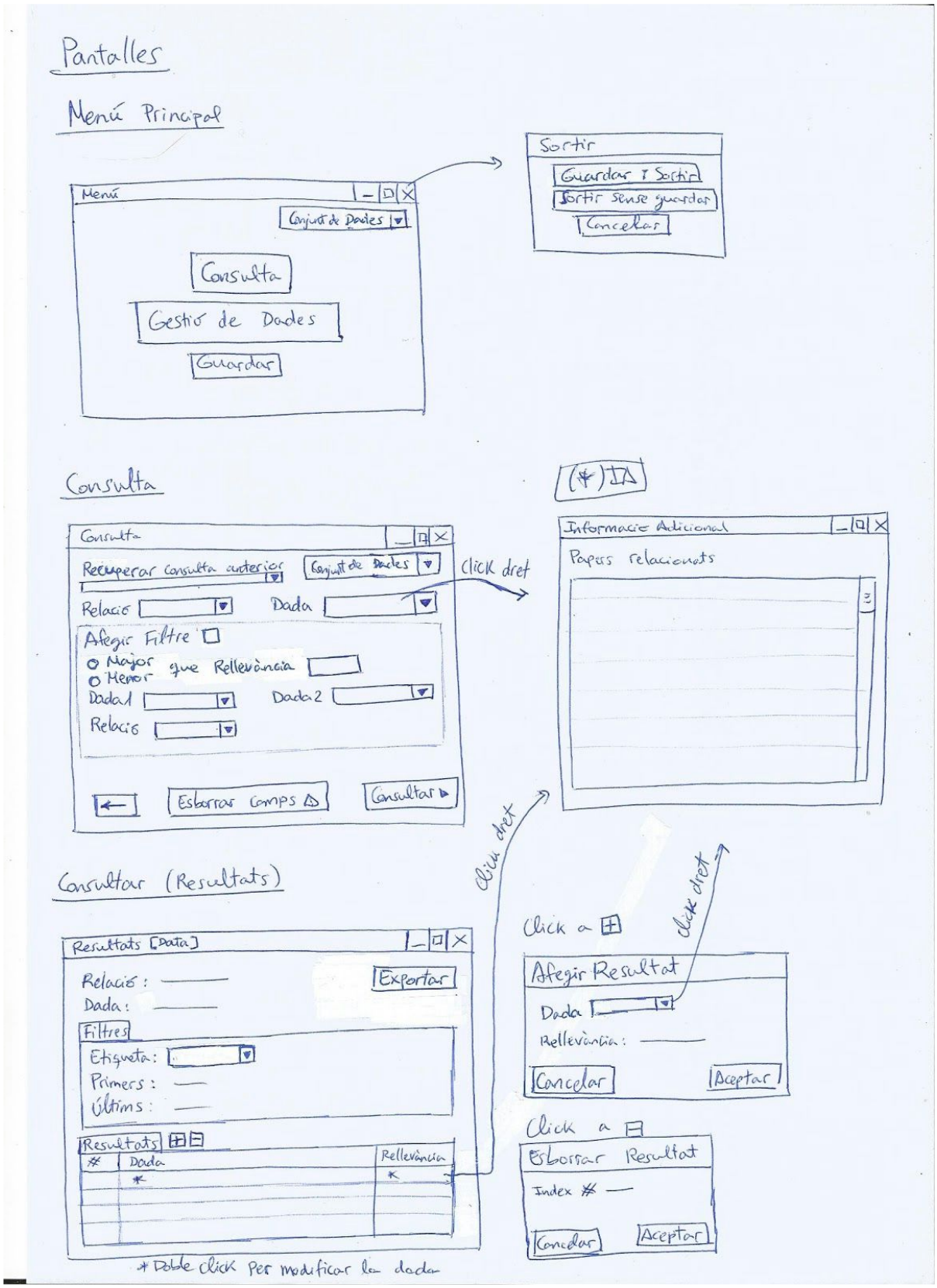
crida si el resultat és true llavors és probable que aquesta posició hagi canviat degut a l'ordre dels resultats.

- boolean **setDada**(int index, int idNode) throws *IllegalArgumentException*: Canvia la dada d'una posició dels resultats de la última consulta i retorna si ha sigut possible (la dada existeix i l'índex es troba dins del rang correcte) o bé llença una excepció en cas que no existeixi una última consulta.
- boolean **canviarNom**(int index, String nom) throws *IllegalArgumentException*: Canvia el nom de la dada d'una posició dels resultats de la última consulta. Retorna si ha sigut possible (l'índex es troba dins del rang correcte) o bé llença una excepció en cas que no existeixi una última consulta.

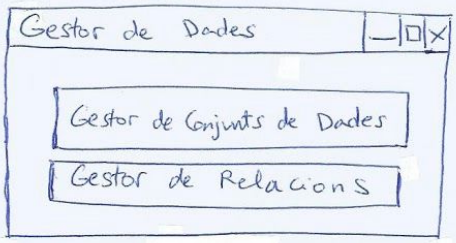
- void **setThreshold**(int idNode1, int idNode2, String path) throws *IllegalArgumentException*: Canvia el threshold de l'última consulta per un amb els nodes i el path indicats i refà la consulta, o bé llença una excepció en cas que no existeixi una última consulta.
- void **setPath**(String path, int id) throws *IllegalArgumentException*: Canvia el path i la dada de l'última consulta pels indicats i refà la consulta, o bé llença una excepció en cas que no existeixi una última consulta.
- void **setDada**(int id) throws IllegalArgumentException: Canvia la dada de l'última consulta per l'indicada i refà la consulta, o bé llença una excepció en cas que no existeixi una última consulta

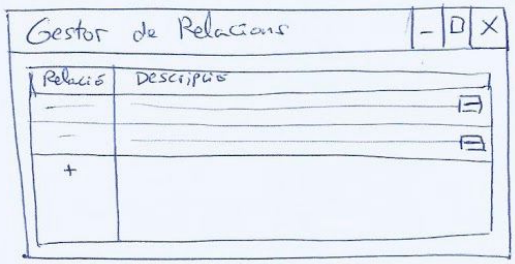
- void **exportarResultat**(String filesystem_path) *throws IOException, IllegalArgumetException*: Exporta el resultat consultat més recentment al fitxer donat. Llencarà una excepció si no es pot crear o escriure en el fitxer o bé si no existeix una última consulta.
- void **guardarResultats**() *throws IOException*: Exporta tots els resultats al fitxer per defecte. Llencarà una excepció si no es pot crear o escriure en aquest fitxer.
- void **carregarResultats**() *throws IOException*: Carrega els resultats del fitxer per defecte. Llençarà una una altra *IOException* si no existeix el fitxer, no es pot llegir o no hi ha resultats amb format correcte.

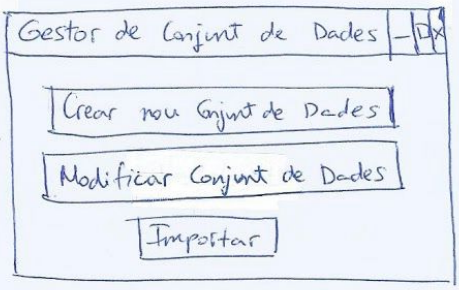
Disseny de pantalles

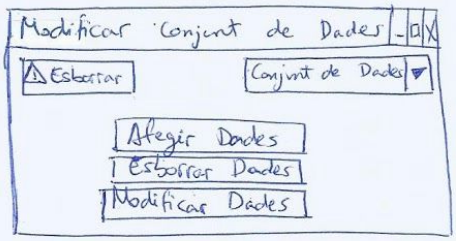


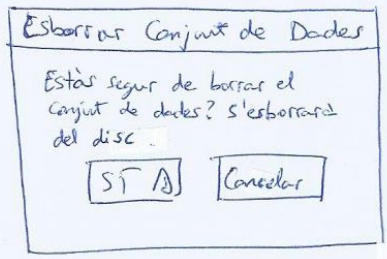
Gestió de Dades

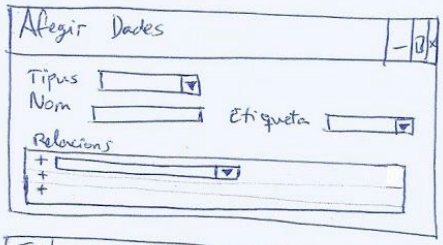


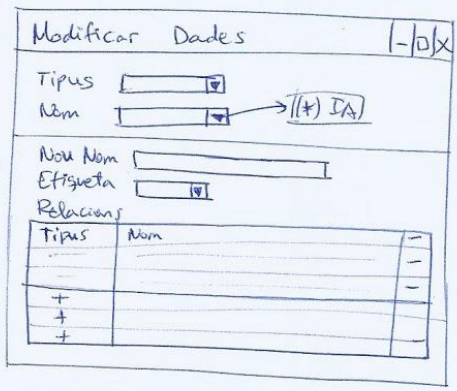


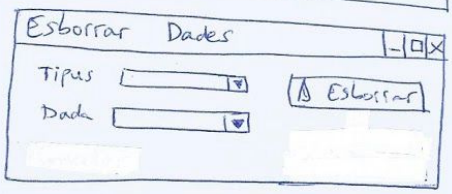












Descripció d'estructures de dades i algorismes

HeteSim

Per fer el càlcul del HeteSim, en tots el casos, primer descomposem el *path* i obtenim les matrius d'adjacència necessaries pel càlcul, posteriorment inserim les matrius intermèdies si el path té longitud senar i finalment totes són normalitzades. Per poder fer bé els càlculs totes les matrius són convertides de Matriu<Byte> a Matriu<Double> creant una nova Matriu<Double> i copiant els valors originals a aquesta. Les matrius resultants són guardades en un ArrayList<Matriu<Double>> per fer més fàcil la iteració sobre el contigut.

En el cas del càlcul de clausures, les matrius obtingudes de la descomposició es multipliquen des de la posició 0 fins a la meitat per obtenir la matriu *left* i des de la meitat fins al final per obtenir la matriu *right*. Finalment, obtenim la clausura multiplicant ambdues matrius i, només si el *path* tenia dos elements, normalitzant els seus elements. Es retorna una Matriu<Double> amb les rellevàncies entre els tipus de Node indicats pel *path*. Per tal de fer més eficients la consulta de nou a la clausura consultada, aquesta es guarda a un HashMap<String, Matriu<Double>> on l'*String* es el *path* de la clausura.

Per fer el càlcul de la relleància entre dos Node, s'agafa la fila corresponent al primer Node i es va fent la multiplicació amb les matrius següents (obtenint sempre una altra fila) fins la meitat, després agafem la columna de l'última matriu corresponent al segon Node i l'anem multiplicant amb les matrius anteriors (sempre obtenint una columna) fins la meitat. Posteriorment, es fa la multiplicació entre la fila i columnes resultants i es retorna el valor númeric resultant normalitzat.

Per fer el càlcul entre un Node i tots els Node d'un altre tipus, s'agafa la fila corresponent al Node i es va fent la multiplicació amb les matrius següents (obtenint sempre una altra fila) fins la meitat, després es multipliquen les matrius des de la meitat fins al final. Finalment es multiplica la fila per la matriu resultants i s'obté una llista de les rellevàncies entre el Node i tots els Node d'un altre tipus. Finalment, segons si es vol tenir les rellevàncies amb el IDs del Node o bé amb els seus noms es retorna un ArrayList<Pair<Double, Integer>> o un ArrayList<Pair<Double, String>>, essent Pair una implementació de la *interface Entry*.

Resultat

La classe Resultat és una classe que conté informació sobre un resultat. Concretament conté informació sobre la dada, el path, el nom del graf i el filtre (threshold) a partir dels quals s'ha realitzat la consulta. A més, contindrà el conjunt de tuples (rellevància,dada) que s'han obtingut al fer la consulta.
Cal dir que la classe resultat només conté informació, sense mantenir totes les regles que s'haurien de complir per tal de que els resultats tinguin sentit, ja que l'usuari ha de poder

modificar-lo al seu gust. L'única propietat que sempre es manté és que la llista serà decreixent segons la rellevància de cada tupla.
Per tant, un objecte de la classe por contenir informació incoherent. Per exemple, si es canvia el threshold, no es reajustaran els resultats al nou threshold, si es canvia la rellevància d'una tupla no es comprovarà que aquesta s'ajusti al threshold, i tampoc es comprovarà que els tipus dels nodes concordin amb el Path del resultat.

Repartiment de responsabilitats

Arnau Badía Sampera

- *ControladorDominiPersistència*
- ControladorDominiPersistènciaPropi
- ControladorMultigraf
- Resultat

Carla Claverol González

- ControladorConsultes
- *Pair<K, V>*
- Threshold

Carlos Lázaro Costa

- *HeteSim*
- *ControladorPersistència*
- ControladorExportació

Guillem Castro Olivares

- *HeteSim*
- ControladorPersistènciaPropi