Documentació

Entrega v1.0

Projectes de Programació QT 2025-26

Grup 33-5

Arnau Sánchez Coll *id:arnau.sanchez.coll*

Roger Corcoles *id:roger.corcoles*

Adrià Aguilar *id:adria.aguilar*

Joaquim Fuentes *id:joaquim.fuentes*

Igor Bolige Cubas *id:igor.bolige*

ÍNDEX

1. Descripció dels Casos d'ús

Els casos d'ús descrits a continuació representen les principals funcionalitats que ofereix el sistema d'enquestes, tant des del punt de vista del creador o administrador com del participant o analista.
Cada cas d'ús especifica breument l'objectiu de la interacció, l'actor principal implicat, les precondicions necessàries, el flux principal d'accions i les possibles alternatives o excepcions.
L'objectiu d'aquesta secció és complementar el diagrama de casos d'ús amb una descripció textual que permeti entendre clarament el comportament del sistema i el paper de cada actor dins del procés general de gestió, resposta i anàlisi d'enquestes.

Gestió d'Usuaris

- Objectiu: agrupar les operacions d'alta, consulta i manteniment d'usuaris.
- Actor: Creador/Administrador.
- Notes: Paraigües dels UC: Crear usuari, Llistar usuaris, Consultar usuari, Modificar usuari, Crear enquesta per usuari, Llistar enquestes d'un usuari.

Crear usuari

- Objectiu: donar d'alta un nou usuari al sistema.
- Actor: Creador/Administrador.
- Precondicions: actor amb permisos; nom i email facilitats.
- Flux principal: l'actor introdueix nom i email; el sistema valida (no buits, format d'email vàlid, email no duplicat); el sistema assigna un identificador únic (idU) i crea l'usuari; confirma l'alta mostrant idU.
- Alternatives: nom/email nuls o buits → error de validació; email amb format invàlid → error; email duplicat → operació rebutjada.
- Postcondicions: usuari creat i disponible per a consultes i per crear-hi enquestes.

Llistar usuaris

- Objectiu: obtenir el llistat d'usuaris registrats.
- Actor: Creador/Administrador.
- Precondicions: cap.
- Flux principal: l'actor sol·licita el llistat; el sistema retorna la col·lecció d'usuaris (idU, nom, email) ordenada (p. ex. per idU).
- Alternatives: no hi ha usuaris → es mostra llista buida o missatge informatiu.
- Postcondicions: cap canvi d'estat.

Consultar usuari

- Objectiu: visualitzar les dades d'un usuari concret.
- Actor: Creador/Administrador.
- Precondicions: idU vàlid i existent.
- Flux principal: l'actor indica idU; el sistema recupera i mostra nom, email i metadades associades (p. ex. nombre d'enquestes creades).
- Alternatives: idU inexistent → "usuari no trobat"; falta de permisos → accés denegat.

- Postcondicions: cap canvi d'estat.

Modificar usuari
- Objectiu: actualitzar dades bàsiques de l'usuari (nom, email).
- Actor: Creador/Administrador.
- Precondicions: idU vàlid; dades noves facilitades.
- Flux principal: l'actor edita nom i/o email; el sistema valida (no buits, email amb format vàlid, email no duplicat); el sistema aplica els canvis i confirma.
- Alternatives: validacions fallides (nom/email buits, email invàlid, email ja existent) → operació rebutjada; cancel·lació de l'actor.
- Postcondicions: l'usuari queda actualitzat en memòria (i en persistència si després s'executa "Guardar").

Crear enquesta per usuari
- Objectiu: associar una nova enquesta a un usuari existent.
- Actor: Creador/Administrador.
- Precondicions: idU existent; títol d'enquesta no buit.
- Flux principal: l'actor indica idU i títol; el sistema valida; crea l'enquesta amb idE únic i la vincula a idU; retorna confirmació amb idE.
- Alternatives: usuari inexistent → error; títol nul/buit → error de validació.
- Postcondicions: enquesta creada i associada a l'usuari; disponible per ser modificada/guardada.

Llistar les enquestes d'un usuari
- Objectiu: obtenir totes les enquestes creades per un usuari.
- Actor: Creador/Administrador.
- Precondicions: idU vàlid i existent.
- Flux principal: l'actor indica idU; el sistema retorna la llista d'enquestes (idE, títol, estat si escau) associades a l'usuari.
- Alternatives: usuari sense enquestes → es retorna llista buida o missatge informatiu; idU inexistent → error.
- Postcondicions: cap canvi d'estat.

Gestió d'Enquestes
- Objectiu: agrupar les operacions d'administració d'enquestes.
- Actor: Creador/Administrador.
- Notes: Paraigües dels UC: Guardar Enquesta, Modificar Enquesta (Modificar títol enquesta, Afegir pregunta a enquesta, Eliminar pregunta de enquesta), Eliminar Enquesta, Crear enquesta, Consultar enquesta.

Guardar Enquesta
- Objectiu: persistir en disc/BD l'estat actual de l'enquesta.
- Actor: Creador/Administrador.
- Precondicions: Enquesta carregada i vàlida.
- Flux principal: el sistema serialitza i desa; confirma operació.
- Alternatives: error d'IO; enquesta invàlida → no es desa.
- Postcondicions: versió guardada disponible per reobrir/importar.

Modificar Enquesta
- Objectiu: canviar metadades i/o conjunt de preguntes d'una enquesta existent.
- Actor: Creador/Administrador.
- Precondicions: Enquesta existent carregada.
- Flux principal: l'actor edita títol, descripció, obligatòries, afegeix/elimina/modifica preguntes; el sistema valida la definició després de cada canvi.
- Alternatives: cancel·lar canvis; errors de validació (camp buit, opcions buides…).
- Postcondicions: l'estat editat queda en memòria de treball; (persistència si després es fa Guardar Enquesta).

Crear Enquesta
- Objectiu: donar d'alta una nova enquesta al sistema.
- Actor: Creador/Administrador.
- Precondicions: usuari autenticat i existent al sistema, títol d'enquesta disponible i no buit.
- Flux principal: l'actor indica el títol (i opcionalment una descripció), el sistema valida camps obligatoris, assigna un identificador únic (idE) i crea l'enquesta buida (sense preguntes), el sistema associa l'enquesta a l'actor creador i el sistema mostra confirmació amb l'idE.
- Alternatives: títol nul o buit → error de validació, usuari inexistent o sense permisos → operació rebutjada.
- Postcondicions: l'enquesta queda creada en memòria de treball i disponible per ser consultada o modificada; la persistència en disc/BD es fa només si posteriorment s'executa "Guardar Enquesta".

Consultar Enquesta
- Objectiu: visualitzar la informació d'una enquesta existent.
- Actor: Creador/Administrador.
- Precondicions: enquesta existent identificador d'enquesta (idE) vàlid, permisos suficients.
- Flux principal: l'actor selecciona l'enquesta pel seu idE; el sistema recupera i mostra el títol, l'id del creador, i la llista de preguntes (idP, tipus i enunciat); el sistema pot mostrar metadades addicionals (nombre de preguntes, estat, etc.).
- Alternatives: idE inexistent → error "enquesta no trobada"; enquesta sense preguntes → es mostra buida; falta de permisos → accés denegat.
- Postcondicions: cap canvi d'estat; l'enquesta continua disponible per a modificació o guardat.

Eliminar Enquesta
- Objectiu: suprimir una enquesta del repositori/persistència.
- Actor: Creador/Administrador.
- Precondicions: enquesta existent seleccionada; confirmació de l'usuari.
- Flux principal: el sistema elimina i informa.
- Alternatives: cancel·lació; dependències que impedeixen eliminar.
- Postcondicions: l'enquesta ja no és disponible a llistats ni per a resposta.

Gestió de Preguntes
- Objectiu: agrupar totes les operacions de definició i consulta del banc de preguntes.
- Actor: Creador/Administrador.
-Notes: Paraigües dels UC: Crear pregunta (text / numèrica / opció única / opció múltiple / ordinal) i Imprimir totes les preguntes. La vinculació d'una pregunta a una enquesta es fa des de Modificar

Enquesta (afegir/eliminar pregunta) i dona accés al catàleg de preguntes perquè després puguin ser associades a enquestes o modificades en altres casos d'ús d'administració.

Crear pregunta (text / numèrica / opció única / opció múltiple / ordinal)

- Objectiu: donar d'alta una nova pregunta del tipus desitjat per utilitzar-la a les enquestes.
- Actor: Creador/Administrador.
- Precondicions: sistema en execució, gestor de preguntes accessible, actor amb permisos; enunciat no buit; tipus seleccionat.
- Flux principal: L'actor tria el tipus de pregunta. Introdueix enunciat i marca si és obligatòria. Segons el tipus, el sistema demana rang numèric o llistat d'opcions/ordre. El sistema valida: enunciat no buit; rangs coherents; opcions no buides ni duplicades; paràmetres específics. Si tot és correcte, s'assigna idP únic i s'afegeix al catàleg. Es mostra confirmació.
- Alternatives: cancel·lació de la creació; errors de validació (enunciat buit, min ≥ max, opcions nul·les/buides/insuficients, qMax incoherent). Es mostra missatge i es permet reintroduir dades sense tancar el gestor.
- Postcondicions: la pregunta queda registrada amb idP únic i disponible per vincular a enquestes; la persistència dependrà del cas d'ús Guardar Enquesta.

Tipus i validacions específiques:

Text

- Paràmetres: enunciat, obligatòria (opcionalment longitud màxima).
- Validacions: enunciat no buit.
- Errors típics: enunciat buit o nul.

Numèrica (integer)

- Paràmetres: enunciat, obligatòria, min, max.
- Validacions: min < max i valors vàlids.
- Errors típics: min ≥ max, valors no numèrics.

Opció única (nominal_unica)

- Paràmetres: enunciat, obligatòria, opcions.
- Validacions: ≥ 2 opcions, sense duplicats ni buides.
- Errors típics: una sola opció, duplicats, cadenes buides.

Opció múltiple (nominal_multiple)

- Paràmetres: enunciat, obligatòria, opcions, qMax (màxim seleccions).
- Validacions: ≥ 2 opcions, sense duplicats ni buides; 1 ≤ qMax ≤ nombre d'opcions.
- Errors típics: qMax fora de rang; opcions invàlides.

Ordinal

- Paràmetres: enunciat, obligatòria, llista ordenada d'opcions (p. ex. [Baix, Mitjà, Alt]).
- Validacions: ≥ 2 opcions, sense duplicats ni buides; ordre explícit.
- Errors típics: menys de 2 opcions, duplicats, cadenes buides.

Imprimir totes les preguntes

- Objectiu: llistar en pantalla totes les preguntes definides al sistema, amb informació bàsica.
- Actor: Creador/Administrador.

- Precondicions: cap (si el catàleg és buit, es mostra un missatge).

- Flux principal: l'actor selecciona l'opció d'imprimir; el sistema recorre el catàleg i mostra per cada pregunta: idP, tipus (text, integer, nominal única, nominal múltiple, ordinal), enunciat, obligatorietat i, si escau, rang min/max, opcions i qMax.
- Alternatives: catàleg buit → missatge "No hi ha preguntes registrades"; si hi ha errors d'E/S de consola, es pot repetir l'operació o tornar al menú.
- Postcondicions: cap canvi d'estat; només es mostra l'estat actual del banc per facilitar-ne l'edició o la vinculació a enquestes en altres UC.

Gestió de Respostes
- Objectiu: agrupar les operacions per crear, consultar i gestionar respostes d'enquestes.
- Actor: Respondent/Usuari (quan contesta); Creador/Administrador (quan consulta o elimina).
- Notes: Paraigües dels UC: Mostrar Respostes, Eliminar Respostes, Contestar Enquesta, Get Resposta, Eliminar Resposta.

Mostrar Respostes
- Objectiu: visualitzar totes les respostes d'un usuari a una enquesta.
- Actor: Creador/Administrador.
- Precondicions: existeixen respostes per a la parella (idE, idU).
- Flux principal: L'actor indica idE (enquesta) i idU (usuari). El sistema recupera el conjunt Respostes amb getRespostes(idE, idU) i llista, per cada idP, el tipus i el contingut. El sistema pot calcular i mostrar metadades (p. ex., nombre de respostes vs. preguntes totals).
- Alternatives: (idE, idU) inexistent → missatge "No existeixen respostes per a (idE, idU)". Enquesta sense cap resposta d'aquest usuari → llista buida o missatge informatiu.
- Postcondicions: cap canvi d'estat; només visualització.

Eliminar Respostes
- Objectiu: suprimir totes les respostes d'un usuari a una enquesta.
- Actor: Creador/Administrador.
- Precondicions: (idE, idU) vàlids; confirmació de l'actor.
- Flux principal: L'actor indica idE i idU i confirma l'operació. El sistema elimina el bloc de respostes amb EliminarRespostes(idE, idU). El sistema informa que les respostes s'han eliminat correctament.
- Alternatives: No existeixen respostes per (idE, idU) → missatge d'error i no es fa res. Cancel·lació de l'actor → no es fa res.
- Postcondicions: el conjunt Respostes(idE, idU) deixa d'existir; l'usuari queda "sense respondre" aquesta enquesta.

Contestar Enquesta
- Objectiu: permetre que un usuari respongui totes o part de les preguntes de una enquesta (crear bloc de respostes i afegir-ne una per pregunta).
- Actor: Respondent/Usuari (o Creador/Administrador actuant en nom seu).
- Precondicions: Enquesta publicada i accessible. idE existent; idU existent. L'usuari no té ja creat un bloc Respostes(idE, idU) (si ja existeix, caldria esborrar o disposar d'un "modificar resposta").
- Flux principal: El sistema crea el contenidor buit de respostes amb creaRespostes(idE, idU). Per a cada pregunta (idP) de l'enquesta: L'actor introdueix el contingut segons el tipus de pregunta.
El sistema valida:

• Text: cadenes vàlides (i obligatòries si escau). Integer: dins del rang [min, max]. Nominal única: valor dins de les opcions definides.Nominal múltiple: subconjunt d'opcions, sense duplicats i ≤ qMax. Ordinal: índex/opció dins de la llista ordenada. Si és vàlid, el sistema desa la resposta amb AfegeixResposta(idE, idU, idP, contingut). En acabar, el sistema confirma que l'enquesta s'ha contestat.
- Alternatives: Bloc ja existent per (idE, idU) → error: ja hi ha respostes; es pot cancel·lar o eliminar prèviament i tornar a començar. Validació fallida d'una resposta → es mostra l'error (fora de rang, opció inexistent, massa seleccions, camp obligatori buit, duplicats, etc.) i es demana reintroduir el valor. Pregunta no pertany a l'enquesta idE → error i no es desa. Sortida sense guardar.
- Postcondicions: totes les respostes vàlides queden registrades per (idE, idU); si hi ha preguntes no contestades i eren obligatòries, el procés es considera incomplet fins que es validin.

Contestar Pregunta
- Objectiu: capturar una resposta per a una pregunta concreta.
- Actor: Enquestat.
- Precondicions: pregunta visible; si és obligatòria, s'ha de completar abans d'enviar.
- Flux principal: l'usuari introdueix valor (numèric, opció/s, text); el sistema comprova coherència bàsica (tipus).
- Alternatives: deixar buida si no és obligatòria; format invàlid → missatge.
- Postcondicions: la resposta queda en l'esborrany/sessió.

Save
- Objectiu: desar l'estat parcial de les respostes (continuar més tard).
- Actor: Enquestat.
- Precondicions: hi ha almenys una resposta editada.
- Flux principal: el sistema desa parcialment (sessió/esborrany).
- Alternatives: error d'IO.
- Postcondicions: es pot reprendre la resposta des del mateix punt.

Eliminar Resposta
- Objectiu: esborrar la resposta d'una pregunta dins de la sessió de resposta.
- Actor: Enquestat.
- Precondicions: existeix una resposta prèviament guardada/introduïda.
- Flux principal: l'actor demana eliminar; el sistema la treu de l'esborrany.
- Alternatives: cancel·lació.
- Postcondicions: la pregunta torna a estat "sense resposta".

Get Resposta
- Objectiu: obtenir la resposta d'un usuari a una pregunta concreta d'una enquesta.
- Actor: Creador/Administrador.
- Precondicions: idE, idU, idP vàlids; la resposta existeix.
- Flux principal: L'actor indica idE, idU, idP. El sistema recupera la resposta amb GetResposta(idE, idU, idP) i en mostra el contingut i metadades (tipus, validacions si es vol).
- Alternatives: No existeix el bloc (idE, idU) → error: no hi ha respostes. No existeix la resposta per idP → error: resposta inexistent.
- Postcondicions: cap canvi d'estat; només consulta puntual.

Eliminar Resposta

- Objectiu: eliminar la resposta d'un usuari per a una pregunta concreta.
- Actor: Creador/Administrador.
- Precondicions: idE, idU, idP vàlids; la resposta existeix.
- Flux principal: L'actor indica idE, idU, idP i confirma l'operació. El sistema elimina la resposta amb EliminarResposta(idE, idU, idP). El sistema informa de l'eliminació correcta.
- Alternatives: Resposta inexistent per idP → error i no es fa res. Si en eliminar l'última resposta d'un bloc (idE, idU), el sistema pot (segons disseny) deixar el bloc buit o esborrar el bloc sencer — en la vostra implementació, el bloc es manté fins que es cridi EliminarRespostes.
- Postcondicions: la resposta esdevé inexistent; si era obligatòria, l'enquesta queda incompleta per aquest usuari fins que torni a respondre.

Gestió d'Anàlisi

- Objectiu: agrupar les operacions de configuració i execució d'anàlisi de respostes (clustering).
- Actor: Creador/Administrador.
- Notes: Paraigües dels UC: Carregar Anàlisi, Executar K-Means, Executar K-Medoids, Visualitzar Resultat d'Anàlisi (silhouette, inèrcia/cost, assignacions i llistat de clústers).

Carregar Anàlisi

- Objectiu: preparar i llançar una anàlisi de clustering sobre una enquesta existent.
- Actor: Creador/Administrador/Analista.
- Precondicions: Existeix una enquesta (idE) vàlida al sistema. Hi ha respostes registrades per a aquesta enquesta ($n \geq 1$). Paràmetres d'execució disponibles: algorisme (K-Means o K-Medoids), k ($1 < k \leq n$) i maxIter (> 0).
- Flux principal: L'actor obre el menú d'Anàlisi. L'actor selecciona l'algorisme: K-Means o K-Medoids. L'actor introdueix idE, k i maxIter. El sistema valida: idE existent; n de respostes suficient; $1 < k \leq n$; maxIter > 0. El sistema carrega el dataset de (idE) i prepara la representació: Si és K-Means: vectoritza totes les preguntes (one-hot/multi-hot per nominals, normalització per numèriques, encoder de text) i empra distància euclidiana. Si és K-Medoids: utilitza la distància de domini entre Respostes. El sistema executa l'algorisme amb els paràmetres indicats (maxIter) i obté assignacions/clústers i mètrica principal (inèrcia per K-Means o cost per K-Medoids). El sistema calcula la silhouette: Euclidiana sobre els vectors (K-Means). De domini sobre la matriu de distàncies (K-Medoids). El sistema mostra el resultat resum: Algorisme, k, temps (ms), silhouette, cost/inèrcia. Assignació de cada mostra al seu clúster i, opcionalment, el contingut de cada clúster (llista d'índexs/ids).
- Alternatives: idE inexistent → error "enquesta no trobada". Sense respostes o k fora de rang → error de validació (indicar condicions vàlides). maxIter no vàlid → error (ha de ser > 0). Falla la vectorització (K-Means) o la distància de domini (K-Medoids) → es mostra error i es cancel·la l'execució. Execució sense convergència dins de maxIter → es retorna l'últim estat amb avís.
- Postcondicions: No es modifica cap dada persistent; s'ha generat un resultat d'anàlisi disponible a pantalla (i opcionalment exportable). L'actor pot repetir l'anàlisi amb altres k/maxIter o canviar d'algorisme.

K-means (inclou → Guardar, Desplegar Resultats)

- Objectiu: aplicar el clúster K-means sobre les respostes importades.
- Actor: Creador/Analista.
- Precondicions: dades carregades (Importar Respostes) o Carregar Anàlisi.

- Flux principal: l'actor tria K i paràmetres; el sistema executa l'algorisme; inclou Guardar del model/resultats i inclou Desplegar Resultats (taules/gràfics).
- Alternatives: K invàlid; dades incompletes.
- Postcondicions: resultats calculats, mostrats i guardats si s'escull.

K-medoids (inclou → Guardar, Desplegar Resultats)
- Objectiu: aplicar el clúster K-medoids sobre les respostes importades.
- Actor: Creador/Analista.
- Precondicions: dades carregades (Importar Respostes) o Carregar Anàlisi.
- Flux principal: l'actor tria K i paràmetres; el sistema executa l'algorisme; inclou Guardar del model/resultats i inclou Desplegar Resultats (taules/gràfics).
- Alternatives: K invàlid; dades incompletes.
- Postcondicions: resultats calculats, mostrats i guardats si s'escull.

Guardar (resultats d'anàlisi)
- Objectiu: persistir models/resultats d'anàlisi (centroides, assignacions, mètriques).
- Actor: Creador/Analista.
- Precondicions: existeix un resultat recent (de K-means o K-medoids).
- Flux principal: el sistema serialitza resultats/metadades.
- Alternatives: error d'IO.
- Postcondicions: anàlisi disponible per a càrrega futura.

Desplegar Resultats
- Objectiu: mostrar resultats del clúster (taules, mètriques, gràfics bàsics).
- Actor: Creador/Analista.
- Precondicions: s'han generat resultats d'un algorisme.
- Flux principal: el sistema calcula i presenta resums (p. ex., mida de clústers, distàncies, centrals).
- Alternatives: dades insuficients.
- Postcondicions: resultats visibles; sense canvis en persistència.

NouImportCSV
- Objectiu: agrupar les operacions d'importació d'enquestes a partir d'arxius CSV.
- Actor: Creador/Administrador.
- Notes: Paraigües dels UC: Importar enquesta des de CSV, Veure enquestes importades.

Importar enquesta des de CSV
- Objectiu: crear una nova enquesta al sistema a partir de carregar un fitxer CSV resolt (capçalera = camps/preguntes; files = usuaris/respostes).
- Actor: Creador/Administrador.
- Precondicions: fitxer CSV accessible; format vàlid (columna d'identificador d'usuari i columnes de preguntes); usuari creador existent.
- Flux principal: l'actor indica ruta CSV, títol i id del creador; el sistema llegeix la capçalera, infereix tipus de preguntes (numèriques/nominals/ordinals… segons regles), crea la nova enquesta (idE) amb les preguntes, dona d'alta les respostes per cada usuari detectat i confirma.
- Alternatives: fitxer inexistent o danyat → error d'I/O; capçalera invàlida/columnes obligatòries absents → error de validació; valors no parsables → es normalitzen a "MISSING" o es reporta fila descartada.

- Postcondicions: enquesta creada amb idE nou i respostes associades; queda disponible per consulta i anàlisi.

Veure enquestes importades
- Objectiu: llistar totes les enquestes que han estat creades via importació CSV.
- Actor: Creador/Administrador.
- Precondicions: almenys una importació prèvia (en cas contrari, llista buida).
- Flux principal: l'actor sol·licita el llistat; el sistema mostra idE, títol, creador, nombre de preguntes i nombre de respostes (i, si escau, la ruta o etiqueta del CSV d'origen).
- Alternatives: cap importació registrada → missatge informatiu "No hi ha enquestes importades".
- Postcondicions: cap canvi d'estat; només consulta.

2. Descripció breu d'Atributs i Mètodes de cada classe de la capa de Domini:

En aquest apartat es descriuen les classes principals del model conceptual en la seva versió de disseny, juntament amb els seus atributs i mètodes més rellevants.
L'objectiu d'aquesta secció és detallar l'estructura i el comportament de cada classe, ressaltant-ne la seva funció dins del sistema i les relacions que mantenen entre elles.
Els *setters* i *getters* s'han omès per simplicitat, ja que no aporten informació addicional sobre la lògica de domini.

2.1 Classe Pregunta (abstracta)

Propòsit:
Defineix l'estructura comuna de totes les preguntes del sistema, incloent l'enunciat, la obligatorietat i la validació bàsica.

Atributs:
- idP: Integer, identificador intern assignat pel sistema.
- Enunciat: String, text que formula la pregunta.
- Obligatoria: Bool, indica si cal respondre la pregunta obligatòriament.

Mètodes:
- validarDefinicio() comprova que l'enunciat no sigui buit.
- modificarEnunciat(String) actualitza el text de la pregunta.
- modificarObligatoria(Bool) canvia el valor de l'obligatorietat.
- getTipus() retorna el tipus de pregunta.

2.2 Classe PreguntaInteger

Propòsit:
Representa una pregunta numèrica, que emmagatzema un valor enter dins d'un rang.

Atributs:
- maxValue: Integer, valor màxim permès.

- minValue: Integer, valor mínim permès.

Mètodes:
- validarDefinicio() comprova que el rang sigui coherent (MinValue ≤ MaxValue).
- modificarMinValue(Integer) modifica el valor del MinValue.
- modificarMaxValue(Integer) modifica el valor del MaxValue.
- getTipus() retorna el tipus de pregunta “integer”.

2.3 Classe PreguntaOrdinal

Propòsit:
Modela una pregunta amb opcions ordenades (p. ex. “poc / normal / molt”).

Atributs:
- opcions: Vector<String>, llista d’opcions definides pel creador en ordre ascendent.

Mètodes:
- validarDefinicio() verifica que la llista no sigui buida i que cap opció sigui nul·la.
- modificarOpcions(Vector<String>) substitueix les opcions actuals per unes noves.
- getTipus() retorna el tipus de pregunta “ordinal”.

2.4 Classe PreguntaNominalUnica

Propòsit:
Pregunta no ordenada on l’usuari pot triar una sola opció.

Atributs:
- opcions: Set<String>, conjunt d’opcions sense duplicats.

Mètodes:
- validarDefinicio() comprova que hi hagi almenys una opció i que totes siguin vàlides.
- modificarOpcions(Set<String>) actualitza el conjunt d’opcions disponibles.
- getTipus() retorna el tipus de pregunta “nominal_unica”.

2.5 Classe PreguntaNominalMult

Propòsit:
Pregunta no ordenada que permet seleccionar diverses opcions.

Atributs:
- opcions: Set<String>, conjunt d’opcions sense duplicats.
- qMax: Integer, màxim d’opcions que es poden triar.

Mètodes:
- validarDefinicio() comprova que qMax ≥ 1 i que no superi el nombre d’opcions disponibles i que les opcions siguin vàlides.
- modificarOpcions(Set<String>) actualitza les opcions.
- modificarQMax(Integer) actualitza el valor màxim de seleccions.
- getTipus() retorna el tipus de pregunta “nominal_multiple”.

2.6 Classe PreguntaText

Propòsit:
Pregunta oberta que permet una resposta lliure en format text.

Atributs:
No té atributs addicionals.

Mètodes:

- validarDefinicio() reusa la validació bàsica d'enunciat.
- getTipus() retorna el tipus de pregunta "string".

2.7 Classe Enquesta

La classe Enquesta representa una enquesta concreta creada per un usuari, identificada per un id, amb un títol i un conjunt ordenat de preguntes associades.
Centralitza la gestió de les preguntes d'aquesta enquesta (alta, baixa, modificació i consulta), delegant la creació i obtenció de les preguntes reals al controlador de domini.

Atributs:

- idE: Integer. Identificador únic de l'enquesta
- titol: String. Títol descriptiu de l'enquesta
- idsPreguntes: ArrayList<Integer>. Llista ordenada dels identificadors de totes les preguntes que formen part de l'enquesta
- idCreador: Integer. Identificador de l'usuari que ha creat l'enquesta
- ctrlDomini: CtrlDomini. Referència al controlador de domini, utilitzada per registrar i obtenir les preguntes i l'usuari creador

Mètodes principals (sense detallar getters/setters trivials):

- Enquesta(Integer idE, String titol, Integer idCreador, CtrlDomini ctrlDomini): constructora que inicialitza una enquesta buida i valida que idE, titol, idCreador i ctrlDomini siguin vàlids.
- getIdsPreguntes(): retorna una llista no modificable amb els ids de totes les preguntes de l'enquesta, preservant l'ordre intern.
- getNumPreguntes(): retorna el nombre de preguntes actuals de l'enquesta a partir de la mida d'idsPreguntes.
- getPreguntes(): construeix i retorna una llista d'objectes Pregunta recuperats del CtrlDomini a partir de cada id emmagatzemat a idsPreguntes.
- getCreador(): retorna l'objecte Usuari corresponent a idCreador, fent la consulta al CtrlDomini.
- setTitol(String nouTitol) i modificarTitol(String nouTitol): actualitzen el títol de l'enquesta validant que el nou títol no sigui nul ni en blanc.
- afegeixPregunta(Integer idPregunta): afegeix al final de l'enquesta una pregunta ja existent (identificada pel seu id); comprova que l'id no sigui nul i que no estigui duplicat dins d'idsPreguntes.
- creaPregunta(String tipus, String enunciat, boolean obligatoria, Object[] params, Integer index): crea una nova Pregunta del tipus indicat (integer, ordinal, nominal_unica,

nominal_multiple, text) mitjançant els mètodes de CtrlDomini, validant els paràmetres específics de cada tipus; registra la pregunta al domini, n'obté l'id i l'afegeix a idsPreguntes a la posició indicada (o al final si index és nul).

- eliminaPregunta(Integer idPregunta): elimina de l'enquesta la pregunta identificada per idPregunta si existeix. Retorna true si s'ha eliminat correctament i false si no.
- eliminaPregunta(int index): elimina i retorna l'id de la pregunta situada a la posició indicada.
- modificaPregunta(int index, Integer nouIdPregunta): substitueix la pregunta que hi ha a la posició index per una nova pregunta identificada per nouIdPregunta, validant que l'índex sigui vàlid, que el nou id no sigui nul i que no provoqui duplicats a la llista.

2.8 Classe Resposta

Propòsit:
Conté la informació d'una resposta individual a una pregunta determinada.

Atributs:

- contingut: Vector<String>, valor o conjunt de valors introduïts.

Mètodes:

- calcula_distancia(Resposta) calcula la distància entre dues respostes per comparacions o anàlisi:

Integer ("integer")

Distància relativa al rang:
$d = |v1 - v2| / (max - min)$
Fent servir max i min per normalitzar

Ordinal ("ordinal")

Les respostes són índexs dins d'una llista ordenada.
$d = |i1 - i2| / (nOpcions - 1)$

Nominal única ("nominal_unica")

$d = 0$ si resposta1 = resposta2, altrament $d = 1$.

Nominal múltiple ("nominal_multiple")

Distància de Jaccard entre conjunts d'opcions triades A i B:
$d = 1 - (|A \cap B| / |A \cup B|)$

Text lliure ("text")

Levenshtein normalitzat per la longitud màxima:
$d = Levenshtein(s1, s2) / max(|s1|, |s2|)$

2.9 Classe Respostes

Propòsit:
Agrupa un conjunt de respostes individuals provinents d'un mateix enquestat o sessió.

Atributs:
No té atributs addicionals.

Mètodes:

- ❖ calcula_distanciaRespostes(Resposta, Resposta) calcula la distància global entre dues col·leccions de respostes.

2.10 Classe Usuari

Propòsit:
Representa un usuari del sistema (creador, enquestat o analista)

Atributs:

- ❖ idU: integer, identificador de l'usuari.
- ❖ Email: String, correu electrònic de contacte (ha de contenir el caracter @ i és únic)
- ❖ Nom: String, nom identificador.

Mètodes:

- ❖ getEnquestesUsuari() : Retorna un llistat de les Enquestes que ha creat l'usuari.
- ❖ creaEnquesta(idE, títol) crea una nova enquesta com a creador.

2.11 Classe K-means

Base

- Hereta d'Algorisme (disposa de ctrlDomini, k i el dataset respostes).

Mètodes principals

- executa(maxIter, unusedFlag) : KMeansResultat

    1. **Vectoritza** totes les Respostes a un únic espai double[] consistent (one-hot/multi-hot per nominals, normalització per integers/ordinals, bossa de caràcters + longitud per text).

    2. **Inicialitza** centroides (RANDOM o K-Means++).

    3. Alterna **assignació** (punt→centroide) i **recomputació** de centroides fins a convergència o maxIter.

    4. Calcula **inèrcia** (suma de distàncies quadràtiques al centroide) i construeix els **clústers**.

Tipus de retorn

- KMeansResultat amb:

    - centroids: List<double[]> — k centroides a RD\mathbb{R}^DRD.

    - clusters: Map<Integer, List<Integer>> — idCluster → índexs de mostres.

- assignacio: int[] — per a cada mostra, el cluster assignat.
- inertia: double — suma d'errors quadràtics intra-cluster.

Helpers interns típics

- initRandom(...) / initKMeansPlusPlus(...) — estratègies d'inicialització.
- assignPoints(...) — pas d'assignació (minimitza distància L2 al centroide).
- recomputeCentroids(...) — mitjana per dimensió; resinsembrat si un clúster queda buit.
- sqDist(a,b) — distància quadràtica (sense arrel) per eficiència.

Notes de disseny

- Es desacobla la **vectorització** (via un component Vectoritzador) de l'algorisme K-Means, per tal que:
  - l'espai sigui **fix** i coherent per a tot el dataset,
  - es puguin afegir fàcilment noves codificacions (p. ex. TF-IDF a futur),
  - i la **silhouette euclidiana** de Analisi sigui comparable amb el mateix embedding.

2.12 Classe Algorisme (abstracta)

Propòsit:
Centralitza el comportament comú dels algorismes de clustering del domini (K-means i K-medoids), gestionant el conjunt de respostes a una enquesta, el nombre de clústers k i la validació estructural de les dades abans d'executar l'algorisme.

Atributs:

ctrlDomini: CtrlDomini
Referència al controlador de domini, utilitzada per accedir a altres entitats de domini si cal. Es valida que no sigui nul al constructor.

idAlgorisme: int
Identificador intern de l'algorisme, no negatiu, que permet distingir diferents configuracions o execucions dins del domini.

k: int
Nombre de clústers a generar. Ha de ser estrictament positiu; qualsevol valor ≤ 0 provoca una IllegalArgumentException.

respostes: List<Respostes>
Conjunt de Respostes sobre el qual s'executarà l'algorisme. Es guarda com a còpia interna i es valida que no sigui nul ni buit i que tingui estructura coherent.

Constructors principals:

Algorisme(CtrlDomini ctrlDomini, int idAlgorisme, int k)
Inicialitza l'algorisme amb el controlador de domini, l'identificador i el nombre de clústers. Comprova que ctrlDomini no sigui nul, que idAlgorisme sigui no negatiu i que k sigui positiu.

Algorisme(CtrlDomini ctrlDomini, int idAlgorisme, int k, List<Respostes> respostes)
Reutilitza el constructor anterior i, a més, estableix la llista de Respostes cridant setRespostes(respostes), fent totes les comprovacions estructurals necessàries.

Mètodes públics principals:

int getIdAlgorisme()
Retorna l'identificador intern associat a aquest algorisme.

int getK()
Retorna el valor actual de k.

void setK(int k)
Modifica el nombre de clústers, comprovant que k > 0.

void setRespostes(List<Respostes> respostes)
Estableix el conjunt de Respostes sobre el qual treballarà l'algorisme.
Comprova que la llista no sigui nul·la ni buida.

protected List<Respostes> getRespostes()
Retorna la llista interna de Respostes.

abstract Object executa(int maxIter, boolean useAllPairsSwap)

Operació abstracta que defineix el contracte d'execució dels algorismes de clustering.

Mètodes protegits de validació:

protected final void validaEstructura(List<Respostes> respostes)

Verifica que totes les instàncies de Respostes dins la llista siguin compatibles entre si abans d'executar qualsevol algorisme.

2.12 Classe K-medoids

Propòsit:
La classe KMedoids implementa l'algorisme de clustering k-medoids (PAM) sobre un conjunt de Respostes, utilitzant la distància definida al domini per agrupar usuaris amb patrons de resposta similars.
Hereta de la classe abstracta Algorisme.

Atributs principals:
La classe no afegeix atributs d'instància propis, sinó que reutilitza els de la classe base Algorisme. Disposa d'una classe interna estàtica Resultat que encapsula el resultat final del clustering: la llista de medoids, el mapa de clusters i el cost total de la solució.

Classe interna Resultat:

- medoids: List<Integer>. Índexs dins del dataset (llista de Respostes) que han estat seleccionats com a medoids
- clusters: Map<Integer, List<Integer>>. Per a cada medoid (índex del dataset), guarda la llista d'índexs de les Respostes assignades a aquest clúster.
- costTotal: double. Suma de les distàncies de cada element al seu medoid, d'acord amb la matriu de distàncies precomputada.

Constructors:

- KMedoids(CtrlDomini ctrlDomini, int idAlgorisme, int k, List<Respostes> respostes): inicialitza un algorisme k-medoids amb un controlador de domini, un identificador d'algorisme, el nombre de clusters k i el dataset complet de Respostes sobre el qual s'ha de clusteritzar.
- KMedoids(CtrlDomini ctrlDomini, int idAlgorisme, int k): inicialitza l'algorisme amb el mateix conjunt de paràmetres però deixant el dataset per establir posteriorment a la classe base.

Mètode principal:

- Object executa(int maxIter, boolean useAllPairsSwap): és el punt d'entrada de l'algorisme. Comprova que k sigui positiu i no superi el nombre de respostes.
  A cada iteració calcula el millor intercanvi possible entre un medoid i un element no medoid, actualitza els medoids si el cost millora i recalcula les assignacions fins que no hi ha millora o s'arriba al màxim d'iteracions. Al final construeix el mapa de clusters i retorna un objecte Resultat amb medoids, clusters i cost total.

Mètodes privats més rellevants:

- precomputaMatriuDistancies(List<Respostes> respostes): construeix una matriu simètrica D on D[i][j] és la distància mitjana entre les Respostes i i j; reutilitza la funció distanciaEntreRespostes i evita recomputacions futures.
- distanciaEntreRespostes(Respostes a, Respostes b): recorre totes les preguntes i, per a cada parell de Resposta, crida calcula_distanciaRespostes a la classe Resposta, normalitza cada distància a l'interva i retorna la mitjana de totes les distàncies.
- inicialitzaMedoids(int n, int k): selecciona els índexs inicials dels medoids; per k = 1 tria el primer element, i per k > 1 reparteix els medoids de manera aproximadament uniforme sobre els n elements, afegint elements addicionals si cal per garantir que es tenen k medoids diferents.
- assignaTots(double[][] D, List<Integer> medoids): per a cada element del dataset selecciona el medoid més proper segons la matriu D i construeix un mapa que assigna cada índex al seu medoid.
- costTotal(double[][] D, List<Integer> medoids, Map<Integer, Integer> assignacio): calcula el cost total de la solució sumant D[i][m] per a cada element i assignat al seu medoid m.
- deltaSwap(double[][] D, List<Integer> medoids, Map<Integer, Integer> assignacio, int mOut, int hIn): estima la variació de cost que es produiria en intercanviar un medoid existent mOut per un element no medoid hIn, utilitzant per a cada punt la millor i segona millor distància actual.
- construeixClusters(List<Integer> medoids, Map<Integer, Integer> assignacio): crea, a partir de l'assignació d'elements a medoids, un mapa des de medoid a llista d'índexs de membres del clúster, emprat finalment per generar l'objecte Resultat.

2.13 Classe Anàlisi

Atributs clau

- ctrl: CtrlDomini — façana per obtenir metadades (tipus de preguntes, rangs, etc.) i accedir a respostes quan cal calcular distàncies.

Mètodes principals

- executaKMeans(dades, k, maxIter) : ResultatAnalisi
  Orquestra una execució de K-Means sobre el conjunt dades (llista de Respostes). Crea l'algorisme, injecta el dataset, mesura el temps, recupera assignacions/clústers i, si escau, calcula **silhouette euclidiana** al mateix espai vectoritzat que usa KMeans. Emplena ResultatAnalisi amb algorisme="KMeans", k, millis, assignacio, clusters, qualitatSilhouette i inerciaOCost (inèrcia).

- executaKMedoids(dades, k, useAllPairsSwap, maxIter) : ResultatAnalisi
  Idèntic flux per a K-Medoids, recuperant cost i calculant **silhouette de domini** amb la distància pròpia de les respostes (la teva calcula_distanciaRespostes).

- calculaSilhouette(dades, assign) : double
  Calcula la silhouette mitjana [−1,1][-1,1][−1,1] utilitzant **distància de domini** (agregant la

distància pregunta a pregunta, normalitzada [0,1][0,1][0,1]).

- calculaSilhouetteEuclidiana(X, assign) : double
  Silhouette sobre vectors double[] (espai euclidià) — s'utilitza per validar K-Means quan el pipeline treballa amb one-hot/multi-hot/text-bag-of-chars, etc.

Notes de disseny

- ResultatAnalisi és un DTO homogeni per a qualsevol algorisme: encapsula cronometratge, k, assignació, clústers, mètrica de qualitat i inèrcia/cost. El camp rawResult conserva el resultat específic (p. ex. KMeansResultat) per si es vol inspeccionar després.

2.14 Classe CtrlDomini

Propòsit:
CtrlDomini és el controlador de domini que actua com a punt d'entrada a la lògica del sistema. Manté les col·leccions principals (usuaris, enquestes, preguntes, respostes) i coordina la persistència mitjançant CtrlPersistencia.

Atributs rellevants (sense getters/setters):

- CtrlPersistencia ctrlPersistencia: connexió amb la capa de persistència.
- ArrayList<Enquesta> enquestes: llista d'enquestes existents.
- ArrayList<Usuari> usuaris: llista d'usuaris existents.
- ArrayList<Pregunta> preguntes: llista de preguntes existents.
- Map<Integer, Map<Integer, Map<Integer, Resposta>>> respostes: estructura jeràrquica per guardar respostes (idE → idU → idP → Resposta).
- Comptadors d'IDs (countP, countE, countU) per assegurar identificadors únics.

Mètodes principals:

- CtrlDomini(): inicialitza les estructures i carrega usuaris des de persistència.
- tancarAplicacio(): prepara el tancament de l'aplicació (p. ex. desant canvis necessaris).
- creaEnquesta(String titol, Integer idUsuari): crea una enquesta associada a un creador existent.
- getEnquesta(Integer idE): retorna una enquesta per id.
- existeixUsuari(Integer idUsuari): comprova si un usuari existeix.
- afegeixPreguntaAEnquesta(Integer idE, Integer idP): associa una pregunta existent a una enquesta.
- eliminaPreguntaAEnquesta(Integer idE, Integer idP): elimina l'associació pregunta–enquesta.

- setTitolEnquesta(Integer idE, String nouTitol): modifica el títol d'una enquesta.
- getInfoEnquestes(): retorna informació resumida d'enquestes (per mostrar a presentació).
- eliminarEnquesta(Integer idE): elimina una enquesta.
- getInfoPreguntes(): retorna informació resumida de preguntes (id, tipus, enunciat).
- eliminarPregunta(Integer idP): elimina una pregunta.
- Creació de preguntes:
    - creaPreguntaText(...)
    - creaPreguntaInteger(...)
    - creaPreguntaNominalUnica(...)
    - creaPreguntaNominalMult(...)
    - creaPreguntaOrdinal(...)
- Consulta: getPreguntes(), getPregunta(Integer idP)
- Gestió d'usuaris: creaUsuari(...), getUsuari(Integer idU), getallUsuaris()
- Enquestes d'un usuari: getEnquestesUsuari(Integer idU)
- Gestió de respostes:
    - creaRespostes(Integer idE, Integer idU)
    - getRespostes(Integer idE, Integer idU)
    - getLlistaRespostes(Integer idE, Integer idU)
    - AfegeixResposta(Integer idE, Integer idU, Integer idP, List<String> contingut)
    - GetResposta(Integer idE, Integer idU, Integer idP)
    - EliminarResposta(Integer idE, Integer idU, Integer idP)
    - EliminarRespostes(Integer idE, Integer idU)
    - getRespostesEnquesta(int idE)
- Importació CSV: importarEnquestaDesdeCSV(String filePath, String titolEnquesta, Integer idCreador)
- Execució d'algorismes sobre una enquesta:
    - executarKMeansSobreEnquesta(int idEnquesta, int k, int maxIter)
    - executarKMedoidsSobreEnquesta(int idEnquesta, int k, int maxIter)

2.15 Classe Algorisme (abstracta)

Propòsit:
Defineix una interfície comuna pels algorismes de clustering del projecte i encapsula el resultat en forma de clústers i representants (centroides o medoides).

Atributs rellevants:

- int[] clusters: assignació de clúster per a cada element (índex → clúster).

Mètodes principals:

- agrupar(double[][] data, int k, int maxIter): mètode abstracte que executa l'algorisme i omple clusters.
- getClusters(): retorna el vector d'assignacions.
- getClusterOf(int index): retorna el clúster assignat a un element.
- getCentroids(): per implementacions que treballen amb centroides (ex. K-means).
- getMedoids(): per implementacions que treballen amb medoides (ex. K-medoids).

2.16 Classe Vectoritzador

Propòsit:
Transforma respostes (en format textual) a representacions numèriques (double[]) per poder aplicar K-means/K-medoids de manera homogènia, gestionant diferents tipus de pregunta.

Atributs rellevants:

- Map<Integer, Encodador> encoders: associació idPregunta → codificador.
- int dim: dimensionalitat final del vector generat.
- List<Integer> ordrePreguntes: ordre fix de preguntes per construir vectors consistents.

Mètodes principals:

- build(List<Pregunta> preguntes, List<Respostes> respostes): construeix els codificadors i calcula la dimensionalitat total.
- dim(): retorna la dimensionalitat (dim) del vector final.
- encode(Respostes r, List<Pregunta> preguntes): genera un double[] per a un conjunt de respostes, seguint l'ordre i codificadors preparats.

2.17 Classe LoanCSVImporter

Propòsit:
Classe utilitària per llegir un CSV (dataset de loans) i convertir-lo en una estructura numèrica (double[][]) apta per executar algorismes de clustering.

Atributs rellevants:

- No manté estat persistent rellevant; opera a partir de fitxers CSV i retorna dades.

Mètodes principals:

- importFromCSV(String path): llegeix el CSV, ignora la capçalera i retorna les dades en una matriu double[][].

3. Relació de les classes implementades per cada membre de l'equip.

A continuació es presenta la relació de classes implementades per cada membre de l'equip durant el primer lliurament del projecte. Cada membre ha estat responsable de la implementació, validació i prova de les classes que se li han assignat.

Membre de l'equip	**Classes implementades**	**Responsabilitats principals**
Igor Bolige	Pregunta, PreguntaInteger, PreguntaOrdinal, PreguntaNominalUnica, PreguntaNominalMult, PreguntaText, Driver Pregunta, TestPregunta i Test de totes les subclasses de Pregunta	Implementació del model de preguntes i validació de definicions.
Adrià Aguilar	CtrlDomini, DriverCsvImporter.java, DriverUsuari, Usuari, DriverDomain, Main, TestUsuari, DriverDomain	Gestió d'usuaris, creació d'enquestes a traves d'usuari i interacció amb preguntes. Importar enquestes en csv i mantenir les instàncies de les clases al CtrlDomini. Guardar les instancies dels drivers específics i fer el menú principal o es crida a cada driver.
Arnau	Enquesta, K-Medoids, Algorisme, Test Enquesta, Test K-Medoids	Gestió de les enquestes. Desenvolupament de k-medoid. Desenvolupament de la classe Algorisme
Roger Corcoles	Resposta, Respostes, LoanCSVImporter, DriverCSVClustering, DriverResposta, TestResposta, TestRespostes CsvImporter GestorFitxersUsuari	Estructuració i tractament de les respostes individuals i conjuntes. Encarregat de llegir arxius csv i passar a format intern. Persistencia de la classe usuari.

Joaquim Fuentes	Anàlisi, K-Means, Vectoritzador, Driver Anàlisi,Driver K-Means, Test Anàlisi, Test K-Means, Driver	Desenvolupament dels algorismes d'anàlisi i clustering.

4. Estructures de dades i algorismes utilitzats

En aquest apartat es descriuen les estructures de dades i els algorismes utilitzats per implementar les funcionalitats principals del model. (gestió d'usuaris, enquestes, preguntes, respostes, importació CSV i anàlisi per clustering).
L'objectiu és justificar les decisions de disseny preses, mostrant com aquestes estructures permeten una gestió eficient de les preguntes, respostes i operacions d'anàlisi dins del sistema.

4.1 Estructures de dades

Utilitzarem aquest patró per definir totes les estructures
• **Element del sistema**
- **Estructura utilitzada**
- **Justificació**

• Preguntes ordinals
- Estructura: Vector<String> (opcions)
- Justificació: manté l'ordre definit pel creador i permet accés per índex.

• Preguntes nominals (única i múltiple)
- Estructura: Set<String> (opcions) i (en el cas múltiple) un límit qMax.
- Justificació: evita duplicats; no és necessari imposar ordre en les opcions.

• Enquesta
- Estructura: idsPreguntes: ArrayList<Integer> (llista ordenada d'IDs de preguntes)
- Justificació: es guarda l'ordre de les preguntes de l'enquesta i es recuperen els objectes Pregunta quan cal a través del controlador de domini, facilitant el recorregut seqüencial.

• Resposta
- Estructura: contingut: Vector<String>
- Justificació: permet emmagatzemar 1 o més valors (p. ex. nominal múltiple) de forma homogènia.

• Dades principals del domini (CtrlDomini)
- Estructures:
  - ArrayList<Enquesta> enquestes
  - ArrayList<Usuari> usuaris
  - ArrayList<Pregunta> preguntes
  - Map<Integer, Map<Integer, Map<Integer, Resposta>>> respostes (idE → idU → idP → Resposta)
  - Comptadors d'IDs (countP, countE, countU)

- Justificació: el Map jeràrquic permet localitzar ràpidament una resposta concreta d'una enquesta/usuari/pregunta i mantenir l'organització per enquesta i usuari.

• Vectorització per aplicar K-means (Vectoritzador)
- Estructures:
  - Map<Integer, Encodador> encoders (idPregunta → codificador)
  - List<Integer> ordrePreguntes (ordre fix)
  - int dim (dimensionalitat final)
- Justificació: garanteix que tots els vectors double[] tenen el mateix espai i ordre de variables.

• Resultats d'anàlisi
- Estructura: ResultatAnalisi (DTO amb k, assignació, clústers, mètrica de qualitat, temps i inèrcia/cost, etc.)
- Justificació: unifica el format de sortida independentment de l'algorisme (K-means o K-medoids).

• Persistència (fitxers CSV)
- Estructura: ArrayList<ArrayList<String>> per representar el CSV com a files i camps.
- Justificació: format intermedi simple per llegir/escriure i transformar cap al domini.

• Estructures internes dels algorismes de clustering
- K-means:
  - List<double[]> dades
  - List<double[]> centroides
  - int[] assignacio
  - Map<Integer, List<Integer>> clusters
- K-medoids:
  - double[][] matriuDistancies
  - List<Integer> medoids
  - Map<Integer, List<Integer>> clusters
- Justificació: K-means treballa amb vectors numèrics i mitjanes; K-medoids reutilitza una matriu de distàncies i opera amb índexs/medoids, reduint recalculs.

4.2 Algorismes principals

• Validació de definicions
- Les classes de pregunta inclouen validacions (p. ex. rangs coherents a numèriques, opcions vàlides a ordinals/nominals, etc.) abans de permetre crear o modificar elements.

• Importació i persistència CSV
- Càrrega d'usuaris/enquestes a partir de CSV: lectura línia a línia, ignorar línies buides, separar camps i construir una estructura ArrayList<ArrayList<String>>.
- Guardat a CSV: serialització de cada fila ajuntant camps i escrivint-los al fitxer.

• Distància de domini entre respostes (per K-medoids i silhouette de domini)
- Distància global agregant distàncies locals pregunta a pregunta i normalitzant-les.
- Distàncies locals destacades:

- Nominal múltiple: distància de Jaccard entre conjunts d'opcions.
- Text: distància de Levenshtein normalitzada per la longitud màxima.

• Vectorització (per K-means i silhouette euclidiana)
- Transformació de Respostes a double[] consistent:
  - one-hot / multi-hot per preguntes nominals
  - normalització per numèriques/ordinals
  - representació per text (p. ex. bossa de caràcters + longitud), segons el pipeline definit al projecte.

• K-means
- Passos principals: vectoritzar → inicialitzar centroides (Random o K-means++) → iterar assignació i recomputació de centroides fins convergència o maxIter → calcular inèrcia i construir clústers.

• K-medoids
- Passos principals: precomputar matriu de distàncies → inicialitzar medoids → assignar punts al medoid més proper → calcular cost → bucle d'intercanvis (swaps) per millorar el cost → construir clústers finals.

• Qualitat del clustering (silhouette)
- Silhouette euclidiana sobre vectors double[] (per validar K-means).
- Silhouette de domini utilitzant la distància pròpia del projecte (per K-medoids / distància de respostes).

5. Descripció d'algorismes i Pseudocodi

5.1 K-medoids

5.1.1 Descripció

K-Medoids és un algorisme de particionat que vol agrupar n observacions en k clusters minimitzant el cost total: la suma de distàncies de cada punt al seu representant, anomenat medoide. A diferència de K-Means, que utilitza centroides (mitjanes que no necessàriament pertanyen al conjunt de dades) i acostuma a dependre de la distància euclidiana, K-Medoids sempre tria com a centres punts reals del dataset i admet mètriques arbitràries; per això és més robust a valors atípics i especialment útil amb dades categòriques o mixtes.

El codi comença precomputant una matriu de distàncies entre totes les parelles de respostes. Tenim que la distància entre dos usuaris és el promig de distàncies pregunta a pregunta, acotades entre 0 i 1 . Després s'inicialitzen els medoids triant k índexs repartits al rang de $[0..n-1]$. Amb aquests medoids inicials, s'assigna cada punt al medoid més proper i es calcula el cost inicial com la suma D[i][m] de totes les assignacions.

El nucli d'optimització és un bucle d'intercanvis. Per a cada medoid m i cada no-medoid h es calcula el guany potencial de substituir m per h.. Si la suma de canvis és negativa, el cost baixa, es fa l'intercanvi, es reassignen tots els punts i s'actualitza el cost. Aquest procés es repeteix fins que no hi ha millora o s'assoleix el màxim d'iteracions.

La conté la llista de medoids finals, el mapa de clústers (cada medoid amb els índexs dels seus membres) i el cost total. L'algorisme pressuposa que totes les instàncies tenen el mateix nombre i ordre de característiques (mateixa enquesta i mateixes preguntes alineades), i que la mètrica per pregunta està ben normalitzada.

5.1.2 Pseudocodi k-medoids

```
ALGORISME K_MEDOIDS(respostes, k, maxIter, useAllPairsSwap)

  //  Paràmetres bàsics
  n = mida(respostes)

  SI (k ≤ 0 O k > n) ALESHORES
    llença error "k fora de rang"
  FI SI


  //1. Construir la matriu de distàncies D[n][n]
  crea matriu D de mida n × n

  PER i DE 0 FINS n-1 FER
    D[i][i] = 0

    PER j DE i+1 FINS n-1 FER
      // Calcular distància mitjana entre resposta i i resposta j
      suma = 0
      numPreguntes = nombre de preguntes de cada enquesta

      PER p DE 0 FINS numPreguntes-1 FER
        // distància entre la resposta de l'usuari i i la de j a la pregunta p
        d_p = distanciaResposta(respostes[i], respostes[j], p)
        suma = suma + d_p
      FI PER

      d = suma / numPreguntes

      SI d < 0 ALESHORES d = 0 FI SI
      SI d > 1 ALESHORES d = 1 FI SI

      D[i][j] = d
      D[j][i] = d       // matriu simètrica
    FI PER
  FI PER
```

```
//2. Inicialitzar els medoids
medoids = llista buida

SI k = 1 ALESHORES
   afegeix 0 a medoids
SI NO
   pas = (n - 1) / (k - 1)    // repartim els medoids al llarg del conjunt

   PER i DE 0 FINS k-1 FER
      idx = arrodoneix(i * pas)
      SI idx no és a medoids ALESHORES
         afegeix idx a medoids
      FI SI
   FI PER

   //Si falten medoids per culpa dels arrodoniments, omplim amb els primers índexs lliures
   cur = 0
   MENTRE mida(medoids) < k I cur < n FER
      SI cur no és a medoids ALESHORES
         afegeix cur a medoids
      FI SI
      cur = cur + 1
   FI MENTRE
FI SI

// 3. Assignació inicial de cada punt al medoid més proper
crea vector assignacio[0..n-1]

PER i DE 0 FINS n-1 FER
   millorMedoid = -1
   millorDist = +∞

   PER CADA m A medoids FER
      d = D[i][m]
      SI d < millorDist ALESHORES
         millorDist = d
         millorMedoid = m
      FI SI
   FI PER

   assignacio[i] = millorMedoid
FI PER

// 4. Càlcul del cost inicial
cost = 0.0
PER i DE 0 FINS n-1 FER
   cost = cost + D[i][assignacio[i]]
```

```
FI PER

// 5. Bucle principal de millora per "swaps"
millora = CERT
iter = 0

// vectors auxiliars per al càlcul dels swaps
crea vectors bestDist[0..n-1], secondBestDist[0..n-1], bestMedoid[0..n-1]
MENTRE (millora I iter < maxIter) FER
   millora = FALS
   millorDelta = 0.0
   millorMedoid = -1
   millorNoMedoid = -1

   conjuntMedoids = conjunt(medoids)


   //5.1 Precomputar millor i segona millor distància per a cada punt
   PER i DE 0 FINS n-1 FER
      mAct = assignacio[i]
      dBest = D[i][mAct]
      dSecond = +∞

      PER CADA m A medoids FER
         SI m ≠ mAct ALESHORES
            dTmp = D[i][m]
            SI dTmp < dSecond ALESHORES
               dSecond = dTmp
            FI SI
         FI SI
      FI PER

      bestDist[i] = dBest
      secondBestDist[i] = dSecond
      bestMedoid[i] = mAct
   FI PER

   // 5.2 Explorar tots els swaps "medoid per no-medoid"
   PER CADA mOut A còpia(medoids) FER

      PER h DE 0 FINS n-1 FER
         SI h ∈ conjuntMedoids ALESHORES
            continua amb el següent h       // només volem punts que no són medoid
         FI SI

         //  5.2.1 Calcular delta de cost d'aquest swap concret mOut per h
         delta = 0.0
```

```
        PER i DE 0 FINS n-1 FER
            dBest = bestDist[i]
            dSecond = secondBestDist[i]
            curBest = bestMedoid[i]
            dToHin = D[i][h]

            SI curBest ≠ mOut ALESHORES
                // El medoid actual no desapareix
                nouCost = mínim(dBest, dToHin)
            SI NO
                // El medoid actual desapareix, escollim entre el nou medoid h i el segon millor
                nouCost = mínim(dToHin, dSecond)
            FI SI

            delta = delta + (nouCost - dBest)
        FI PER


        //  5.2.2 Actualitzar el millor swap trobat fins ara
        SI delta < millorDelta ALESHORES
            millorDelta = delta
            millorMedoid = mOut
            millorNoMedoid = h

            SI (NO useAllPairsSwap) ALESHORES
                // Ens quedem amb el primer swap que millora
                SURT del bucle de h
            FI SI
        FI SI

    FI PER   // fi del bucle de h

    SI (NO useAllPairsSwap I millorMedoid ≠ -1) ALESHORES
        SURT del bucle de mOut
    FI SI

FI PER   // fi del bucle de mOut


// 5.3 Si hi ha algun swap que millori el cost, aplicar-lo
SI millorMedoid ≠ -1 ALESHORES
    // Actualitzar el conjunt de medoids
    elimina millorMedoid de medoids
    afegeix millorNoMedoid a medoids

    // Recalcular assignació de tots els punts als nous medoids
    PER i DE 0 FINS n-1 FER
        millorMedoidPunt = -1
```

```
            millorDistPunt = +∞

            PER CADA m A medoids FER
               d = D[i][m]
               SI d < millorDistPunt ALESHORES
                  millorDistPunt = d
                  millorMedoidPunt = m
               FI SI
            FI PER

            assignacio[i] = millorMedoidPunt
         FI PER

         // Actualitzar el cost total amb la millora trobada
         cost = cost + millorDelta

         millora = CERT
      FI SI

      iter = iter + 1
   FI MENTRE


   // 6. Construir els clústers finals a partir de l'assignació
   crea diccionari clusters   // clau: medoid, valor: llista d'índexs assignats

   PER CADA m A medoids FER
      clusters[m] = llista buida
   FI PER

   PER i DE 0 FINS n-1 FER
      m = assignacio[i]
      afegeix i a clusters[m]
   FI PER


   // 7. Resultat
   RETORNA (medoids, clusters, cost)

FI ALGORISME
```

6. Testos

Clase Usuari:

setUp:

- El mètode s'executa abans de cada test per establir l'estat inicial, es a dir, inicialitza una instancia de CtrlDomini (necessari per a la creació d'usuaris i la gestió d'enquestes). Crea un objecte Usuari amb ID 0, nom "Pere" i email "pere@exemple.cat" utilitzant el CtrlDomini i l'assigna a la variable privada del test usuari.

testConstructoraValida:
- Objectiu: Comprovar la correcta inicialització de l'usuari.
- Resultat esperat: L'ID és 0, el nom és "Pere" i l'email és "pere@exemple.cat".

testConstructoraNomNul:
- Objectiu: Comprovar la validació del nom nul en la creació.
- Resultat esperat: Llança IllegalArgumentException.

testConstructoraNomBuit:
- Objectiu: Comprovar la validació del nom buit o només espais.
- Resultat esperat: Llança IllegalArgumentException.

testConstructoraEmailNul:
- Objectiu: Comprovar la validació de l'email nul en la creació.
- Resultat esperat: Llança IllegalArgumentException.

testConstructoraEmailBuit:
- Objectiu: Comprovar la validació de l'email buit o només espais.
- Resultat esperat: Llança IllegalArgumentException.

testConstructoraEmailInvalid:
- Objectiu: Comprovar la validació de l'email sense '@'.
- Resultat esperat: Llança IllegalArgumentException.

testConstructoraEmailDuplicat:
- Objectiu: Comprobar la restricció de duplicitat d'email (implícita al CtrlDomini).
- Resultat esperat: Llança IllegalArgumentException (assumint que CtrlDomini gestiona aquesta restricció).

testSetNomValid:
- Objectiu: Comprovar el canvi de nom amb un valor vàlid.
- Resultat esperat: El nom de l'usuari canvia a "NouNom".

testSetNomNul:
- Objectiu: Comprovar la validació quan el nou nom és nul.
- Resultat esperat: Llança IllegalArgumentException.

testSetNomBuit:
- Objectiu: Comprovar la validació quan el nou nom és buit o espais.
- Resultat esperat: Llança IllegalArgumentException.

testSetEmailValid:

- Objectiu: Comprovar el canvi d'email amb un valor vàlid.
- Resultat esperat: L'email de l'usuari canvia a "nou@exemple.cat".

testSetEmailNul:
- Objectiu: Comprovar la validació quan el nou email és nul.
- Resultat esperat: Llança IllegalArgumentException.

testSetEmailBuit:
- Objectiu: Comprovar la validació quan el nou email és buit o espais.
- Resultat esperat: Llança IllegalArgumentException.

testSetEmailInvalid:
- Objectiu: Comprovar la validació quan el nou email no conté '@'.
- Resultat esperat: Llança IllegalArgumentException.

testCreaEnquestaValida:
- Objectiu: Comprovar la creació d'una enquesta amb títol vàlid.
- Resultat esperat: L'enquesta es crea correctament, no és nul·la, té el títol assignat i el seu ID de creador coincideix amb l'ID de l'usuari.

testCreaEnquestaTitolNul:
- Objectiu: Comprovar la validació del títol nul.
- Resultat esperat: Llança IllegalArgumentException.

testCreaEnquestaTitolBuit:
- Objectiu: Comprovar la validació del títol buit o només espais.
- Resultat esperat: Llança IllegalArgumentException.

testGetEnquestesUsuariSenseEnquestes:
- Objectiu: Comprovar la recuperació d'enquestes quan l'usuari no n'ha creat cap.
- Resultat esperat: Llança IllegalStateException amb el missatge d'error esperat ("Aquest usuari no ha creat cap enquesta.").

testGetEnquestesUsuariAmbEnquestes:
- Objectiu: Comprovar la recuperació de múltiples enquestes creades.
- Resultat esperat: Retorna una llista amb 2 enquestes, i ambdues enquestes creades es troben a la llista.

testGetEnquestaUsuariValida:
- Objectiu: Comprovar la recuperació d'una enquesta específica creada per l'usuari.
- Resultat esperat: L'enquesta es recupera correctament, no és nul·la, i l'ID i el títol coincideixen.

testGetEnquestaUsuariIdNul:
- Objectiu: Comprovar la validació de l'ID d'enquesta nul.
- Resultat esperat: Llança IllegalArgumentException.

testGetEnquestaUsuariNoAccessible:

- Objectiu: Comprovar que un usuari no pot accedir a una enquesta creada per un altre.
- Resultat esperat: Llança IllegalArgumentException amb el missatge d'error de no tenir accés.

testGetId:
- Objectiu: Comprovar la recuperació de l'ID.
- Resultat esperat: Retorna l'ID assignat (0).

testGetNom:
- Objectiu: Comprovar la recuperació del nom.
- Resultat esperat: Retorna el nom "Pere".

testGetEmail:
- Objectiu: Comprovar la recuperació de l'email.
- Resultat esperat: Retorna l'email "pere@exemple.cat".

testMultipleUsuarisIdsIncrementals:
- Objectiu: Comprovar que l'ID s'incrementa correctament en crear nous usuaris.
- Resultat esperat: Els IDs dels usuaris creats són 0, 1 i 2, respectivament.

Clase Resposta:

setUp:
- El mètode s'executa abans de cada test per establir l'estat inicial. Inicialitza una instància de CtrlDomini, necessària per crear preguntes (Integer, Ordinal, Nominal única, Nominal múltiple i Text) i perquè la classe Resposta pugui consultar la informació de la pregunta associada quan calcula distàncies. No crea usuaris ni enquestes, ja que aquests tests comproven funcionalitat interna de Resposta i no depenen de persistència.

testGettersBasicsOk:
- Objectiu: Comprovar que la constructora de Resposta inicialitza correctament els atributs bàsics.
- Resultat esperat: getIdPregunta() retorna l'ID de pregunta assignat i getContingut() retorna exactament la llista de contingut passada a la constructora.

testDistanciaIntegerNormalitzada:
- Objectiu: Comprovar el càlcul de la distància entre respostes d'una pregunta de tipus Integer, amb normalització dins el rang definit.
- Resultat esperat:
    - La distància entre "0" i "100" és 1.0.
    - La distància entre "0" i "50" és 0.5.
    - La distància entre dues respostes idèntiques ("50" i "50") és 0.0.

testDistanciaOrdinalNormalitzada:

- Objectiu: Comprovar el càlcul de distància en respostes d'una pregunta Ordinal, normalitzant la diferència de posicions en funció del nombre d'opcions.
- Resultat esperat:

    - Amb 4 opcions (índex 0..3), la distància entre "0" i "3" és 1.0.

    - La distància entre "1" i "2" és 1/3.

testDistanciaNominalUnica:
- Objectiu: Comprovar el càlcul de distància per una pregunta Nominal Única (selecció d'un únic valor).
- Resultat esperat:

    - Si les respostes tenen el mateix valor (p.ex. "verd" i "verd"), la distància és 0.0.

    - Si les respostes tenen valors diferents (p.ex. "verd" i "roig"), la distància és 1.0.

testDistanciaNominalMultipleJaccard:
- Objectiu: Comprovar el càlcul de distància per una pregunta Nominal Múltiple utilitzant la distància de Jaccard (basada en intersecció i unió dels valors seleccionats).
- Resultat esperat:

    - Per {musica, esport} i {esport, jocs}, la intersecció té mida 1 i la unió mida 3, per tant la similitud és 1/3 i la distància és 1 - 1/3 = 2/3.

    - Si es comparen dues respostes amb exactament el mateix conjunt de valors, la distància és 0.0.

testDistanciaTextLevenshteinNormalitzada:
- Objectiu: Comprovar el càlcul de distància en preguntes de tipus Text mitjançant Levenshtein normalitzat.
- Resultat esperat:

    - Entre "hola" i "ola", la distància és 0.25 (una edició sobre longitud màxima 4).

    - Entre dos textos idèntics ("hola" i "hola"), la distància és 0.0.

Clase Respostes:

setUp:
- El conjunt de proves utilitza dos mètodes de preparació. Primer, netejaPersistencia() (annotat amb @BeforeClass) s'executa un sol cop abans de tots els tests i elimina el fitxer DATA/FitxerUsuaris.csv si existeix, per assegurar que no hi hagi dades persistides d'execucions anteriors que provoquin duplicats d'email o IDs inesperats. Després, setup() (annotat amb @Before) s'executa abans de cada test: inicialitza una instància nova de CtrlDomini, crea dos usuaris reals ("CreadorTest" i "ContestantTest") i guarda els seus IDs (idCreador i idContestant). Finalment, crea una enquesta ("Enquesta Test") associada al

creador i en desa l'ID (idE). Això garanteix que tots els tests treballin amb IDs vàlids i coherents.

creaRespostes_i_getRespostes_buides_ok:
- Objectiu: Comprovar que es pot crear el conjunt de respostes d'un usuari per una enquesta i que inicialment és buit.
- Resultat esperat: Després de creaRespostes(idE, idContestant), getRespostes(idE, idContestant) retorna un objecte no nul i la llista interna de respostes té mida 0.

creaGetRespostes_afegirILlegir_ok:
- Objectiu: Comprovar el cas general d'ús: crear respostes, crear preguntes, associar-les a l'enquesta, afegir respostes i recuperar-les.
- Resultat esperat: Després de crear una pregunta Integer i una Text, afegir-les a l'enquesta i respondre-les, getRespostes(idE, idContestant) retorna un conjunt amb 2 respostes.

getResposta_ok:
- Objectiu: Comprovar la recuperació d'una resposta concreta a partir de (enquesta, usuari, pregunta).
- Resultat esperat: Després d'afegir una resposta de text "hola" a la pregunta creada, GetResposta(idE, idContestant, idPregunta) retorna una Resposta amb el mateix idPregunta i amb getContingut() igual a ["hola"].

creaRespostes_duplicat_llenca:
- Objectiu: Comprovar que no es permet crear dues vegades el conjunt de respostes per la mateixa parella (enquesta, usuari).
- Resultat esperat: Després d'una primera crida correcta a creaRespostes(idE, idContestant), una segona crida amb els mateixos paràmetres llença IllegalStateException.

getRespostes_inexistent_retornaBuit:
- Objectiu: Comprovar el comportament quan es demanen respostes per una parella (enquesta, usuari) existent però sense respostes creades prèviament.
- Resultat esperat: getRespostes(idE, idContestant) retorna un objecte no nul amb una llista de respostes buida (mida 0), sense llençar excepció.

afegeixResposta_duplicadaMateixIdP_llenca:
- Objectiu: Comprovar que no es permet afegir dues respostes diferents a la mateixa pregunta dins la mateixa parella (enquesta, usuari).
- Resultat esperat: Després d'afegir una resposta "A" a una pregunta Text, una segona crida a AfegeixResposta amb el mateix idPregunta llença IllegalStateException.

eliminarRespostes_i_despresGet_retornaBuit:
- Objectiu: Comprovar que l'operació d'eliminar totes les respostes d'un usuari en una enquesta funciona i deixa el conjunt buit.
- Resultat esperat: Després d'afegir una resposta i cridar EliminarRespostes(idE, idContestant), getRespostes(idE, idContestant) retorna un objecte no nul amb 0 respostes.

eliminarResposta_ok_i_cleanupSubmap_retornaBuit:

- Objectiu: Comprovar l'eliminació individual de respostes i el comportament quan s'elimina l'última resposta restant.
- Resultat esperat: Després d'afegir dues respostes, eliminar-ne una redueix la mida a 1, i eliminar la segona fa que getRespostes(idE, idContestant) retorni un conjunt no nul amb la llista buida (mida 0).

Tests Enquesta

setUp
Objectiu: preparar un CtrlDomini nou, crear un usuari i una enquesta base per reutilitzar a tots els tests.
Resultat esperat: enquesta inicialitzada amb id 1, títol de prova i idCreador coherent amb l'usuari creat.

testConstructoraValida
Objectiu: comprovar que la constructora d'Enquesta inicialitza correctament id, títol i idCreador.
Resultat esperat: getIdE() retorna 1, getTitol() el títol passat, i getIdCreador() coincideix amb l'usuari de proves.

testConstructoraTitolNul / testConstructoraTitolBuit
Objectiu: validar que no es permet crear enquestes amb títol nul o buit.
Resultat esperat: la constructora llença IllegalArgumentException en cada cas.

testConstructoraIdCreadorNul / testConstructoraCtrlDominiNul
Objectiu: comprovar la validació de paràmetres obligatoris (idCreador i ctrlDomini) a la constructora.
Resultat esperat: es llença IllegalArgumentException quan idCreador és nul o el controlador és nul.

testModificaTitol / testModificaTitolInvalid
Objectiu: verificar la modificació del títol i les validacions associades.
Resultat esperat: amb un nou títol vàlid, getTitol() s'actualitza; amb un títol en blanc, modificarTitol llença IllegalArgumentException.

testCreaPreguntaNum
Objectiu: provar la creació d'una pregunta de tipus integer des d'Enquesta amb paràmetres mínim i màxim.
Resultat esperat: es retorna un id de pregunta no nul i aquest id s'afegeix a getIdsPreguntes() de l'enquesta.

testCreaPreguntaMulti
Objectiu: provar la creació d'una pregunta nominal_multiple amb conjunt d'opcions i qMax coherent.
Resultat esperat: l'id retornat no és nul i forma part de la llista d'ids de preguntes.

testCreaPreguntaText
Objectiu: comprovar la creació d'una pregunta de tipus text sense paràmetres addicionals.
Resultat esperat: es crea una nova pregunta, l'id no és nul i queda afegit a l'enquesta.

testCreaPreguntaOrdinal
Objectiu: provar la creació d'una pregunta ordinal amb un Vector d'opcions ordenades.
Resultat esperat: es retorna un id vàlid i l'enquesta conté aquesta pregunta en la seva llista interna.

testCreaPreguntaNominalUnica
Objectiu: comprovar la creació de preguntes nominal_unica amb un conjunt d'opcions sense duplicats.
Resultat esperat: l'id creat s'afegeix correctament a getIdsPreguntes().

testCreaPreguntaTipusInvalid
Objectiu: verificar que no es permet crear preguntes amb un tipus desconegut.
Resultat esperat: creaPregunta("num_invalid", ...) llença IllegalArgumentException.

testAfegeixAUnaPosicio
Objectiu: comprovar la inserció d'una nova pregunta en una posició concreta dins de l'enquesta.
Resultat esperat: després de crear Q1 i Q2 al final i Q3 a la posició 1, l'ordre d'ids és [Q1, Q3, Q2].

testInsercioPosicioForaRang
Objectiu: validar que la inserció a una posició fora del rang actual de la llista llença error.
Resultat esperat: intentar crear una pregunta a un índex que supera la mida actual de la llista provoca IndexOutOfBoundsException.

(Recordatori: si al final del fitxer hi ha més tests d'eliminar o modificar pregunta, es poden descriure igual que amb Usuari, Resposta i Respostes, amb Objectiu i Resultat esperat per cadascun.)

Tests K-Medoids

setUp
Objectiu: preparar un CtrlDomini i definir una pregunta de cada tipus (Text, Integer, Ordinal, Nominal única i Nominal múltiple) per construir datasets de proves complets.
Resultat esperat: totes les preguntes es creen correctament al domini i queden disponibles per associar-hi Respostes.

creaRespostesUsuariComplet (mètode auxiliar)
Objectiu: encapsular la creació d'un objecte Respostes amb una resposta per a cada tipus de pregunta, a partir de valors de prova parametritzats.

Resultat esperat: es retorna una instància de Respostes amb cinc Resposta (integer, ordinal, nominal_unica, nominal_multiple i text) coherents amb les preguntes creades al setUp.

testExecutaClusteringAmbTotsElsTipus
Objectiu: provar el cas bàsic d'execució de K-Medoids sobre un dataset petit amb totes les tipologies de preguntes, validant medoids, clusters i cost.
Resultat esperat:

- Es construeix una llista de quatre Respostes amb dos usuaris similars (primer clúster) i dos de molt diferents (segon clúster).
- executa(20, true) retorna un objecte de tipus KMedoids.Resultat amb k = 2, medoids, dos clústers no buits, els usuaris 0 i 1 assignats al medoid 0, els usuaris 2 i 3 al medoid 3, i un cost total de 0.24 dins la tolerància de 1e-6.

testKForaDeRang
Objectiu: comprovar la validació de k respecte al nombre d'elements del dataset.
Resultat esperat: amb només dues Respostes i k = 3, la crida a executa(10, true) llença IllegalArgumentException perquè k és més gran que n.

testEstructuraRespostesInconsistent
Objectiu: assegurar que K-Medoids detecta datasets inconsistents, on diferents Respostes tenen estructura o nombre de respostes diferents.
Resultat esperat: construir una llista amb una Respostes completa i una altra amb només una Resposta fa que el constructor de KMedoids llenci IllegalArgumentException mitjançant la validació de Algorisme.setRespostes().

Tests Pregunta (classe base)

testConstructorValid
Objectiu: comprovar que la constructora de Pregunta assigna correctament l'enunciat i el camp d'obligatorietat en una subclasse simple de prova.
Resultat esperat: en crear una PreguntaFake amb enunciat "Test" i obligatòria, getEnunciat() retorna "Test" i esObligatoria() retorna true.

testModificarEnunciatValid
Objectiu: verificar que el mètode modificarEnunciat de la classe base actualitza l'enunciat quan se li passa un valor vàlid.
Resultat esperat: després de cridar modificarEnunciatPublic("Nou") sobre una pregunta amb enunciat inicial "Original", getEnunciat() retorna "Nou".

Tests PreguntaText

constructorValid
Objectiu: comprovar que la constructora de PreguntaText accepta un enunciat no buit i marca correctament l'obligatorietat.

Resultat esperat: en crear una PreguntaText amb enunciat "Nom" i obligatòria, getEnunciat() retorna "Nom" i esObligatoria() retorna true.

constructorEnunciatNullLlença
Objectiu: validar que no es permet inicialitzar una PreguntaText amb enunciat nul.
Resultat esperat: en passar null com a enunciat, la implementació interna tracta el cas com a definició invàlida (segons la lògica de validarDefinicio de PreguntaText), i el test comprova que l'objecte gestionat no queda en un estat inconsistent (no és null).

constructorEnunciatBuitLlença
Objectiu: validar que no es permet inicialitzar una PreguntaText amb enunciat buit.
Resultat esperat: en passar una cadena buida com a enunciat, la definició es considera invàlida i el test comprova que la instància creada no deixa l'objecte en un estat inconsistent.

Tests PreguntaInteger

constructorValidSenseRangs
Objectiu: comprovar que es pot crear una PreguntaInteger sense especificar min i max (rangs nuls).
Resultat esperat: en crear la pregunta, getEnunciat() retorna "Edat", esObligatoria() és true i tant getMinValue() com getMaxValue() són null.

constructorValidAmbRangs
Objectiu: comprovar que la constructora de PreguntaInteger accepta un rang coherent min ≤ max.
Resultat esperat: amb min = 0 i max = 10, l'enunciat i l'obligatorietat són els esperats, i getMinValue() retorna 0 mentre que getMaxValue() retorna 10.

constructorRangsInvertitsLlenca
Objectiu: verificar que un rang invertit (min > max) es considera invàlid i no deixa l'objecte en un estat incoherent.
Resultat esperat: amb min = 10 i max = 0, la lògica de validació detecta la incoherència de rang i el test comprova que la instància gestionada no queda en un estat inconsistent.

modificarMinValid
Objectiu: comprovar que es pot actualitzar correctament el valor mínim mantenint la invarianta min ≤ max.
Resultat esperat: partint d'un rang , després de setMin(2) el min passa a ser 2 i el max continua sent 10.

modificarMinTrencaInvariantLlenca
Objectiu: assegurar que si es proposa un nou min que trenca la invarianta (min > max), no s'aplica el canvi.
Resultat esperat: partint de , després de setMin(20) els valors es mantenen en min = 0 i max = 10.

modificarMaxValid
Objectiu: comprovar que es pot augmentar el valor màxim respectant min ≤ max.
Resultat esperat: partint de , després de setMax(12) el rang passa a .

modificarMaxTrencaInvariantLlenca
Objectiu: assegurar que reduir max per sota del min no és acceptable i ha de fer rollback.
Resultat esperat: partint de, després de setMax(3) els valors segueixen sent min = 5 i max = 10.

rangsNulsPermetenAjustValidDespres
Objectiu: comprovar que, si inicialment min i max són null, es poden definir posteriorment de forma coherent.
Resultat esperat: amb min i max inicialment null, després de setMax(10) i setMin(0) el rang final és .

nulDespresTrencarInvariantLlenca
Objectiu: verificar que, si s'intenta establir un min i un max incoherents partint de rangs nuls, la classe manté la invarianta de forma segura.
Resultat esperat: després de setMax(5) i setMin(6) amb valors inicialment nuls, es compleix que o bé algun dels extrems és null o bé min ≤ max.

Tests PreguntaOrdinal

constructorValid
Objectiu: comprovar que la constructora de PreguntaOrdinal accepta una llista d'opcions vàlida i la registra correctament.
Resultat esperat: amb opcions ["Baix","Mig","Alt"], getEnunciat() retorna "Nivell", esObligatoria() és true i getNumOpcions() retorna 3.

constructorOpcionsNullesLlenca
Objectiu: validar que no es permet inicialitzar una PreguntaOrdinal amb el vector d'opcions nul.
Resultat esperat: en passar null com a opcions, la definició es considera invàlida i el test comprova que la instància gestionada no queda en un estat inconsistent.

constructorOpcionsBuidesLlenca
Objectiu: validar que no es permet inicialitzar una PreguntaOrdinal amb cap opció.
Resultat esperat: amb un vector buit, la definició no és vàlida i el test comprova que l'objecte gestionat no queda en un estat inconsistent.

modificarOpcionsValid
Objectiu: comprovar que es poden afegir noves opcions ordenades mantenint una definició correcta.
Resultat esperat: partint de ["Baix","Mig"], després de setOpcions(["Molt baix","Baix","Mig","Alt"]) getNumOpcions() passa a ser 4.

modificarOpcionsNullesLlenca
Objectiu: assegurar que, si es passa null a modificarOpcions, es fa rollback i no canvia el nombre d'opcions.
Resultat esperat: el valor de getNumOpcions() es manté igual que abans de la crida.

modificarOpcionsBuidesLlenca
Objectiu: garantir que no es pot deixar una PreguntaOrdinal sense opcions mitjançant modificarOpcions.
Resultat esperat: després d'intentar substituir per un vector buit, getNumOpcions() conserva el valor anterior.

Tests PreguntaNominalUnica

constructorValid
Objectiu: comprovar que la constructora de PreguntaNominalUnica accepta un conjunt d'opcions vàlid i el desa correctament.
Resultat esperat: amb opcions {"A","B","C"}, getEnunciat() retorna "Color", esObligatoria() és true i getOpcions() conté exactament el conjunt passat.

constructorOpcionsNullesLlenca
Objectiu: validar que no es permeten opcions nul·les a la definició inicial (conjunt null).
Resultat esperat: passar null com a conjunt d'opcions es considera invàlid i el test comprova que la instància creada no queda en un estat inconsistent.

constructorOpcionsBuidesLlenca
Objectiu: validar que no es pot crear una PreguntaNominalUnica sense cap opció disponible.
Resultat esperat: amb un conjunt buit, la definició es considera invàlida i el test comprova que l'objecte gestionat no queda inconsistent.

constructorOpcioNulaLlenca
Objectiu: assegurar que un conjunt amb algun element null no és acceptat com a definició d'opcions.
Resultat esperat: amb un conjunt com {"A", null, "C"}, la validació considera la definició invàlida.

constructorOpcioBuidaLlenca
Objectiu: comprovar que tampoc es permeten opcions amb cadena buida dins del conjunt.
Resultat esperat: un conjunt que inclou "" es considera invàlid i no deixa l'objecte en un estat incorrecte.

modificarOpcionsValid
Objectiu: comprovar que es poden substituir les opcions per un nou conjunt vàlid.
Resultat esperat: partint d'{"A","B"}, en fer setOpcions({"X","Y","Z"}) el nou conjunt d'opcions passa a ser exactament {"X","Y","Z"}.

modificarOpcionsNullesLlenca
Objectiu: assegurar que passar null a modificarOpcions no modifica l'estat de l'objecte.
Resultat esperat: després de la crida, getOpcions() continua contenint el conjunt antic.

modificarOpcionsBuidesLlenca
Objectiu: garantir que no es pot deixar la pregunta sense opcions mitjançant una modificació.
Resultat esperat: en passar un conjunt buit, es fa rollback i getOpcions() continua amb les opcions anteriors.

modificarOpcionsAmbElementNulLlenca
Objectiu: validar que un conjunt amb algun element null no s'accepta en una modificació.
Resultat esperat: després d'intentar establir opcions amb un element null, getOpcions() es manté igual que abans.

modificarOpcionsAmbElementBuitLlenca
Objectiu: comprovar que les opcions amb cadena buida no són acceptades en modificacions.
Resultat esperat: després d'intentar establir un conjunt que conté "", les opcions es mantenen sense canvis.

Tests PreguntaNominalMult

constructorValid
Objectiu: comprovar que la constructora de PreguntaNominalMult accepta un conjunt d'opcions vàlid i un qMax coherent.
Resultat esperat: amb opcions {"A","B","C"} i qMax = 2, getEnunciat() retorna "Colors", esObligatoria() és true, getOpcions() conté el conjunt passat i getQMax() retorna 2.

constructorOpcionsNullesLlenca
Objectiu: validar que el conjunt d'opcions no pot ser null a la definició inicial.
Resultat esperat: passar null com a opcions es considera invàlid i el test comprova que la instància gestionada no queda en estat inconsistent.

constructorOpcionsBuidesLlenca
Objectiu: assegurar que no es pot crear una PreguntaNominalMult sense opcions.
Resultat esperat: amb un conjunt buit, la definició no és vàlida i el test comprova que l'objecte no queda inconsistent.

constructorOpcioNulaLlenca
Objectiu: comprovar que les opcions no poden contenir elements nulls.
Resultat esperat: un conjunt com {"A", null, "C"} es considera invàlid i no deixa l'objecte en estat incorrecte.

constructorOpcioBuidaLlenca
Objectiu: comprovar que les opcions tampoc poden ser cadenes buides.
Resultat esperat: un conjunt que inclou "" es considera invàlid i la instància gestionada es manté coherent.

constructorQMaxNulLlenca
Objectiu: validar que qMax no pot ser null en la definició inicial.
Resultat esperat: el valor null per qMax es considera invàlid i el test comprova que l'objecte no queda en estat inconsistent.

constructorQMaxNoPositiuLlenca
Objectiu: assegurar que qMax ha de ser estrictament positiu.
Resultat esperat: amb qMax = 0 la definició no és vàlida i la instància gestionada es manté coherent.

constructorQMaxMesGranQueOpcionsLlenca
Objectiu: comprovar que qMax no pot superar el nombre d'opcions disponibles.
Resultat esperat: amb dos elements al conjunt i qMax = 3 la definició es considera invàlida.

modificarOpcionsValid
Objectiu: comprovar que es poden substituir les opcions per un nou conjunt vàlid sense afectar qMax si continua sent coherent.
Resultat esperat: partint d'{"A","B"} i qMax = 2, en establir opcions {"X","Y","Z"} el nou conjunt s'actualitza i qMax continua sent 2.

modificarOpcionsAmbNulsLlenca
Objectiu: assegurar que no s'accepten nous conjunts d'opcions amb elements nulls.
Resultat esperat: després d'intentar establir {"A", null}, les opcions i qMax es mantenen com abans (rollback).

modificarOpcionsBuidesLlenca
Objectiu: garantir que no es pot buidar el conjunt d'opcions a través de modificarOpcions.
Resultat esperat: si es passa un conjunt buit, getOpcions() i getQMax() queden igual.

modificarOpcionsReduintPerSotaDeQMaxLlenca
Objectiu: comprovar que no es permet reduir el nombre d'opcions per sota de qMax existent.
Resultat esperat: partint d'{"A","B","C"} i qMax = 3, si es proposa el conjunt {"A","B"}, es fa rollback i tanto les opcions com qMax es mantenen.

modificarQMaxValid
Objectiu: comprovar que es pot incrementar qMax sempre que no superi el nombre d'opcions.
Resultat esperat: amb opcions {"A","B","C"} i qMax inicial 2, després de setQMax(3) el nou qMax és 3 i el conjunt d'opcions no canvia.

modificarQMaxNoPositiuLlenca
Objectiu: assegurar que no es pot establir un qMax no positiu.
Resultat esperat: amb opcions {"A","B"} i qMax = 2, després de setQMax(0) es fa rollback i qMax continua sent 2.

modificarQMaxMesGranQueOpcionsLlenca
Objectiu: comprovar que no es permet establir un qMax superior al nombre d'opcions actual. Resultat esperat: amb opcions {"A","B"} i qMax = 2, si es crida setQMax(3) qMax es manté a 2 i les opcions no canvien.

6. Capa de Presentació i Vista

6.1 Descripció global del sistema

Propòsit:
La capa de presentació encapsula tota la lògica d'interacció amb l'usuari i fa de pont entre les vistes Swing i el CtrlDomini. El controlador de presentació (CtrlPresentacio) rep tots els esdeveniments de les vistes, valida les dades d'entrada d'usuari (formats bàsics, camps obligatoris) i delega al domini les operacions de gestió d'usuaris, enquestes, respostes, importació CSV i anàlisi.

Estructura general:
L'aplicació arrenca amb MainGUI, que crea un CtrlDomini i un CtrlPresentacio i invoca el mètode inicia. Aquest mètode mostra la VistaPrincipal, que permet triar entre el rol de creador o de respondent. Per cada cas d'ús hi ha una vista específica (VistaMenuCreador, VistaGestioUsuari, VistaGestioEnquestes, VistaGestioPreguntes, VistaGestioRespostes, VistaAnalisi, VistaIntroduirCodiEnquesta), totes basades en JFrame o JDialog amb diferents panells, taules i formularis. Cada vista manté un enllaç al CtrlPresentacio mitjançant setListener(this), de manera que els listeners de botons i components criden mètodes del controlador de presentació en lloc d'accedir directament al domini.

Gestió d'usuaris i enquestes:
VistaGestioUsuari mostra tots els usuaris en una taula i permet crear nous usuaris, modificar-los o eliminar-los mitjançant camps de text i botons, mentre CtrlPresentacio s'encarrega de recarregar la taula i de mostrar missatges d'error o informació amb JOptionPane (per exemple quan no hi ha usuaris o hi ha problemes de validació).
VistaGestioEnquestes mostra la llista d'enquestes disponibles, permet crear-ne de noves a partir d'un títol i un id de creador, eliminar-les i també importar-ne des d'un fitxer CSV, demanant la ruta, el títol i l'id d'usuari; un cop el domini crea o importa l'enquesta, la taula es refresca automàticament.

Gestió de preguntes i respostes:
VistaGestioPreguntes llista totes les preguntes del sistema amb el seu id, tipus i enunciat, i ofereix formularis per crear preguntes de cada tipus, reutilitzant les validacions de rangs i opcions definides al domini mitjançant crides a CtrlDomini.

VistaGestioRespostes es crea dinàmicament quan un respondent introdueix el codi d'enquesta i el seu id a VistaIntroduirCodiEnquesta, la vista construeix els camps adequats per contestar cada pregunta (text, numèrica, opcions úniques o múltiples) i, en prémer enviar, interacciona

amb CtrlPresentacio perquè aquest creï l'objecte Respostes i afegeixi cada Resposta al domini, gestionant també errors de format o enquestes sense preguntes.

Anàlisi i navegació:
VistaAnalisi mostra la llista d'enquestes disponibles per anàlisi i permet a l'usuari seleccionar l'enquesta, triar l'algorisme (K-Means o K-Medoids), indicar el valor de k i el nombre màxim d'iteracions. En executar, CtrlPresentacio crida els mètodes d'anàlisi del domini i retorna a la vista un resum amb cost/inèrcia, silhouette, temps d'execució i informació bàsica dels clústers, que es mostra en components de text o taula. El controlador també centralitza tota la navegació: amaga i mostra les diferents vistes en funció de l'opció escollida (per exemple, tornar al menú creador des de qualsevol subpantalla, tornar a la pantalla principal, o passar de la introducció de codi d'enquesta a la pantalla de resposta), mantenint el flux d'ús coherent amb els casos d'ús definits a la documentació.

6.2 Breu descripció de cada classe de la capa de presentació

MainGUI
Classe d'entrada de l'aplicació. Conté el main, crea CtrlDomini i CtrlPresentacio i inicia el flux gràfic cridant inicia().

CtrlPresentacio
Controlador de presentació que centralitza la navegació entre pantalles i totes les peticions cap a CtrlDomini. Valida entrades bàsiques (camps buits, formats simples) i gestiona missatges a l'usuari (errors/informació). Exposa mètodes per: gestió d'usuaris, gestió d'enquestes, gestió de preguntes, registre de respostes, importació CSV i execució d'anàlisi (K-means/K-medoids).

VistaPrincipal
Pantalla inicial de selecció de rol. Permet escollir entre accions de creador/administrador o respondent, i delega la navegació a CtrlPresentacio mitjançant setListener(...) i esdeveniments de botons.

VistaMenuCreador
Menú principal del creador. Dona accés a les vistes de gestió (usuaris, enquestes, preguntes i anàlisi). Manté el listener a CtrlPresentacio per executar la transició a la vista corresponent.

VistaGestioUsuari
Pantalla de manteniment d'usuaris. Mostra usuaris en taula i permet crear, modificar i eliminar usuaris amb formularis i botons. Totes les accions criden CtrlPresentacio i es mostren errors/informació amb diàlegs.

VistaGestioEnquestes
Pantalla de manteniment d'enquestes. Mostra enquestes en taula, permet crear-ne, eliminar-ne i també importar una enquesta des de CSV indicant ruta, títol i id de creador. Les operacions deleguen en CtrlPresentacio.

VistaGestioPreguntes
Pantalla de manteniment de preguntes. Permet crear preguntes de diferents tipus (text, integer, ordinal, nominal única, nominal múltiple), valida camps mínims i refresca la taula de preguntes amb dades del domini via CtrlPresentacio.

VistaIntroduirCodiEnquesta
Pantalla per al respondent. Demana l'id d'enquesta i id d'usuari per accedir a la pantalla de resposta. Envia la petició al CtrlPresentacio, que valida l'existència i obre la vista de respostes.

VistaGestioRespostes
Pantalla dinàmica de resposta d'una enquesta. Construeix els camps segons el tipus de cada pregunta (text, numèrica, opcions úniques/múltiples) i, en enviar, delega a CtrlPresentacio la creació/registre d'unes Respostes i les seves Resposta.

VistaAnalisi
Pantalla d'anàlisi. Permet seleccionar enquesta, triar algorisme (K-means/K-medoids), indicar k i maxIter, i mostra el resum retornat pel domini (cost/inèrcia, silhouette, temps, info bàsica de clústers) a través de CtrlPresentacio.

7. Capa de Persistencia

Propòsit:
La capa de persistència encapsula l'accés a fitxers i la importació de dades externes (CSV). La presentació i el domini no treballen directament amb fitxers: ho fan a través de CtrlPersistencia.

Hem dissenyat la capa de persistencia mitjançant 2 tipus de fichers csv, el primer tipus es un fitxer que conté tota l'informació de tots els usuaris, aquest tipus de fitxer serà únic, és a dir, només hi haurà un fitxer per tot el nostre sistema i aquest es carregarà al principi i al final de l'execució del programa carregant i guardant així l'informació dels diferents usuaris. Després tindrem un altre tipus de fitxer que guardarà els enunciats i respostes de les enquestes, (els enunciats són la primera línea del csv que correspon al nom de columna si ho mirem com una taula i la resta de líneas del CSV son les respostes a les preguntes de l'enquesta de cada usuari) d'aquests fitxers tindrem tants com enquestes al sistema i els anirem carregant segons demanda, es a dir, si al executar el programa ens interessa consultar només una enquesta, importem només les dades de l'enquesta desitjada (estalviant espai i temps al no carregar totes les enquestes del sistema). Finalment al tancar el programa els canvis es guardaran al mateix fitxer automàticament.

7.1 CtrlPersistencia

Propòsit:
Controlador de persistència. Centralitza les operacions de lectura/escriptura i utilitza gestors especialitzats.

Atributs rellevants:

- GestorFitxersUsuari gestorUsuaris: gestiona lectura/escriptura del CSV d'usuaris.
- GestorFitxersEnquesta gestorEnquestes: gestiona l'escriptura d'enquestes a fitxer.
- CsvImporter importer: component per parsejar CSV d'enquestes.

Mètodes principals:

- carregarUsuaris(): retorna el contingut del fitxer d'usuaris en format ArrayList<ArrayList<String>> (files i camps).
- guardarUsuaris(ArrayList<ArrayList<String>> usuaris): escriu/sobreescriu el CSV d'usuaris amb les dades rebudes.

- guardarEnquesta(String titol, ArrayList<ArrayList<String>> dadesCSV): desa una enquesta (en format CSV estructurat) mitjançant GestorFitxersEnquesta.

- importarEnquestaDesdeCSV(String filePath): llegeix un CSV extern i retorna les dades parsejades en una estructura ArrayList<ArrayList<String>>.

7.2 GestorFitxersUsuari

Propòsit:
Classe responsable de la persistència d'usuaris en un fitxer CSV.

Mètodes principals:

- carregarUsuaris(): Aquest mètode llegeix el fitxer FitxerUsuaris.csv i retorna tots els usuaris en una estructura ArrayList<ArrayList<String>>, on cada línia del CSV es converteix en una llista de camps (per exemple: id, nom i email). Primer comprova si el fitxer existeix; si no existeix, retorna directament una llista buida. Si existeix, el recorre línia a línia amb un BufferedReader, ignora les línies buides, separa cada línia per comes (split(",")) i afegeix els valors a una llista usuari, que després s'afegeix a la llista global usuaris. Si hi ha algun problema d'entrada/sortida, mostra un missatge d'error i retorna el que s'hagi pogut carregar.
-
- guardarUsuaris(): Aquest mètode escriu al fitxer FitxerUsuaris.csv la llista d'usuaris rebuda per paràmetre, sobreescrivint el contingut anterior del fitxer. Per a cada usuari (que és una llista de String amb els camps), crea una línia CSV ajuntant els camps amb comes (String.join(",", usuari)) i la desa com una línia nova al fitxer amb println. En cas que es produeixi un error d'escriptura (IOException), el mètode no llença l'excepció cap amunt, sinó que mostra un missatge indicant que hi ha hagut un error escrivint el fitxer.

7.3 GestorFitxersEnquesta

Propòsit:
Classe responsable de desar una enquesta en format fitxer (CSV) a partir d'una estructura ja preparada.

Mètodes principals:

- guardarEnquesta(String titol, ArrayList<ArrayList<String>> dadesCSV): crea/escriu el fitxer d'enquesta i bolca les files CSV rebudes.

7.4 CsvImporter

Propòsit:
Classe encarregada de llegir i parsejar un CSV extern d'enquesta i transformar-lo a una estructura manipulable pel domini.

Mètodes principals:

- parseEnquestaFromCSV(String filePath): llegeix el fitxer CSV, processa les línies i retorna les dades en format ArrayList<ArrayList<String>>, preparades per ser transformades en objectes del domini.