Disseny d’un teclat

Projectes de Programació

Quadrimestre Q1 2023/2024

Identificador de l’equip: 41.6

Versió del lliurament: 1.0

Jordi Homedes (jordi.homedes)

Lluís Mir (lluis.mir.agusti)

Marc Nafria (marc.nafria)

Pol Plana (pol.plana)

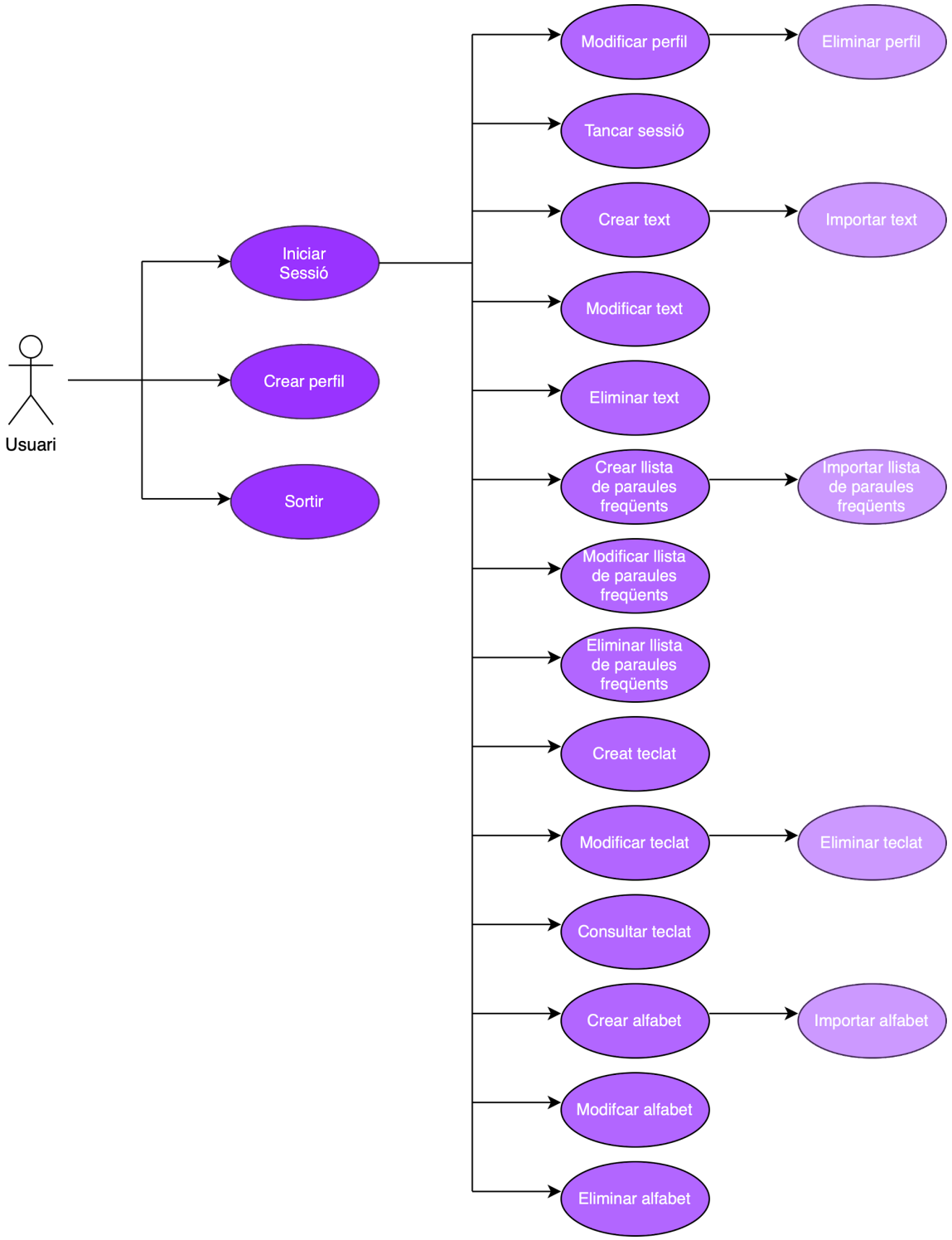
FIB

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Índex

1. CASOS D'ÚS

1.1. DIAGRAMA DE CASOS DÚS



1.2. DESCRIPCIÓ DE CASOS D'ÚS

Cas d'ús #1 - Iniciar sessió

Actor: Usuari registrat
Precondició: L'usuari té un perfil i no està *loguejat*
Detonant: L'usuari vol entrar al seu perfil
Escenari Principal:

1) L'usuari tria el seu perfil
2) L'usuari propociona la seva contrasenya
3) El sistema valida les dades i proporciona accés

Errors Possibles i Cursos Alternatius:

2a) L'usuari propociona unes credencials incorrectes: el sistema avisa de l'error i torna al pas 1

Cas d'ús #2 - Tancar sessió

Actor: Usuari registrat
Precondició: L'usuari està *loguejat* al seu perfil
Detonant: L'usuari vol sortir del seu perfil i deixar d'estar loguejat
Escenari Principal:

1) L'usuari indica que vol tancar la sessió del seu perfil
2) L'usuari confirma la voluntat de sortir del seu perfil
3) El sistema surt del perfil de l'usuari i el porta al cas d'ús #1 - Seleccionar perfil

Errors Possibles i Cursos Alternatius:

2a) L'usuari cancel·la l'acció: el sistema cancel·la l'operació

Cas d'ús #3 - Modificar perfil

Actor: Usuari registrat
Precondició: L'usuari té un perfil i està *loguejat* al seu perfil
Detonant: L'usuari vol modificar les dades del seu perfil (nom i contrasenya)
Escenari Principal:

1) L'usuari accedeix a la secció de modificació del perfil
2) L'usuari realitza els canvis desitjats en els atributs del perfil
3) El sistema valida i guarda les modificacions

Errors Possibles i Cursos Alternatius:

2a) L'usuari proporciona un nom d'usuari ja existent: el sistema avisa de l'error i torna al pas 1

2b) L'usuari propociona una contrasenya que no compleix els requisits de seguretat: el sistema avisa de l'error i torna al pas 1

Cas d'ús #4 - Eliminar perfil

Actor: Usuari registrat
Precondició: L'usuari té un perfil i està loguejat al seu perfil
Detonant: L'usuari vol eliminar el seu perfil
Escenari Principal:

1) L'usuari accedeix a la secció de modificació del perfil i invoca la eliminació del perfil
2) L'usuari confirma la voluntat de suprimir el perfil
3) El sistema elimina el perfil de l'usuari i tota la informació associada, i es torna a les opcions d'inici

Errors Possibles i Cursos Alternatius:

2a) L'usuari cancel·la l'acció d'eliminació: el sistema cancel·la l'operació i reprèn l'apartat de modificació del perfil

Cas d'ús #5 - Crear text

Actor: Usuari registrat
Precondició: L'usuari té almenys un perfil i està loguejat a un d'ells. L'usuari té com a mínim un alfabet creat.
Detonant: L'usuari vol afegir un text a la seva llista de textos
Escenari Principal:

1) L'usuari accedeix a la secció de textos i en crea un.
2) L'usuari pot seleccionar entre introduir manualment el text o importar-lo des de l'ordinador.
3) L'usuari introdueix el text i un nom que identifiqui el text.
4) L'usuari selecciona, de la seva col·lecció d'alfabets, en quin alfabet està el text corresponent.
5) L'usuari confirma la creació del text.
6) Abans de guardar, el sistema comprova que el text estigui en l'alfabet seleccionat.
7) El sistema guarda el text.
8) Es salta al Cas d'ús #13 - Crear teclat

Errors Possibles i Cursos Alternatius:

3a) L'usuari importa un text en format (.txt) de l'ordinador (cas d'ús #6).
4a) L'usuari introdueix un títol per identificar el text.
5a) Pas 4) del curs normal.

+ [Abort]: En qualsevol moment es pot abortar la creació si es cancel·la el procés. Llavors el sistema no guarda cap informació ni crea cap text.
+ [Alfabet no coincideix]: El text no està en l'alfabet corresponent. Es torna al pas 4).

Cas d'ús #6 - Importar text de l’ordinador

Actor: Usuari registrat
Precondició: L’usuari té almenys un perfil i està loguejat a un d’ells
Detonant: En la creació o modificació d’un text, l’usuari selecciona importar des de l’ordinador
Escenari Principal:

1) S’obre una pestanya de l'explorador d'arxius de l’ordinador, on es permet seleccionar un fitxer (.txt).
2) L’usuari proporciona un nom per al text.
3) L’usuari selecciona el fitxer corresponent.
4) El fitxer es llegeix per a obtenir el text que conté, que s’utilitzarà per a l’entrada de tipus text.
5) L’usuari selecciona, de la seva col·lecció d’alfabets, en quin alfabet està el text corresponent.
6) L’usuari confirma la creació del text.
7) Abans de guardar, el sistema comprova que el text estigui en l’alfabet seleccionat.
8) El sistema guarda el text.
9) Es salta al Cas d’ús #13 - Crear teclat

Errors Possibles i Cursos Alternatius:

+ [ErrorFormat]: El format del fitxer no és el correcte.
+ [ErrorLectura]: El fitxer té algun problema i no es pot llegir.

Qualsevol d'aquests errors anul·la l’execució del cas d’ús.

Cas d'ús #7 - Modificar text

Actor: Usuari dins el perfil
Precondició: El text existeix
Detonant: L’usuari vol canviar alguna cosa del text (alfabet, text, títol)
Escenari Principal:

1) L’usuari accedeix a l’apartat de modificar un text en concret.
2) El sistema carrega el títol, l’idioma i l’alfabet, de tal forma que l’usuari pot canviar algunes d’aquestes coses. També pot accedir al cas d’ús d'importar text en comptes d’escriure'l.
3) L’usuari confirma els canvis.
4) El sistema guarda els canvis.
5) Es salta al Cas d’ús #13 - Crear teclat

Errors Possibles i Cursos Alternatius:

+ [AlfabetNoCoincideix]: El text no està en l'alfabet que l'usuari ha seleccionat. Es cancel·la el cas d'ús i no es guarda res.

Cas d'ús #8 - Eliminar text

Actor: Usuari dins el perfil
Precondició: El text existeix
Detonant: L'usuari vol eliminar un text de la llista de textos del seu perfil
Escenari Principal:

1) L'usuari accedeix a la secció de modificació del perfil i invoca la eliminació del text.
2) L'usuari confirma la voluntat de suprimir el text.
3) El sistema elimina el text i tota la informació associada.
4) Es salta al Cas d'ús #15 - Eliminar teclat

Errors Possibles i Cursos Alternatius:

2a) L'usuari cancel·la l'acció d'eliminació: el sistema cancel·la l'operació i reprendre l'apartat de modificació del perfil.

Cas d'ús #9 - Crear LPF

Actor: Usuari registrat
Precondició: L'usuari té almenys un perfil i està loguejat a un d'ells
Detonant: L'usuari vol afegir una LPF a la seva llista de LPF
Escenari Principal:

1) L'usuari accedeix a la secció de LPF i en crea una.
2) L'usuari pot seleccionar entre introduir manualment la LPF o importar-la des de l'ordinador.
3) L'usuari introdueix en el format adequat una llista de paraules i frequencies i un nom que identifiqui la LPF (títol).
4) L'usuari selecciona, de la seva col·lecció d'alfabets, en quin alfabet estan les paraules de la LPF.
5) L'usuari confirma la creació del text.
6) Abans de guardar, el sistema comprova que les paraules de la LPF estiguin en l'alfabet seleccionat.
7) El sistema guarda la LPF.
8) Es salta al Cas d'ús #13 - Crear teclat

Errors Possibles i Cursos Alternatius:

3a) L'usuari importa una LPF en format (.txt) de l'ordinador (cas d'ús #10).
4a) L'usuari introdueix un títol per identificar la LPF.
5a) Pas 4) del curs normal.

+ [Abort]: En qualsevol moment es pot abortar la creació si es cancel·la el procés. Llavors el sistema no guarda cap informació ni crea cap LPF.
+ [Alfabet no coincideix]: Les paraules de la LPF no estan en l'alfabet corresponent. Es torna al pas 4).

Cas d'ús #10 - Importar LPF de l'ordinador

Actor: Usuari registrat
Precondició: L'usuari té almenys un perfil i està loguejat a un d'ells
Detonant: En la creació o modificació d'un text, l'usuari selecciona importar des de l'ordinador
Escenari Principal:

1) S'obre una pestanya de l'explorador d'arxius de l'ordinador, on es permet seleccionar un fitxer (.txt).
2) L'usuari selecciona el fitxer corresponent.
3) El fitxer es llegeix per a obtenir la LPF que conté, que s'utilitzarà per a l'entrada de tipus LPF.
4) L'usuari selecciona, de la seva col·lecció d'alfabets, en quin alfabet està la LPF corresponent.
5) L'usuari confirma la creació de la LPF.
6) Abans de guardar, el sistema comprova que la LPF estigui en l'alfabet seleccionat.
7) El sistema guarda la LPF.
8) Es salta al Cas d'ús #13 - Crear teclat

Errors Possibles i Cursos Alternatius:

+ [ErrorFormat]: El format del fitxer no és el correcte.
+ [ErrorLectura]: El fitxer té algun problema i no es pot llegir.

Qualsevol d'aquests errors anul·la l'execució del cas d'ús.

Cas d'ús #11 - Modificar LPF

Actor: Usuari registrat
Precondició: El LPF existeix
Detonant: L'usuari pitja el botó de modificar un LPF concret
Escenari Principal:

1) L'usuari selecciona la LPF que vol modificar.
2) El sistema carrega la informació actual de la LPF seleccionada
3) L'usuari realitza la modificació desitjada
4) L'usuari guarda les modificacions

5) El sistema actualitza la LPF amb les noves modificacions i confirma a l'usuari que els canvis s'han aplicat
6) Es salta al Cas d'ús #13 - Crear teclat

Errors Possibles i Cursos Alternatius:

+ [LPFNo1Coincideix]: Les paraules de la LPF no estan en l'alfabet corresponent.
+ [ErrorFormat]: El format del fitxer no és el correcte.
+ [ErrorLectura]: El fitxer té algun problema i no es pot llegir.

Cas d'ús #12 - Eliminar LPF

Actor: Usuari registrat
Precondició: L'usuari està *loguejat* al seu perfil i té creades una o més llista de paraules freqüents.
Detonant: L'usuari pitja el botó d'eliminar una LPF.
Escenari Principal:

1) L'usuari accedeix a la secció de les seves llistes de paraules freqüents.
2) L'usuari pitja el botó d'eliminar-ne alguna.
3) El sistema elimina la LPF.
4) Es salta al Cas d'ús #15 - Eliminar teclat

Errors Possibles i Cursos Alternatius:
Cap

Cas d'ús #13 - Crear teclat

Actor: Usuari registrat
Precondició: L'usuari està loguejat al seu perfil i existeix com a mínim una llista de paraules freqüents o un text.
Detonant: L'usuari ha creat una nova entrada de tipus Text o LPF associades a un alfabet.
Escenari Principal:

1) S'especifica quin tipus de teclat es vol, de dues mans o de polzes.
2) S'especifica el número de files que ha de tenir el teclat
3) Es genera el teclat en base a la llista de paraules freqüents associada a una entrada, sigui directament de tipus LPF o un text (conversió a LPF interna).

Errors Possibles i Cursos Alternatius:
Cap

Cas d'ús #14 - Modificar teclat

Actor: Usuari registrat

Precondició: L'usuari està loguejat al seu perfil i existeix com a mínim un teclat.
Detonant: L'usuari pitja el botó de modificar un teclat concret.
Escenari Principal:

1) L'usuari accedeix a la secció de teclats.
2) L'usuari prem el botó de modificar un teclat concret.
3) L'usuari introdueix el nou número de files que vol que tingui el teclat.
4) L'usuari prem el botó de confirmar la modificació.
5) El sistema actualitza el teclat en base als nous textos o llistes de paraules freqüents.

Errors Possibles i Cursos Alternatius:
L'usuari ha seleccionat introduit més files que lletres, o bé files inferiors a 1. En aquest cas es llença una excepció i es demana a l'usuari un número dins els valors establerts.

Cas d'ús #15 - Eliminar teclat

Actor: Usuari registrat.
Precondició: L'usuari està loguejat al seu perfil i existeix com a mínim un teclat.
Detonant: L'usuari ha eliminat una entrada, la qual té un teclat associat.
Escenari Principal:

1) S'elimina el teclat del sistema.

Errors Possibles i Cursos Alternatius:
Cap

Cas d'ús #16 - Consultar teclat

Actor: Usuari registrat
Precondició: L'usuari està loguejat al seu perfil i existeix com a mínim un teclat
Detonant: L'usuari vol conèixer els teclats registrats
Escenari Principal:

1) L'usuari accedeix a la zona de teclats
2) L'usuari selecciona l'opció de consular teclat
3) El sistema carrega el llistat de teclats existents
4) L'usuari selecciona el teclat que vol consular
5) El sistema carrega la informació detallada del teclat seleccionat

Errors Possibles i Cursos Alternatius:
Cap

Cas d'ús #17 - Crear alfabet

Actor: Usuari registrat

Precondició: L'usuari té almenys un perfil i està loguejat al seu perfil
Detonant: L'usuari vol afegir un alfabet a la seva llista d'alfabets
Escenari Principal:

1) L'usuari accedeix a la secció de alfabets i en crea un.
2) L'usuari pot seleccionar l'opció de crear un nou alfabet
3) L'usuari introdueix la informació necessària (nom i lletres separades per comes d'un sol caràcter i no repetides) per al nou alfabet
4) Es guarda el nou alfabet i el sistema confirma a l'usuari que el alfabet ha estat guardat correctament

Errors Possibles i Cursos Alternatius:

+ [NoContingut]: L'usuari no proporciona cap alfabet.
+ [LletresInvàlides]: L'usuari proporciona un conjunt de lletres que no compleix els requisits.

Cas d'ús #18 - Importar alfabet de l'ordinador

Actor: Usuari registrat
Precondició: L'usuari té almenys un perfil i està loguejat a un d'ells
Detonant: En la creació d'un alfabet, l'usuari selecciona importar des de l'ordinador
Escenari Principal:

1) Sobre una pestanya de l'explorador d'arxius de l'ordinador, on es permet seleccionar un fitxer (.txt).
2) L'usuari selecciona el fitxer corresponent
3) L'usuari proporciona un nom per a l'alfabet
4) El fitxer es llegeix per a obtenir els caràcters que conté, que es composen de lletres separades per comes d'un sol caràcter i no repetides.
5) Es guarda el nou alfabet i el sistema confirma a l'usuari que el alfabet ha estat guardat correctament

Errors Possibles i Cursos Alternatius:

+ [ErrorFormat]: El format del fitxer no és el correcte
+ [ErrorLectura]: El fitxer té algun problema i no es pot llegir(tamany)
+ [LletresInvàlides]: L'usuari proporciona un conjunt de lletres que no compleix els requisits.

Cas d'ús #19 - Modificar alfabet (afegir lletra)

Actor: Usuari registrat
Precondició: L'usuari està loguejat al seu perfil i existeix com a mínim un alfabet
Detonant: L'usuari prem el botó de modificar un alfabet concret (afegir lletra)
Escenari Principal:

1) L'usuari accedeix a la secció de alfabets

2) L’usuari prem el botó de modificar un alfabet concret
3) El sdemana quina lletra nova es vol afegir a l’alfabet
4) L’usuari prem el botó de confirmar la modificació
5) El sistema actualiza l’alfabet en base a les noves modificacions
6) El sistema actualitza els teclats associats a totes les entrades que estan en l’idioma de l’alfabet afegint-hi la nova lletra
7) Es confirma que la modificació s’ha fet correctament

Errors Possibles i Cursos Alternatius:

+ [ErrorTamany]: El tamany de l’alfabet supera els el màxim establert

Cas d'ús #20 - Eliminar alfabet

Actor: Usuari registrat
Precondició: L’usuari està loguejat al seu perfil i existeix com a mínim un alfabet.
Detonant: Després de prémer el botó de modificar alfabet, l’usuari pitja l’opció.
Escenari Principal:

1) L’usuari accedeix a la modificació d’un alfabet existent
2) L’usuari prem el botó eliminar alfabet
3) El sistema elimina l’alfabet seleccionat i confirma l’usuari que la eliminació ha set exitosa
4) S’eliminen les entrades (Cas d’ús #8 i #12) i els seus teclats associats (implícit en eliminar les entrades)

Errors Possibles i Cursos Alternatius:
Cap

Cas d'ús #21 - Crear perfil

Actor: Usuari no registrat
Precondició: L'usuari no té cap perfil existent en el sistema
Detonant: L'usuari vol crear un nou perfil
Escenari Principal:

1) L'usuari accedeix a la pàgina de registre
2) L'usuari proporciona les dades requerides per crear un perfil (nom d’usuari, contrasenya i repetició de contrasenya)
3) El sistema valida les dades i crea un nou perfil per l'usuari
4) L'usuari rep una confirmació del registre i pot ara accedir amb les seves noves credencials

Errors Possibles i Cursos Alternatius:

2a) L'usuari proporciona un nom d’usuari ja existent o que no compleix els requisits: el sistema mostra un missatge d'error i demana un nou nom

2b) L’usuari propociona una contrasenya que no compleix els requisits de seguretat: el sistema avisa de l’error i demana una nova contrasenya
2c) L’usuari propociona una repetició de contrasenya diferent a la contrasenya: el sistema avisa de l’error i demana reintroduir les dades contrasenya i repetició

Cas d'ús #22 - Sortir del programa

Actor: Usuari no registrat
Precondició: L'usuari està utilitzant el programa i vol tancar-lo
Detonant: L'usuari vol sortir de la plataforma
Escenari Principal:

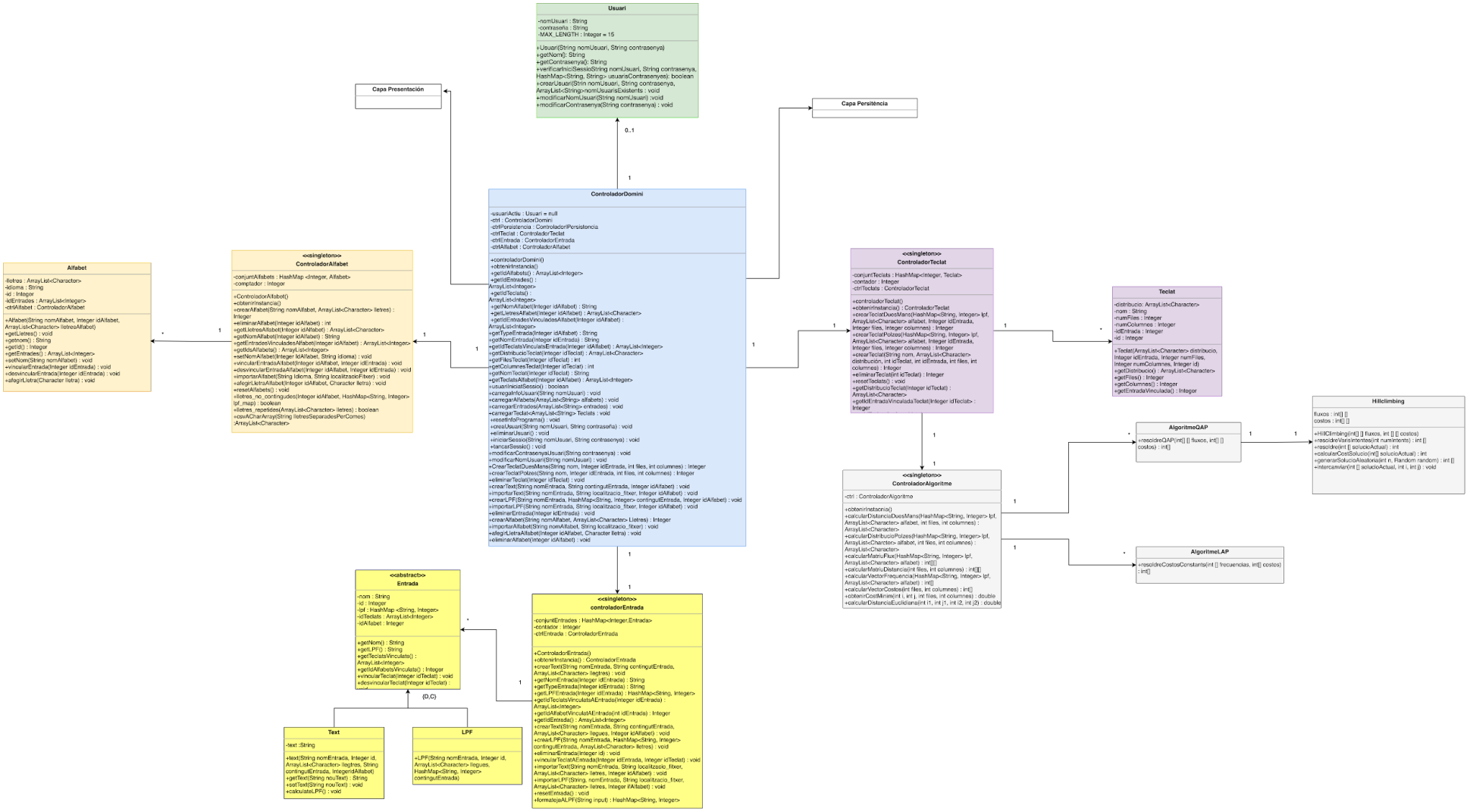
1) L’usuari invoca la instrucció per tancar el programa
2) L’usuari confirma la voluntat de tancar el programa
3) El programa es tanca

Errors Possibles i Cursos Alternatius:

+ []L’usuari cancel·la l'acció de tancament: el sistema cancel·la l'operació

2. MODEL CONCEPTUAL DE DADES

2.1. DIAGRAMA DEL MODEL CONCEPTUAL DE DADES



2.2. DESCRIPCIÓ DEL DIAGRAMA DE CLASSES

Usuari

Nom de la Classe

Usuari

Breu Descripció de la Classe

La classe Usuari representa un usuari al sistema, incloent-hi detalls com el nom i la contrasenya. A més, proporciona funcionalitats per a la creació i l'autenticació d'usuaris.

Cardinalitat

Un usuari sempre tindrà un controlador de domini, però un controlador de domini pot tenir o no un usuari.

Descripció dels Atributs

- nomUsuari: String que desa el nom de l'usuari.
- contrasenya: String que emmagatzema la contrasenya de l'usuari.
- MAX_LENGTH (static final): Enter que defineix la longitud màxima per al nom d'usuari i la contrasenya, fixat en 15 caràcters.

Descripció de les Relacions

No hi ha relacions explícites amb altres classes, més enllà de les interaccions amb estructures de dades com HashMap, ArrayList i el controlador de domini.

Descripció dels Mètodes

- Constructor Usuari(String nomUsuari, String contrasenya):: Usuari: Inicialitza un usuari amb un nom i una contrasenya.
- getContrasenya():: String: Retorna la contrasenya de l'usuari.
- getNom():: String: Retorna el nom de l'usuari.

- modificarNomUsuari(String nomUsuari):: void: Permet modificar el nom de l'usuari.
- modificarContrasenya(String nomUsuari):: void: Permet modificar la contrasenya de l'usuari.
- iniciarSessio(String nomUsuari, String contrasenya, HashMap<String, String> nomUsuariContrasenyes):: Usuari: Mètode estàtic per iniciar sessió, verificant nom dusuari i contrasenya contra un HashMap.
- crearUsuari(String nomUsuari, String contrasenya, ArrayList<String> nomUsuariExistents):: Usuari: Mètode estàtic per crear un nou usuari, verificant que el nom no existeix ja a la llista proporcionada.
- verificarIniciSessio(String nomUsuari, String contrasenya, HashMap<String, String> usuarisContrasenyes):: boolean: Verifica si l'inici de sessió és correcte tenint en compte un nom d'usuari i una contrasenya, comparant-los amb un HashMap.

Text

Nom de la Classe

Text

Breu Descripció de la Classe

La classe Text representa un text associat a una entrada. Inclou contingut textual i associacions a un alfabet i teclats. Estén la classe Entrada, heretant-ne els atributs i les funcionalitats.

Cardinalitat

Com que és una extensió d'entrada, sabem que la seva relació serà Disjoint i Complete.

Descripció dels Atributs

- text: Cadena que desa el contingut textual del text.
- (heretats d'Entrada): Inclou nom, id, idAlfabet, idTeclats i lpf (Llista de freqüència de paraules), juntament amb altres possibles atributs no especificats en aquest fragment de codi.

Descripció de les Relacions

La classe Text és una subclasse d'Entrada, heretant-ne totes les relacions i els atributs.

Descripció dels Mètodes

- Constructor Text(String nomEntrada, Integer id, ArrayList<Character> lletres, String contingutEntrada, Integer idAlfabet): Inicialitza un text amb nom d'entrada, identificador, llista de caràcters, contingut textual i un identificador d'alfabet.
- getText():: String: Retorna el contingut textual del text.
- setText(String nouText):: void: Permet establir o modificar el contingut textual del text.

- calculateLPF():: void: Mètode privat per calcular la Freqüència de Paraules (LPF) del contingut textual, processant i comptant l'ocurrència de cada paraula.

Teclat

Nom de la Classe

Teclat

Breu Descripció de la Classe

La classe Teclat representa un teclat dins del sistema, incloent-hi detalls com el nom, les dimensions, l'identificador i l'entrada vinculada. Se centra en la gestió de la distribució específica de caràcters al teclat.

Cardinalitat

Podem dir que un teclat sempre tindrà un controlador de teclat i que un controlador de teclat podrà tenir o no molts teclats.

Descripció dels Atributs

- distribucio: Una llista de caràcters (ArrayList<Character>) que representa la distribució de caràcters al teclat.
- nom: Cadena que representa el nom del teclat.
- numFiles: Sencer que indica el nombre de files del teclat.
- numColumnes: Sencer que representa el nombre de columnes del teclat.
- idEntrada: Sencer que serveix com a identificador de l'entrada vinculada al teclat.
- id: Sencer que actua com un identificador únic per a cada teclat.

Descripció de les Relacions

La classe està relacionada amb controlador de teclat

Descripció dels Mètodes

- Constructor Teclat(String nom, ArrayList<Character>, Integer idEntrada, Integer numFiles, Integer numColumnes, Integer id): Inicialitza un teclat amb el nom, distribució de caràcters, identificador d'entrada vinculada, número de files, número de columnes i identificador únic.
- getDistribucio():: ArrayList<Character>: Retorna la distribució de caràcters del teclat.

- getFiles():: Interger: Retorna el nombre de files del teclat.
- getNom():: String: Obté el nom del teclat.
- getColumnes():: Integer: Retorna el nombre de columnes del teclat.
- getIdEntradaVinculada():: Integer: Proporciona l'identificador de l'entrada vinculada al teclat.

LPF

Nom de la Classe

LPF

Breu Descripció de la Classe

La classe LPF representa una entrada al sistema que gestiona la freqüència de paraules i la seva associació amb teclats i alfabets específics. Estén la classe Entrada, heretant-ne els atributs i les funcionalitats.

Cardinalitat

Com que és una extensió d'entrada, la cardinalitat de LPF no s'especifica de manera exclusiva, però s'entén que serà com text, és a dir Disjoint i Complete.

Descripció dels Atributs

- (heretats d'Entrada): Inclou nom, id, idAlfabet, idTeclats i lpf (Llista de freqüència de paraules), juntament amb altres possibles atributs no especificats en aquest fragment de codi.
- lpf: És un HashMap<String, Integer> que emmagatzema el contingut de la LPF, és a dir, cada paraula i la seva freqüència.

Descripció de les Relacions

La classe LPF és una subclasse d'Entrada, heretant-ne totes les relacions i els atributs.

Descripció dels Mètodes

- Constructor LPF(String nomEntrada, Integer id, ArrayList<Character> lletres, HashMap<String, Integer> contingutEntrada, Integer idAlfabet): Inicialitza una entrada LPF amb nom, identificador, llista de caràcters, contingut de la LPF (paraula i freqüència) i un identificador d'alfabet. Inclou una validació per assegurar que les freqüències de les paraules no siguin negatives.

Entrada

Nom de la Classe

Entrada

Breu Descripció de la Classe

Entrada és una classe abstracta que representa un element del sistema encarregat de gestionar informació relacionada amb teclats i alfabets. Serveix com una classe base per a altres classes que utilitzen diferents tipus d'entrades.

Cardinalitat

Com que és una classe abstracta, Entrada no l'instància directament, sinó que proporciona una estructura i funcionalitats comunes per a les seves subclasses.

- **Descripció dels Atributs**
- nom: Cadena que representa el nom de l'entrada.
- id: Sencer que actua com a identificador únic per a l'entrada.
- lpf: HashMap<String, Integer> que emmagatzema la Llista de Freqüència de Paraules (LPF) associada amb l'entrada.
- idTeclats: ArrayList<Integer> que conté els identificadors dels teclats vinculats a l'entrada.
- idAlfabet: Sencer que serveix com a identificador de l'alfabet vinculat a l'entrada.
- Descripció de les Relacions
- Entrada actua com una superclasse per a altres classes que gestionen tipus específics d'entrades al sistema. No es detallen les relacions amb classes externes en aquest fragment de codi.

Descripció dels Mètodes

- getNom():: String: Retorna el nom de l'entrada.
- getLPF():: HashMap<String, Integer>: Retorna el HashMap que representa la LPF de l'entrada.
- getTeclatsVinculats():: ArrayList<Integer>: Proporciona una llista d'identificador de teclats vinculats a l'entrada.

- getIdAlfabetVinculat()::Integer: Retorna l'identificador de l'alfabet vinculat a l'entrada.
- vincularTeclat(Integer idTeclat):: void: Vincula un teclat a l'entrada, verificant que ja no estigui vinculat.
- desvincularTeclat(Integer idTeclat):: void: Desvincula un teclat de l'entrada, assegurant-vos que estigueu vinculat anteriorment.

Alfabet

Nom de la Classe

Alfabet

Breu Descripció de la Classe

La classe Alfabet representa l'alfabet d'un idioma, és a dir, un conjunt de lletres no repetides de caràcter. S'enfoca a la gestió de les lletres que componen l'alfabet i la seva relació amb altres entrades al sistema.

Cardinalitat

Hi pot haver múltiples instàncies d'Alfabet, cadascuna representant un alfabet diferent. On el controlador de l'alfabet tindrà molts alfabets i alfabet tindrà un sol controlador d'alfabet.

Descripció dels Atributs

- lletres: ArrayList<Character> que conté les lletres de l'alfabet.
- nom: Cadena que representa el nom o l'idioma de l'alfabet.
- id: Sencer que actua com a identificador únic per a l'alfabet.
- idEntrades: ArrayList<Integer> que conté els identificadors de les entrades associades a l'alfabet.

Descripció de les Relacions

Alfabet està relacionada amb la classe controlado Alfabet

Descripció dels Mètodes

- Constructor Alfabet(String nomAlfabet, Integer idAlfabet, ArrayList<Character> lletresAlfabet): Inicialitza un alfabet amb nom, identificador i lletres.
- getLletres():: ArrayList<Character>: Retorna les lletres de l'alfabet.
- getNom():: String: Retorna el nom o l'idioma de l'alfabet.
- getId():: Integer: Obté l'identificador de l'alfabet.
- getEntrades():: ArrayList<Integer>: Proporciona els identificadors de les entrades associades a l'alfabet.

- setNom(String nomAlfabet):: void: Permet modificar el nom o l'idioma de l'alfabet.
- vincularEntrada(Integer idEntrada):: void: Associa una entrada a l'alfabet, verificant que no estigui ja associada.
- desvincularEntrada(Integer idEntrada):: void: Desvincula una entrada de l'alfabet, assegurant-se que estigui actualment vinculada.
- afegirLletra(Character lletra):: void: Afegeix una nova lletra a l'alfabet, comprovant que ja no sigui present i que no sigui nul·la.

ControladorAlfabet

Nom de la Classe

ControladorAlfabet

Breu Descripció de la Classe

La classe ControladorAlfabet gestiona un conjunt d'alfabets, encarregant-se de crear, modificar i eliminar alfabets al sistema. Implementa el patró Singleton per garantir una única instància del controlador.

Cardinalitat

Singleton, només hi ha una instància de ControladorAlfabet a tot el sistema.

Descripció dels Atributs

- conjuntAlfabets: HashMap<Integer, Alfabet> que emmagatzema els alfabets gestionats, associats a un identificador únic.
- comptador: Sencer que es fa servir per assignar identificadors únics als alfabets.
- ctrlAlfabet (static): Instància estàtica de ControladorAlfabet per al patró Singleton.

Descripció de les Relacions

Interactua amb la classe Alfabet, gestionant-ne instàncies.

Descripció dels Mètodes

- Constructor ControladorAlfabet(): Inicialitza el controlador amb un conjunt buit d'alfabets.
- obtenirInstància(): Mètode estàtic per obtenir la instància única de ControladorAlfabet.
- crearAlfabet(String nomAlfabet, ArrayList<Character> lletres):: Integer: Crea un nou alfabet al sistema amb un nom i un conjunt de lletres.
- eliminarAlfabet(Integer idAlfabet):: void: Elimina un alfabet del conjunt gestionat pel controlador.

- getLletresAlfabet(Integer idAlfabet):: ArrayList<Character>: Obté les lletres d'un alfabet específic.
- getNomAlfabet(Integer idAlfabet):: String: retorna el nom o l'idioma d'un alfabet específic.
- getEntradesVinculadesAlfabet(Integer idAlfabet):: ArrayList<Integer>: Retorna els identificadors de les entrades associades a un alfabet específic.
- getIdAlfabets(): ArrayList<Integer>: Retorna una llista d'identificadors de tots els alfabets gestionats.
- setNomAlfabet(Integer idAlfabet, String idioma):: void: Modifica el nom d'un alfabet específic.
- vincularEntradaAlfabet(Integer idAlfabet, Integer idEntrada):: void: Associa una entrada a un alfabet específic.
- desvincularEntradaAlfabet(Integer idAlfabet, Integer idEntrada):: void: Desvincula una entrada d'un alfabet específic.
- importarAlfabet(String idioma, String localitzacioFitxer):: void Importa un alfabet des d'un fitxer de text.
- afegirLletraAlfabet(Integer idAlfabet, Character lletra):: void: Afegeix una nova lletra a un alfabet específic.
- resetAlfabets():: void: Reinicialitza el conjunt d'alfabets gestionats.
- lletres_no_contingudes(Integer idAlfabet, HashMap<String, Integer> lpf_map):: boolean: Verifica si les lletres d'una LPF estan contingudes en un alfabet específic.
- lletres_repetides(ArrayList<Character> lletres):: boolean: Comprova si hi ha lletres repetides en un conjunt de caràcters.
- csvACharArray(String lletresSeparadesPerComes):: ArrayList<Character>: Converteix una cadena de text amb lletres separades per comes en un ArrayList de caràcters.

ControladorAlgoritme

Nom de la Classe

ControladorAlgoritme

Breu Descripció de la Classe

ControladorAlgoritme és responsable de la gestió de dos algorismes de distribució de lletres en teclats. Implementa el patró Singleton per assegurar una única instància d'aquest controlador al sistema.

Cardinalitat

Singleton, el que significa que només existeix una instància de ControladorAlgoritme a tot el sistema.

Descripció dels Atributs

ctrl (static): Instància estàtica de ControladorAlgoritme utilitzada per implementar el patró Singleton.

Descripció de les Relacions

- Interactua amb les classes AlgoritmeQAP i AlgoritmeLAP per fer càlculs específics relacionats amb la distribució de lletres en teclats.

Descripció dels Mètodes

- obtenirInstància(): Mètode estàtic que retorna lúnica instància de ControladorAlgoritme.
- calcularDistribucioDuesMans(HashMap>String, Integer> lpf, ArrayList<Character> alfabet, int files, int columnes):: ArrayList<Character>: Calcula la distribució òptima de lletres en un teclat per a ús amb dues mans.
- calcularDistribucioPolzes(HashMap<String, Integer> lpf, ArrayList<Character> alfabet, int files, int columnes):: ArrayList<Character>: Calcula la distribució òptima de lletres en un teclat dissenyat per ser utilitzat amb els polzes.
- CalculeuMatriuFlux(HashMap <String, Integer> lpf, ArrayList<Character> alfabet): int[][]: Mètode privat per calcular la matriu de flux utilitzada en el problema d'assignació quadràtica (QAP).

- calcularMatriuCostos(int files, int columnes):: int[][]: Mètode privat per calcular la matriu de costos utilitzada al QAP.
- calcularVectorFrequencies(HashMap<String, Integer> lpf, ArrayList<Character> alfabet):: int[]: Mètode privat per calcular el vector de freqüències utilitzat en el problema d'assignació lineal (LAP).
- calcularVectorCostos(int files, int columnes):: int[]: Mètode privat per calcular el vector de costos utilitzat al LAP.
- obtenirCostMinim(int i, int j, int files, int columnes):: double: Calcula el cost mínim d'una tecla en funció de la distància a les cantonades inferiors del teclat.
- calcularDistànciaEuclidiana(int i1, int j1, int i2, int j2):: double: Mètode privat per calcular la distància euclidiana entre dos punts, utilitzat per mesurar distàncies al teclat.

ControladorTeclat

Nom de la Classe

ControladorTeclat

Breu Descripció de la Classe

ControladorTeclat gestiona un conjunt de teclats i s'encarrega de crear, modificar i eliminar aquests teclats. Col·labora amb ControladorAlgoritme per crear distribucions òptimes de lletres als teclats per a cada entrada.

Cardinalitat

La classe implementa un patró Singleton, la qual cosa implica que només hi ha una instància de ControladorTeclat al sistema.

Descripció dels Atributs

- conjuntTeclats: HashMap<Integer, Teclat> que emmagatzema els teclats gestionats, associats a un identificador únic.
- comptador: Sencer que s'utilitza per assignar identificadors únics als teclats.
- ctrlTeclats (static): Instància estàtica de ControladorTeclat per al patró Singleton.

Descripció de les Relacions

Interactua amb la classe ControladorAlgoritme per calcular distribucions òptimes de lletres en teclats.
Gestiona instàncies de la classe Teclat.

Descripció dels Mètodes

- Constructor ControladorTeclat(): Inicialitza el controlador amb un conjunt buit de teclats.
- obtenirInstància(): Mètode estàtic per obtenir la instància única de ControladorTeclat.
- crearTeclatDuesMans(String nom, HashMap<String, Integer> lpf, ArrayList<Character> alfabet, Integer uidEntrada, Integer files, Integer

columnes):: Integer: Crea un teclat dissenyat per ser utilitzat amb les dues mans.

- crearTeclat(String nom, ArrayList<Character> distribucio, int idTeclat, int idEntrada, int files, int columnes):: Integer: Aquest mètode crear un teclat amb els parametres passat per referències i ho guarda en una col·lecció conjunt Teclats. Finalment retorna un idTeclat.
- crearTeclatPolzes(String nom, HashMap<String, Integer> lpf, ArrayList<Character> alfabet, Integer uidEntrada, Integer files, Integer columnes):: Integer: Crea un teclat optimitzat per ser utilitzat amb els polzes.
- eliminarTeclat(Integer idTeclat):: void: Elimina un teclat del conjunt gestionat pel controlador.
- resetTeclats():: void: Reinicialitza el conjunt de teclats, eliminant tots els existents.
- getDistribucioTeclat(Integer idTeclat):: ArrayList<Character>: Obté la distribució de lletres en un teclat específic.
- getIdEntradaVinculadaTeclat(Integer idTeclat):: ArrayList<Integer>: Retorna l'identificador de l'entrada vinculada a un teclat.
- getIdTeclats():: ArrayList<Integer>: Retorna una llista d'identificadors de tots els teclats creats.
- getNomTeclat(Integer idTeclat):: String: Obté el nom d'un teclat específic.
- getFilesTeclat(Integer idTeclat):: Integer: Retorna el nombre de files d'un teclat específic.
- getColumnesTeclat(Integer idTeclats):: Integer: obté el nombre de columnes d'un teclat específic.

AlgoritmeLAP

Nom de la Classe

AlgoritmeLAP

Breu Descripció de la Classe

AlgoritmeLAP és una classe que s'enfoca a resoldre problemes d'assignació lineal (LAP), específicament aquells on els costos d'assignació són constants. La classe utilitza un mètode per assignar recursos (representats per freqüències) a tasques (representades per costos) de manera eficient.

Descripció dels Mètodes

- resoldreCostosConstants(int[] frecuencies, int[] costos):: int[]: Aquest mètode resol un problema d'assignació lineal donat un vector de freqüències i un vector de costos. El mètode funciona sota la premissa que els costos són constants per a cada possible assignació al problema.

Funcionament del Mètode:

Es crea una matriu per a cada vector (freqüències i costos), on cada fila conté el valor i el seu índex original.

Aquestes matrius s'ordenen de manera que les freqüències estiguin en ordre descendent (de major a menor) i els costos en ordre ascendent (de menor a major).

Després, el mètode assigna les freqüències més altes als costos més baixos, seguint el principi que cal maximitzar l'assignació dels recursos més valuosos (majors freqüències) a les tasques menys costoses.

El resultat és un vector d'assignacions on cada índex del vector representa la tasca (índex al vector de costos) i el seu valor corresponent representa el recurs assignat (índex al vector de freqüències).

AlgoriteQAP

Nom de la Classe

AlgoritmeQAP

Breu Descripció de la Classe

AlgoritmeQAP està dissenyada per resoldre problemes d'assignació quadràtica, que són comuns en l‟optimització de la disposició física de recursos. Utilitza un enfocament de Hill Climbing per trobar una solució eficient al problema.

Descripció del Mètode

- resoldreQAP(int[][] fluxos, int[][] costos):: int[]: Aquest mètode resol un problema QAP utilitzant el mètode Hill Climbing.

Paràmetres:

fluxos: Una matriu de flux que representa les interaccions o el flux de recursos entre diferents localitzacions o tasques.

costos: Una matriu de costos que representa el cost d'assignar recursos a diferents localitzacions o la realització de tasques a diferents llocs.

Funcionament del Mètode:

Inicialment, es crea una instància de la classe HillClimbing, passant les matrius de flux i costos com a paràmetres.

Després, s'anomena el mètode resoldre VarisIntentents de HillClimbing, que intenta trobar la solució òptima del QAP fent diverses iteracions de l'algorisme de Hill Climbing, en aquest cas, 10 intents.

El mètode resoldreVarisIntentents torna un vector d'enters que representa les assignacions òptimes trobades per al problema QAP.

HillClimbing

Nom de la Classe

HillClimbing

Breu Descripció de la Classe

HillClimbing implementa l'algorisme de Hill Climbing, un mètode d'optimització que iterativament realitza ajustaments en una solució per millorar-ne la qualitat. La classe està dissenyada per treballar amb problemes que es poden representar mitjançant matrius de fluxos i costos.

Descripció dels Atributs

- fluxos: Matriu de fluxos que representa les interaccions o el flux de recursos entre diferents localitzacions o tasques.
- costos: Matriu de costos que representa el cost d'assignar recursos a diferents localitzacions o la realització de tasques a diferents llocs.

Descripció dels Mètodes

- Constructor HillClimbing(int[][] fluxos, int[][] costos): Inicialitza una instància de HillClimbing amb les matrius de fluxos i costos proporcionades.
- resoldreVarisIntentents(int numIntents):: int[]: Intenta resoldre el problema diverses vegades amb solucions inicials aleatòries i torna la millor solució trobada. Utilitza el mètode generarSolucioAleatoria per crear solucions inicials i després aplica l'algorisme Hill Climbing a cadascuna mitjançant el mètode resoldre.
- resoldre(int[] solucioActual):: int: Executa l'algorisme Hill Climbing per trobar una millora iterativa a partir d'una solució donada. Aquest mètode avalua la solució actual i fa intercanvis entre els seus elements per trobar una solució amb un cost menor.
- CalculeuCostSoluci(int[] solucioActual):: int: Calcula el cost d'una solució donada basat en les matrius de fluxos i costos.
- generarSolucioAleatoria(int n, Random random):: int[]: Genera una solució aleatòria d'una mida determinada. Aquest mètode es fa servir per crear un punt de partida per a l'algorisme Hill Climbing.

- intercanviar(int[] solucioActual, int i, int j):: void: Intercanvia dos elements en una solució. Aquest mètode es fa servir per modificar la solució actual en el procés de Hill Climbing.

ControladorDomini

Nom de la Classe

ControladorDomini

Breu Descripció de la Classe

Aquesta classe actua com un controlador de domini al sistema. Gestiona la capa de domini, coordinant les interaccions entre els usuaris, alfabets, teclats i entrades. Implementa el patró de disseny Singleton per assegurar una única instància d'aquesta classe al sistema.

Cardinalitat

Singleton: Hi ha una única instància de ControladorDomini al sistema.

Descripció dels Atributs

ctrlPersistència: Instància de ControladorPersistència per a la gestió de persistència.
ctrlTeclat: Instància de ControladorTeclat per a la gestió de teclats.
ctrlEntrada: Instància de ControladorEntrada per a la gestió d'entrades.
ctrlAlfabet: Instància de ControladorAlfabet per a la gestió dalfabets.
usuariActiu: Objecte de tipus Usuari que representa l'usuari actualment actiu al sistema.

Descripció de les Relacions

ControladorPersistencia: ControladorDomini interactúa amb aquesta classe per realitzar operacions de persistència de dades. A través d'aquesta relació, es gestionen les operacions de càrrega i emmagatzematge de la informació dels usuaris, alfabetos, teclats i entrades.

ControladorTeclat: ControladorDomini estableix una relació amb ControladorTeclat per a la gestió dels teclats dins del sistema. Aquesta connexió permet a ControladorDomini delegar la creació, modificació i eliminació de teclats, així com consultar els seus detalls.

ControladorEntrada: ControladorDomini es relaciona amb ControladorEntrada per administrar les entrades, incloent textos i llistes de paraules freqüents. Aquesta associació facilita les operacions de creació, importació, eliminació i vinculació d'entrades amb altres elements del sistema.

ControladorAlfabet: ControladorDomini utilitza ControladorAlfabet per la gestió dels alfabetos. Aquesta relació permet realitzar accions sobre els alfabetos, com la seva creació, modificació, importació i eliminació, així com la seva vinculació amb entrades específiques.

Usuari: ControladorDomini manté una relació amb l'entitat Usuari per gestionar les accions de l'usuari actiu. Aquesta relació és clau per a personalitzar l'experiència de l'usuari, permetent a ControladorDomini realitzar operacions basades en el context de l'usuari actual.

Descripció dels Mètodes

- ControladorDomini(): Constructor privat. Inicialitza una instància de ControladorDomini i estableix les instàncies de controladors per a la persistència, teclats, entrades i alfabets.
- obtenirInstància():: ControladorDomini: Mètode estàtic que assegura l'existència duna única instància de ControladorDomini en el sistema (patró Singleton). Si no hi ha una instància, la crea i la torna; si ja existeix, simplement la torna.
- getIdAlfabets():: ArrayList<Integer>: Retorna una llista dels identificadors dels alfabets associats a l'usuari actual. Llança excepció si no hi ha cap usuari amb sessió iniciada.
- getIdEntrades():: ArrayList<Integer>: Retorna una llista d'identificadors de les entrades associades a l'usuari actual. Llança excepció si no hi ha cap usuari amb sessió iniciada.
- getIdTeclats():: ArrayList<Integer>: Retorna una llista dels identificadors dels teclats associats a l'usuari actual. Llança excepció si no hi ha cap usuari amb sessió iniciada.

- getNomAlfabet(Integer idAlfabet):: String: Retorna el nom de l'alfabet identificat per idAlfabet. Llança excepció si no hi ha un usuari amb sessió iniciada o si l'alfabet no existeix.
- getLletresAlfabet(Integer idAlfabet):: ArrayList<Character>: Retorna les lletres de l'alfabet identificat per idAlfabet. Llança excepció si no hi ha un usuari amb sessió iniciada o si l'alfabet no existeix
- getIdEntradesVinculadesAlfabet(Integer idAlfabet):: ArrayList<Integer>: Retorna els identificadors de les entrades vinculades a l'alfabet identificat per idAlfabet. Llança excepció si no hi ha un usuari amb sessió iniciada o si l'alfabet no existeix.
- getTypeEntrada(Integer idEntrada):: String: Retorna el tipus de l'entrada identificada per idEntrada, que pot ser "text" o "lpf". Llança excepció si no hi ha un usuari amb sessió iniciada o si l'entrada no existeix.
- getNomEntrada(Integer idEntrada):: String: Retorna el nom de l'entrada identificada per idEntrada. Llança excepció si no hi ha un usuari amb sessió iniciada o si l'entrada no existeix.
- getIdTeclatsVinculatsAEntrada(Integer idEntrada):: ArrayList<Integer>: Retorna els identificadors dels teclats vinculats a l'entrada identificada per idEntrada. Llança excepció si no hi ha un usuari amb sessió iniciada o si l'entrada no existeix.
- getDistribucioTeclat(Integer idTeclat):: ArrayList<Character>: Retorna la distribució de tecles del teclat identificat per idTeclat. Llança excepció si no hi ha un usuari amb sessió iniciada o si el teclat no existeix.
- getFilesTeclat(Integer idTeclat):: int: Retorna el nombre de files del teclat identificat per idTeclat. Llança excepció si no hi ha un usuari amb sessió iniciada o si el teclat no existeix.
- getColumnesTeclat(Integer idTeclat):: int: Retorna el nombre de columnes del teclat identificat per idTeclat. Llança excepció si no hi ha un usuari amb sessió iniciada o si el teclat no existeix.
- getNomTeclat(Integer idTeclat):: String: Retorna el nom del teclat identificat per idTeclat. Llança excepció si no hi ha un usuari amb sessió iniciada o si el teclat no existeix.

- getTeclatsAlfabet(Integer idAlfabet):: ArrayList<Integer>: Retorna els identificadors dels teclats associats a l'alfabet identificat per idAlfabet. Llança excepció si no hi ha un usuari amb sessió iniciada o si l'alfabet no existeix.
- usuariIniciatSessio():: boolean: Comprova si hi ha un usuari amb la sessió iniciada. Torna true si hi ha un usuari actiu, i false en cas contrari.
- crearUsuari(String nomUsuari, String contrasenya):: void: Crea un nou usuari amb el nom nomUsuari i la contrasenya contrasenya. Llança excepció si ja existeix el nom d'usuari.
- eliminarUsuari():: void: Elimina l'usuari actiu actual. Llança excepció si no hi ha cap usuari amb sessió iniciada.
- iniciarSessio(String nomUsuari, String contrasenya):: void: Inicia sessió amb el nom d'usuari nomUsuari i la contrasenya contrasenya. Llança excepció si ja hi ha una sessió iniciada o si les credencials són incorrectes.
- cerrarSessio():: void: Tanca la sessió de l'usuari actiu. Llança excepció si no hi ha cap usuari amb sessió iniciada.
- modificarContrasenyaUsuari(String contrasenya):: void: Modifica la contrasenya de l'usuari actiu amb contrasenya. Llança excepció si no hi ha cap usuari amb sessió iniciada.
- modificarNomUsuari(String nomUsuari):: void: Modifica el nom de l'usuari actiu amb nomUsuari. Llança excepció si no hi ha cap usuari amb sessió iniciada.
- crearTeclatDuesMans(String nom, Integer idEntrada, int fils, int columnes):: Integer: Crea un teclat optimitzat per a dues mans, basat en una entrada i dimensions específiques. El teclat s'identifica per nom associat a l'entrada idEntrada i té un nombre definit de fils i columnes. Torna l'identificador del teclat creat. Llança excepció si hi ha errors a la creació.
- crearTeclatPolzes(String nom, Integer idEntrada, int fils, int columnes):: Integer: Crea un teclat optimitzat per a polzes, basat en una entrada i dimensions específiques. Funciona de manera similar a crear TeclatDuesMans, però el teclat està optimitzat per ser usat amb els polzes.
- eliminarTeclat(Integer idTeclat):: void: Elimina el teclat identificat per idTeclat. Llança excepció si el teclat no existeix o si hi ha un error a l'eliminació.
- crearText(String nomEntrada, String contingutEntrada, Integer idAlfabet):: void: Crea una entrada de text al sistema amb el nom nomEntrada, contingut

contingutEntrada, i associada a l'alfabet identificat per idAlfabet. Llança excepció si hi ha un error a la creació.

- importarText(String nomEntrada, String localizacio_fitxer, Integer idAlfabet):: void: Importa un text des d'una ubicació de fitxer especificada localizacio_fitxer, creant una entrada amb el nom nomEntrada i associada a l'alfabet idAlfabet. Llança excepció si hi ha errors a la importació.
- crearLPF(String nomEntrada, HashMap<String, Integer> contingutEntrada, Integer idAlfabet):: void: Crea una entrada de llista de paraules freqüents (LPF) amb el nom nomEntrada, contingut contingutEntrada (un mapa de paraules i les seves freqüències), i associada a l'alfabet idAlfabet. Llança excepció si hi ha errors a la creació.
- importarLPF(String nomEntrada, String localizacio_fitxer, Integer idAlfabet):: void: Importa una LPF des d'una ubicació de fitxer localizacio_fitxer, creant una entrada amb el nom nomEntrada i associada a l'alfabet idAlfabet. Llança excepció si hi ha errors a la importació.
- eliminarEntrada(Integer idEntrada):: void: Elimina l'entrada identificada per idEntrada. Llança excepció si l'entrada no existeix o si hi ha un error a l'eliminació.

- crearAlfabet(String nomAlfabet, ArrayList<Character> lletres):: Integer: Crea un alfabet al sistema amb el nom nomAlfabet i les lletres lletres. Retorna l'identificador de l'alfabet creat. Llança excepció si hi ha errors a la creació
- importarAlfabet(String nomAlfabet, String localizacio_fitxer):: void: Importa un alfabet des d'una ubicació de fitxer localizacio_fitxer, creant un alfabet amb el nom nomAlfabet. Llança excepció si hi ha errors a la importació.
- afegirLletraAlfabet(Integer idAlfabet, Character letra):: void: Afegeix la lletra lletra a l'alfabet identificat per idAlfabet. Llança excepció si l'alfabet no existeix o si hi ha un error a l'addició.
- eliminarAlfabet(Integer idAlfabet):: void: Elimina l'alfabet identificat per idAlfabet, així com totes les entrades i teclats derivats d'aquest. Llança excepció si l'alfabet no existeix o hi ha un error en l'eliminació.

3. BREU DESCRIPCIÓ DE L'ALGORTIME

La funcionalitat principal del programa és l'ordenació de les lletres d'un alfabet de mida **n**, per poder ser col·locades en un teclat. Hem optat per definir el layout com una matriu de **f** files per **c** columnes. És a dir, només es poden tenir teclats rectangulars, i en cas que la mida de l'alfabet sigui inferior a **f*c**, s'afegiran lletres en blanc perquè encaixi el layout.

La classe **ControladorAlgortime** és la que s'encarrega de generar aquestes ordenacions. Hi ha dues funcions per ordenar les lletres d'un alfabet, en funció de l'objectiu que es busqui, és a dir, quin és el cost que es vol minimitzar. Les dues reben els mateixos paràmetres i són suficients per tenir una noció absoluta del problema:

- `HashMap<String, Integer> lpf`: on tenim les paraules i el seu nombre d'aparicions
- `ArrayList<Character> alfabet`: on tenim les lletres a ordenar
- `int files`: nombre de files del teclat
- `int columnes`: nombre de columnes del teclat

El que retornen les dues funcions és un `ArrayList<Character>` amb les lletres ordenades, on les **f** primeres lletres són les que van a la primera fila del teclat i així successivament.

Ara veurem les dues funcions i què es busca amb cadascuna d'elles.

3.1. CALCULAR DISTRIBUCIÓ POLZES

Aquesta manera d'ordenar les lletres busca reduir el cost del moviment que han de fer els polzes quan escrivim en un teclat de mòbil. Per a resoldre el problema, el traduïm a un **LAP**. Aquest LAP és resolt per la classe **AlgortimeLAP** que resol aquest tipus de problemes. Tot i això, és una variació del problema, ja que en comptes d'una matriu de costos tenim un **vector de costos**. Això és així perquè el cost d'una tecla del teclat és independent de la lletra que se li assigni, i com que el cost total de les assignacions és la multiplicació de la freqüència de cada lletra pel cost de la seva tecla assignada, resoldrem el problema **assignant les lletres amb més freqüència a les tecles amb menys cost**. Dit de manera més simple, les lletres que fem servir més sovint estaran més a prop dels polzes. **ControladorAlgortime** crida a la següent funció de **AlgortimeLAP**:

```
int[] resoldreCostosConstants(int[] freqüències, int[] costos)
```

Càlcul de les freqüències

Per calcular les freqüències, recorrem la **lpf** i per cada lletra augmentem la seva freqüència en el nombre de vegades que apareix la paraula. D'aquesta manera, creem un `int[]` on per cada lletra (mateixos índexos que al vector d'entrada **alfabet**) tenim la seva freqüència. Cal tenir en compte que si el vector de freqüències no és de mida f*c, s'afegiran lletres amb freqüència zero, perquè encaixi el layout.

Càlcul dels costos

Per calcular el cost de cada tecla, calculem per a cada tecla el **mínim** de les distàncies d'aquesta tecla a qualsevol de les cantonades inferiors del teclat. En un layout de **f** files per **c** columnes, decidim que la tecla (0, 0) és la cantonada esquerra superior, i la (**f** - 1, **c** - 1) és la cantonada dreta inferior. Ens quedem sempre amb el mínim ja que ens és indiferent amb quin dels dos polzes la polsem.

En resum, **ControladorAlgortime** genera un `int[]` de **f*c** on per cada tecla tenim el seu cost.

AlgortimeLAP

Aquesta classe resoldrà les assignacions, com hem dit anteriorment, assignant les lletres amb més freqüència a les tecles amb menys cost. Retornarà un `int[]` on per cada element **i** s'indica l'índex de la tecla associada a la **lletra alfabet[i]**.

3.2. CALCULAR DISTRIBUCIÓ DUES MANS

Aquesta altra manera d'ordenar les lletres busca minimitzar el cost a l'hora d'escriure en un teclat de portàtil. Ho fa transformant el problema a un **QAP**, que s'encarrega de resoldre la classe **AlgortimeQAP**, mitjançant la funció:

`int[] resoldreQAP(int[][] fluxos, int[][] costos)`.

Càlcul dels fluxos

Els fluxos representen la quantitat de vegades que dues lletres apareixen seguides. Per tant, calculem una matriu de **(f*c)*(f*c)** recorrent la **lpf** i incrementant el valor de la matriu corresponent per cada parell de lletres de cada paraula, en la mateixa magnitud que el nombre de vegades que apareix la paraula.

Càlcul dels costos

En aquest cas, tenim una matriu de **(f*c)*(f\c)** on per cada parell de tecles tenim el cost de polsar-les consecutivament. En calcular aquests costos penalitzem només dues coses:

- Que dues tecles estiguin a la mateixa columna, amb un cost de 2 (greu). Utilitzar el mateix dit dues vegades és ineficient, ja que busquem distribuir les polzades per tots els dits igualment.
- Que dues tecles estiguin a la mateixa meitat (vertical) del layout, amb un cost de 1. Utilitzar la mateixa mà dues vegades seguides és ineficient.

AlgortimeQAP

Per a resoldre el problema es fa ús de l'Algortime HillClimbing, implementat en la classe **HillClimbing**. El que fa aquest Algortime és definir una solució inicial aleatòria i anar fent tots els intercanvis d'assignacions possibles mentre es redueixi el cost total. Com que és un Algortime amb molt risc de trobar mínims locals, la funció `public int[] resoldreVarisIntents(int numIntents)` de la classe **HillClimbing** resol el mateix problema amb diferents solucions inicials diverses vegades i retorna la millor. La classe **AlgortimeQAP** crida aquesta funció amb un nombre d'intents de 50.