**DISSENY D’UN TECLAT**

# **Descripció de les Capes**

## **Projectes de Programació:** Quadrimestre Tardor 2023-24

## **Versió del lliurament: 2.1**

## **2ª Entrega**

**Grup 12.3:**

Pau Costafreda Tur

Xavier López Mañes

Hakima Marouan Marouan

Diego Rodriguez Milagro 

# Descripció de la capa domini

## 1.CtrlDomini

**Breu descripció de la classe:** S’utilitza per la funcionalitat de les classes del domini i poder provar el seu funcionament.

**Cardinalitat:** Una.

**Descripció dels atributs:**

* instance: Instància única de la classe de CtrlDomini**.**
* controladorPerfil: Instància única de la classe de CtrlPerfil**.**
* controladorTeclat: Instància única de la classe de CtrlTeclat**.**

**Descripció dels mètodes:**

* getInstance(): crea una instància de Ctrl Domini.
* getControladorPerfil(): retorna el CtrlPerfil ja inicialitzat dins la classe CtrlDomini.
* getControladorTeclat(): retorna el CtrlTeclat ja inicialitzat dins la classe CtrlDomini.

**Descripció de les relacions:**

* Relació d’associació amb “CtrlPerfil”: el controlador del domini crea una instancia de CtrlPerfil i accedeix a ella quan vol començar amb les funcionalitats del sistema.
* Relació d’associació amb “CtrlTeclat”: el controlador del domini crea una instancia de CtrlTeclat i accedeix a ella quan vol començar amb les funcionalitats del sistema.

## 2.CtrlPerfil

**Breu descripció de la classe:** La classe CtrlPerfil gestiona el conjunt de perfils, i les operacions que s’han de fer respecte a aquests.

**Cardinalitat:** Una.

**Descripció dels atributs:**

* instance: Instància única de la classe de CtrlPerfil**.**
* buff: vector de chars que emmagatzema un conjunt de caràcters utilitzats en l’algoritme MD5.
* int\_buff: vector de longs que emmagatzema un conjunt de valors de 64 bits convertits a enters (longitud de 16 elements) utilitzats durant els càlculs de l'algoritme MD5.
* A,B,C,D: longs que emmagatzemen valors inicials constants per a l'algoritme MD5.
* K: vector de longs queemmagatzema una sèrie de constants de 32 bits utilitzades en el procés de l'algoritme MD5.
* S: vector d’enters que emmagatzema una sèrie de constants de desplaçament utilitzades en el procés de l'algoritme MD5.

**Descripció dels mètodes:**

* LongF(long X, long Y, long Z): funció que retorna el resultat de fer una operació a nivell de bit entre els 3 paràmetres.
* LongG(long X, long Y, long Z): funció que retorna el resultat de fer una operació a nivell de bit entre els 3 paràmetres.
* LongH(long X, long Y, long Z): funció que retorna el resultat de fer una operació a nivell de bit entre els 3 paràmetres.
* LongI(long X, long Y, long Z): funció que retorna el resultat de fer una operació a nivell de bit entre els 3 paràmetres.
* buff\_to\_intbuff: funció que agafa un array de chars buff i el transforma en una array de ints intbuff.
* long left\_bit\_shift(long AA, int S): donat un long AA i un int S, retorna AA shiftat S bits a l’esquerra.
* process(): processa el inbuff amb les operacions respectives de l’algoritme MD5.
* String output(): retorna un string amb el format de sortida de MD5 a partir de A, B, C i D.
* String MD5(String password): donat un string, retorna el resultat de passar el string per MD5.
* CtrlPerfil getInstance(): si ja existia una instància de la classe CtrlPerfil, la retorna. Si no, crea una nova instància de CtrlPerfil.
* Perfil signupPerfil(String user, String password, Perfil.ma ma): crea un nou perfil amb els atributs que es passen per paràmetre i el retorna.
* Perfil logginPerfil(String user, String password): retorna el perfil que té com a nom user i contrasenya password.
* modificarPerfil(String user, String password, Perfil.ma ma): modifica els atributs del Perfil amb nom user.
* eliminar\_perfil(String user): elimina el perfil amb nom user.

**Descripció de les relacions:**

* Relació d’associació amb la classe “Perfil”: gestiona els perfils del sistema.
* Relació d’associació amb la classe “CtrlDomini”: el controlador de domini crea una instancia de CtrlPerfil i s'utilitza en les funcionalitats necessàries del sistema.

## 3.Perfil

**Breu descripció de la classe:** La classe "Perfil" gestiona la informació associada a un perfil, incloent el nom d'usuari, contrasenya i la mà dominant de l'usuari**.**

**Cardinalitat:** Una per cada perfil.

**Descripció dels atributs:**

* username: String que emmagatzema el nom de l'usuari.
* password: String que conté la contrasenya de l'usuari.
* maDominant: Enumeració que representa la mà dominant de l'usuari (esquerra o dreta).

**Descripció de les relacions:**

* Relació d’associació amb la classe “Teclat”: indica qui crea el teclat.
* Relació d’associació amb la classe “CtrlPerfil”: indica que el perfil és gestionat pel CtrlPerfil.

## 4.CtrlTeclat

**Breu descripció de la classe:** La classe CtrlTeclat gestiona el conjunt de teclats, i les operacions que s’han de fer respecte a aquests.

**Cardinalitat:** Una

**Descripció dels atributs:**

* instance:: Instancia única de la classe de CtrlTeclat**.**

**Descripció dels mètodes:**

* CtrlTeclat getInstance(): retorna una instància singleton de la classe “CtrlTeclat”. Si una instància no existeix, en crea una de nova.
* crearTeclat(String username, String nomTeclat, Perfil.ma ma, int pulgar\_x, intpulgar\_y, int altura\_inicial, int amplada\_inicial, String idioma, String palabrasF, String text): crea un teclat nou amb els paràmetres especificats, realitza validacions d'entrada i l'afegeix a la capa de persistència de dades.
* obtenirEstadistiquesUltimaDistribucio(String username, String nomTeclat): obté estadístiques de la darrera distribució d'un teclat ja existent amb nomTeclat i usuari username específic.
* modificarTeclat(String username, String nomTeclat, int nova\_altura, int nova\_amplada): modifica la distribució d'un teclat ja existent, amb nom nomTeclat i usuari proprietari username, canviant-ne l'alçada i l'amplada.
* eliminar\_teclat(String username, String nomTeclat): elimina un teclat ja existent d’un usuari específic amb nomTeclat de la capa de persistència de dades.
* getTeclat(String username, String nomTeclat): retorna un teclat ja existent i identificat per a un usuari i nomTeclat determinat.
* obtenirTeclatsUsuari(String username): recupera una llista de teclats associats a un usuari específic ja existent.
* consultarUltimaDistribucio(String username, String nomTeclat): retorna l'última distribució d'un teclat nomTeclat específic que pertany a un usuari.

**Descripció de les relacions:**

* Relació d’associació amb la classe “Teclat”: gestiona els teclats del sistema.
* Relació d’associació amb la classe “CtrlDomini”: el controlador de domini crea una instancia de CtrlTeclat i s'utilitza en les funcionalitats necessàries del sistema.

## 5.Teclat

**Breu descripció de la classe:** La classe “Teclat” representa un teclat personalitzat que inclou les preferències de disposició de tecles d'un usuari. Conté mètodes per gestionar les distribucions de tecles i la seva modificació, així com per consultar aquestes distribucions.

**Cardinalitat:** Una per cada teclat.

**Descripció dels atributs:**

* pulgar\_x: Enter de la posició en l'eix X preferida per al polze de l'usuari.
* pulgar\_y: Enter de la posició en l'eix Y preferida per al polze de l'usuari.
* nomUsuari: String del nom de l'usuari associat al teclat.
* nomTeclat: String del nom del teclat.
* dataCreacio: Date de la data de creació del teclat.
* distribucions: Llista de distribucions associades al teclat.
* alfabet: Objecte que representa l'alfabet associat al teclat.
* ma: Enumeració de perfil que representa la mà preferida per l'usuari.
* algoritmeUtilitzat: Algoritme que l’usuari escull utilitzar entre QAP i Hill Climbing.

**Descripció dels mètodes:**

* getNomTeclat(): retorna el nom identificador del teclat.
* getMa(): retorna la mà preferida de l'usuari.
* getNomUsuari(): retorna el nom identificador de l'usuari.
* getAlfabet(): retorna l'alfabet associat al teclat.
* getPulgar\_y(): retorna la posició Y preferida per al polze de l'usuari.
* getPulgar\_x(): retorna la posició X preferida per al polze de l'usuari.
* getDataCreacio(): retorna la data de creació del teclat.
* getDistribucions(): retorna la llista de distribucións associades al teclat.
* llegirFilecomAString(String fileName): llegeix el contingut d'un fitxer fileName i el retorna com a cadena de string.
* afegirDistribucio(Distribucio distribucio, String nomTeclat): afegeix una distribució a la llista de distribucions associades al teclat.
* Teclat(String nomUsuari, Perfil.ma ma, String nomTeclat, int pulgar\_x, int pulgar\_y, Date dataCreacio, List<Distribucio> distribucions, Alfabet alfabet): constructor per crear un Teclat nou amb les característiques especificades.
* Teclat(String nom d'usuari, Perfil.ma ma\_usuari, String nomTeclat, int pulgar\_x, int pulgar\_y, int altura\_inicial, int amplada\_inicial, String idioma, String palabrasF, String fileName) llança IOException: és un altre constructor que permet crear un nou Teclat amb diferents paràmetres.
* modificarDistribucio(String nomTeclat, int nova\_altura, int nova\_amplada): afegeix una nova distribució a la llista de distribucions del teclat amb noves alçada i amplada.
* consultarDistribucions(): retorna una llista de distribucions amb identificador nomTeclat.

**Descripció de les relacions:**

* Relació d’associació amb la classe “CtrlTeclat”: indica que el teclat és gestionat pel CtrlTeclat.
* Relació d’associació amb “Perfil”: indica que el teclat es creat per un perfil.
* Relació d’associació amb “Alfabet”: indica que al teclat li correspon un alfabet.
* Relació d’associació amb “Distribució”: indica que el teclat té una o més distribucions.

## 6.Algoritme

*[Aquesta classe està explicada amb més detall al document Descripció de les estructures de dades i algorismes]*

**Breu descripció de la classe:** La classe és el procediment per generar una distribució bona per un cert input. Es pot escollir un de dos algoritmes per crear la distribució. Una és amb Hill Climbing, un algoritme en el que es fa una búsqueda de la millor solució local donat un teclat endreçat aleatòriament. I l’altra opció és l’algoritme QAP en el qual, en comptes de buscar en l’espai de solucions completes, es comença desde la solució buida i es va construint la solució a partir de cotes mesurades amb un cert procediment fins arribar a la solució completa.

**Cardinalitat:** Una per cada distribució.

**Descripció dels mètodes:**

Mètodes de Hill Climbing:

* dist(char c1, char c2, char[][] teclat): calcula la distància entre les lletres c1 i c2 en la distribució teclat. Per fer-ho, es fa un recorregut per trobar la posició de les dues lletres en coordenades x i y. Quan ja s’han trobat les dues posicions es calcula la distància que hi ha entre elles.
* heurstica(char[][] teclat, int[][] costLletres, String alfabet): Aquesta primera funció heurística calcula a distancia necessària per teclejar totes les paraules del input. Per fer-ho s’utilitza la funció dist per cada parell de lletres adjacents a l’input, i multiplicant-la pel flux de les lletres, es suma a una variable h que es el resultat final de la heurística.
* heuristica2(char[][] teclat, M ap<Character, Integer> lletraFreq, int x, int y): La segona funció heurística calcula, donat un punt sobre el teclat que simula la posició natural del polze de l’usuari, es retorna un valor que mesura com de prop estan les lletres més utilitzades respecte d’aquest punt. El calcul es fa per cada lletra del teclat, es calcula la distància respecte a la tecla (x,y) i es suma, multiplicant-la per la freqüència de la lletra, a la variable h que un cop acabat el bucle es retorna com a resultat de la funció.
* heuristicaTotal(char[][] teclat, int[][] costLletres, Map<Character, Integer> lletraFreq, int x, int y, String alfabet): calcula la heurística total de la distribució del teclat, fent una crida a les dues funcions heurístiques i sumant-les, la primera amb pes 1 i las segona amb pes 5.
* cambiar(char[][] teclat, int fila1, int fila2, int col1, int col2): retorna el teclat passat per paràmetre pero amb la lletra de la fila i columna 1 a la posició de la fila i columna 2 i viceversa.
* copiaT(char[][] teclat): retorna una copia del teclat passat per paràmetre.
* hillClimbing(char[][] teclat, int[][] cotLletres, Map<Character, Integer> lletraFreq, int x, int y, String alfabet): Aquest és el mètode principal de la classe i el que es crida en caso de voler fer el teclat a partir de Hill Climbing. Aquesta funció es basa en la creació de teclats auxiliars generats a partir de la modificació del teclat original i els teclat “fills”. Mentre es van creant aquests teclats, es calcula la qualitat de cadascun per així retornar el millor teclat un cop acabada la búsqueda. Les modificacions dels teclats, s’apliquen al millor teclat trobat en el moment en el que s’apliquen i es basa en l’intercanvi de cada lletra per la què tenen a la seva dreta i la que tenen a baix. Creant així 2N teclats auxiliars (sent N el número de lletres de l’alfabet seleccionat).

Mètodes de QAP:

* asignacion(ArrayList<ArrayList<Integer>> matriu, int[] assignació, int pos, int n0Final): funció recursiva que retorna una assignació òptima de la matriu. És a dir que retorna el màxim numero de zeros assignats sense que hi hagin dos zeros marcats en la mateixa línia i columna.
* liniesMinimes(Boolean[] línies, Boolean[] columnes): retorna el número mínim de línies necessaries per cobrir tots els zeros de la matriu. Per fer-ho s’utilitza la funció assignació.
* zeroPerFilaIColumna(ArrayList<ArrayList<Integer>> C): Retorna cert si a cada fila i columna hi ha un 0 i retorna fals en caso contrari.
* hungarianAlgorithm(ArrayList<ArrayList<Integer>> C): Funció utilitzada per a que segons els costos de C calculats durant l’execució del qap trobi la millor combinació de les lletres i posicions i retorni el cost d’aquesta combinació.
* qap(int[][] costLletres, String alfabet, int alt, int amp): Aquesta funció és la segona funció principal de la classe que es crida quan es vol crear una distribució per un teclat amb qap. Per fer-ho, es parteix de una solució buida i es va construint el teclat provant totes les combinacions entre les tecles no col·locades i les posicions no ocupades fins que el teclat esta completament ple.

**Descripció de les classes privades:**

Classes de Hill Climbing:

* teclatInfo: aquesta classe conte dos atributs, char[][] teclat, que es la distribució del teclat i int h que es la heurística del teclat. S’utilitza per poder ordenar la cua de prioritat de solucions respecte a la qualitat de cada teclat.

Classes de QAP:

* par: és una classe amb dos valors int x y on cadascun representa els dos eixos del teclat.

**Descripció de les relacions:**

* Relació d’associació amb “Distribució”: Indica que algoritme té una o moltes distribucions.
* Relació d’associació amb “TeclatInfo”: Indica que l’algoritme té un o més “TeclatInfo”.

## 7.Alfabet

**Breu descripció de la classe:** La classe "Alfabet" és un conjunt de funcions que permeten analitzar i processar textos en diversos idiomes. Proporciona eines per identificar l'idioma del text, calcular la freqüència de lletres i paraules, així com per manipular i generar alfabetes específics per a cada idioma detectat.

**Cardinalitat:** Un per cada teclat.

**Descripció dels atributs:**

* idioma: String que guarda l'idioma detectat o establert per a l'anàlisi del text.
* caracters: String que representa els caràcters de l’alfabet de l'idioma detectat o establert.
* costLletres: Matriu de integer per emmagatzemar les adjacències de les lletres del text.
* frequenciaParaules: Mapa de string i integer que té la freqüència de paraules.
* frequenciaLletres: Mapa de character i integer que té la freqüència de les lletres.
* numAdjacències: Comptador integer del nombre total d'adjacències entre lletres.
* contadorLletres: Integer que té el recompte total de lletres.
* paraulesCat, paraulesCast, paraulesAng: Vector de strings amb paraules freqüents per a cada idioma.

**Descripció dels mètodes:**

* inicialitzarAlfabets(String text): aquest mètode inicialitza les propietats de la classe Alfabet basades en el text rebut. Detecta l'idioma, crea l'alfabet, calcula les freqüències de paraules i lletres, i determina els costos de les lletres.
* obtenirAlfabet(): retorna l'alfabet corresponent a l'idioma seleccionat. Si no el té, el program llança una excepció.
* detectarIdioma(String text): analitza un text per determinar l'idioma més probable basant-se en la freqüència de paraules clau en català, castellà i anglès. Si no el troba, envia una excepció especificant que requereix de més text per fer-ho.
* contador(String text, String[ ] paraules): compta quantes vegades apareixen les paraules contingudes en la llista de paraules dins del text donat.
* getParaulesIdioma(): retorna les paraules predefinides segons l'idioma emmagatzemat.
* calcularFrequenciaParaules(String text): compta la freqüència de cada paraula en el text rebut per paràmetre.
* calcularFrequenciaLletres(String text): compta la freqüència de cada lletra en el text rebut per paràmetre.
* calcularCost(String[ ] paraules): determina el cost de les lletres en base a les paraules rebudes.

**Descripció de les relacions:**

* Relació d’associació amb “Teclat”: indica que l’alfabet estarà a un teclat o més.

## 8.Distribució

**Breu descripció de la classe:** La classe “Distribució” representa una distribució específica de tecles en un teclat personalitzat. Es centra en la disposició de les tecles segons les preferències de l'usuari i utilitza algunes tècniques d'optimització per generar una disposició eficient.

**Cardinalitat:** Entre una i moltes distribucions per teclat.

**Descripció dels atributs:**

* nomTeclat: String que identifica el teclat al qual pertany la distribució de tecles.
* distribucio\_act: Matriu de chars que guarda la disposició actual de les tecles en el teclat.
* dataModificacio: Date que conté la data de quan es va modificar per última vegada la disposició de tecles.
* dimensio\_alt: Enter que representa l’altura desitjada del teclat per l’usuari.
* dimensio\_amp: Enter que representa l’amplada desitjada del teclat per l’usuari.
* estadístiques: Objecte Estadístiques que conté les estadístiques relacionades amb la distribució del teclat.
* algoritme: És l’algoritme escollit per l’usuari per crear una distribució del teclat, serà QAP o Hill Climbing.

**Descripció dels mètodes:**

* getEstadistiques(): retorna les estadístiques relacionades amb la distribució del teclat.
* getDistribucio\_act(): retorna la distribució actual del teclat.
* getNomTeclat(): retorna el nom identificador del teclat associat.
* setNomTeclat(String nomTeclat): estableix el identificador a la distribució associada per tal de saber a quin teclat correspon.
* getDimensio\_alt(): retorna l'alçada desitjada del teclat per l’usuari.
* getDimensio\_amp(): retorna l'amplada desitjada del teclat per l’usuari.
* getDataModificacio(): retorna la data de l'última modificació de la distribució.
* distribucioTeclatNova(int contadorLletres, int numAdjecencies, Map<String, Integer> frequenciaParaules, int alt, int amp, int[][] costLletres, Map<Character, Integer> lletraFreq, int x, int y, int numLletres, String alfabet, char ma): genera una nova distribució del teclat mitjançant l’algoritme Hill Climbing quan li passa els atributs necessaris per realitzar un teclat optimitzat.
* distribucioTeclatModificada(int alt, int amp, int[][] costLletres, Map<Character, Integer> lletraFreq, int x, int y, intnumLletres, String alfabet, char ma, Estadistiques e): modifica la distribució del teclat existent mitjançant l’algoritme Hill Climbing i torna a calcular les estadístiques, ja que per cada modificació de distribució, les estadístiques han de canviar també.
* distribucioTeclatNovaQAP(int alt, int amp, int[][] costLletres, String alfabet, char ma): genera una nova distribució del teclat mitjançant l’algoritme QAP quan li passa els atributs necessaris per realitzar un teclat optimitzat.
* distribucioTeclatModificada(int alt, int amp, int[][] costLletres, String alfabet, char ma, Estadistiques e): modifica la distribució del teclat existent mitjançant l’algoritme QAP i torna a calcular les estadístiques, ja que per cada modificació de distribució, les estadístiques han de canviar també.

**Descripció de les relacions:**

* Relació d’associació amb “Teclat”: indica que la distribució pertany a un teclat.
* Relació d’associació amb “Estadística”: indica que la distribució té una estadística.
* Relació d’associació amb “Algoritme”: indica que la distribució s’optimitza/modifica mitjançant un algoritme.

## 9.Estadística

**Breu descripció de la classe:** Aquesta classe conté i calcula les estadístiques d’una distribució d’un teclat. Les estadístiques són el percentatge d’ús de cada lletra, la distància mitjana que hi ha entre les lletres adjacents, el percentatge de lletres adjacents que es toquen al teclat i la distància mitjana utilitzada per teclejar les paraules freqüents.

**Cardinalitat:** Un per cada distribució.

**Descripció dels mètodes:**

* recalcularEstadistiques(char[][] teclat): recalcula les estadístiques donada la actualització de la distribució del teclat. Es crida a totes les funcions que calculen les diferents estadístiques i es guarden a la classe per poder consultar-la pròximament.
* dist(char c1, char c2, char[][] teclat): igual que a la classe algoritme, aquesta funció calcula la distància que hi ha entre les lletres c1 i c2 en la distribució que es passa com a paràmetre.
* calcDistMitjLletresAdjecents(char[][] teclat, String alfabet, int[][] costLletres, int nAdj): calcula la distància mitjana entre les lletres adjacents a l’input. Per fer el càlcul, es recorre la matriu de flux de lletres sencera i per cada parell de lletres es es crida a la funció dist perquè ens retorni les distàncies entre les dues tecles. Seguidament, es suma la distancia multiplicada pel flux de les lletres a la variable h que será la que al final de tot es retornarà, dividida entre nAdj.
* calcLletresAdjPercentatge(char[][] teclat, int[][] lCost, int length, string alfabet): Es recorre tota la matriu de flux i si el flux entre dues lletres és més que 0, i la distància entre les dues al teclat és més petit o igual a 0 (és a dir que es toquen o son la mateixa lletra), es suma un a nAdy. Mentrestant es calcula el número de parells de lletres que el seu flux és major a 0 a la variable n. Al final de tot es retorna nAdy dividit entre n i multiplicat per 100 per fer el percentatge. La funció acaba retornant el percentatge de lletres adjacents a l’input que en el teclat están una al costat de l’altra.
* calcDistMitjParaules(char[][] teclado, Map<String, Integer> pFreq): Calcula la distància mitjana que cal recórrer per teclejar una paraula de l’input. Per calcular el valor a retornar, es recorre el mapa de paraules, i per cada parell de lletres de la paraula, es crida a la funció dist perquè ens retorni la distància entre les dues lletres. Un cop calculada la distancia total per escriure la paraula, es suma a la variable dR multiplicada per la freqüència de la paraula. Al final de tot es retorna dR dividit entre el número de paraules que hi han per fer la mitjana.
* Estadistiques(String alfabet, char[][] teclat, Map<Character, Integer> lFreq, int[][] lCost, Map<String, Integer> pFreq, int nAdj, int nInput): Aquesta funció es la constructora de la classe i lo únic que fa és atribuir els valors que es passen com a paràmetres a les seves respectives variables locals de la classe i cridar a les funcions privades necessàries per completar totes les variables.

**Descripció dels atributs:**

* lletraFreq: Mapa de chars amb les lletres i enter del número de cops que apareix a les paraules freqüents o a la frase.
* distMitjanaLletresAdj: Double de la distància mitjana que hi ha entre les lletres adjacents del input.
* distMitjParaules: Double de la mitjana utilitzada per teclejar les paraules freqüents.
* lletresAdjPercentatge: Double de percentatge de lletres adjacents que es toquen al teclat.
* alfabet: L’objecte alfabet de la distribució.
* nInput: Integer del número de lletres que té l’input.

**Descripció de les relacions:**

* Relació d’associació amb la classe “Distribució” per indicar les estadístiques de la distribució.

## 10.TelcatInfo

**Breu descripció de la classe: “**teclatInfo” és una classe privada estàtica que emmagatzema informació relativa a una distribució de teclat. Conté una puntuació (heurística) i una matriu que representa la disposició del teclat.

**Cardinalitat:** Entre una i moltes distribucions per algoritme.

**Descripció dels atributs:**

* h: enter que guarda la puntuació o heurística associada a una distribució de teclat.
* teclat: matriu de chars que representa la disposició del teclat, on cada fila pot contenir una configuració de tecles.

**Descripció de les relacions:**

* Relació d’associació amb “Algoritme”: indica que el “TeclatInfo” pertany un algoritme.

# Descripció de la capa persistència

## 1.GestorCSV

**Breu descripció de la classe:** La classe GestorCSV s’encarrega de tota la persistencia relacionada amb els perfils i els teclats, és a dir, gestiona les dades dels teclats creats per cada perfil.

**Descripció dels atributs:**

* path**:** Variable string que conté el path de la carpeta del perfil actual.

**Descripció dels mètodes:**

* getPerfil(String user):Retorna el perfil amb el username corresponent si existeix.
* addPerfil(String user, Perfil tem\_perfil):Es crea una carpeta amb el nom d’usuari.
* removePerfil(String user):Es borra la carpeta del nom d’usuari.
* getTeclat(String user, String nomTeclat):Retorna el teclat de l'usuari amb el nom de teclat corresponent si existeix.
* addTeclat(String user, String nomTeclat, Teclat temp\_teclat):Dins del directori del perfil es crea un fitxer .txt amb l’informació del teclat que s’ha creat.
* removeTeclat(String user, String nomTeclat):Es borra el fitxer .txt del nom del teclat.
* existsTeclat(String user, String nomTeclat):Retorna si existeix el teclat amb el seu nom corresponent dins del directori del perfil.
* obtenirTeclats(String username):Retorna una llista de string amb els noms del teclat dins del directori del perfil.

## 2.CtrlPersistencia

**Breu descripció de la classe:** La classe CtrlPersistencia és el controlador de la capa persistència que comunica amb la capa domini i els mètodes de les classes de la persistència.

**Descripció dels mètodes:** Accés a tots els mètodes anteriorment mencionats.

# Descripció de capa presentació

## 1.VistaPrincipal

**Breu descripció de la classe:** La VistaPrincipal s’encarrega de fer el registrament dels usuaris i de l’inici de sessió. Quan es vulgui enregistrar un usuari al sistema, es donarà un espai per posar el username i el password i també un desplegable per triar si la mà predominant és la dreta o l'esquerra. Per iniciar sessió, hi haurà un espai per posar l’usuari i el password. Si l’enregistrament es fa incorrectament o l’usuari i el password no coincideixen, saltarà una excepció.

**Descripció dels atributs:**

* framePrincipal: JFrame que representa la vista principal.
* panelsign Panell de la que conté el JComboBox per a crear un nou perfil.
* registrarPerfilBtn JButton per a registrar un perfil.
* IniciarSesioPerfilBtn: JButton per a iniciar sesió.
* exitBtn: Jbutton per a sortir de l'aplicació.
* userLabel: JLabel que mostra “User”.
* passwordLabel : JLabel que mostra “Password”.
* signLog JLabel que indica si s’està iniciant sessió o registrant un perfil.
* User: JTextArea on l’usuari introduira el seu identificador d’usuari.
* Password: JTextArea on l’usuari introduirà la seva contrasenya.
* comboma: JComboBox per a poder elegir quina mà es vol fer servir.
* registrar: Boolean que indica si actualment s’està registrant un perfil o s’està iniciant sesió.
* ctlpresenta: instància del control de presentació.

**Descripció dels mètodes:**

* errorPopUp(): Crea un nou panell amb que mostrarà un text amb la exepció que s’hagi donat.
* creatBtn(): Inicialitza tots els JButton de la vista.
* crearLb(): Inicialitza tots els JLabels de la vista.
* addtoFrame(): Afegeix tots els elements al frame.
* createTa(): Inicialitza tots els JTextArea de la vista.
* createCombo: Inicialitza el JComboBox de la mà.
* panelSignup(): crea el Jpanel que amb elements necessaris per a registrar el perfil.

## 2.VistaPerfil

**Breu descripció de la classe:** La Vista Perfil s’encarrega de mostrar l’usuari les diverses comandes que pot executar ja sigui a nivell de perfil o a nivell de teclats. A nivell de perfil podrà modificar-lo o eliminar-lo, tancar sessió o sortir de l’aplicació. Per altra banda, a nivell de teclat podrà crear-ne, llistar els que ha creat i consultar les estadístiques d’un.

**Descripció dels atributs:**

* modificarPerfilBtn JButton per a modificar el perfil.
* eliminarPerfilBtn: JButton per eliminar el perfil.
* tancarSessioBtn: JButton per a tancar la sesió.
* exitBtn: Jbutton per a sortir de l'aplicació.
* crearTeclatBtn: JButton per a crear un teclat.
* llistarTeclatsBtn: JButton per a llistar els teclats de l’usuari.
* consultarEstadisticasBtn: JButton per a consultar les estadístiques d’un teclat.
* opcionsLabel: JLabel que mostra “Opcions”.
* perfilLabel: JLabel que mostra “Comandes perfil:”.
* teclatsLabel: JLabel que mostra “Comandes teclat:”.
* ctlpresenta: instància del control de presentació.

**Descripció dels mètodes:**

* creatBtn(): Inicialitza tots els JButton de la vista.
* crearLb(): Inicialitza tots els JLabels de la vista.
* addtoFrame(): Afegeix tots els elements al frame.

## 3.VistaCrearTeclat

**Breu descripció de la classe:** La VistaCrearTeclat s’encarrega de fer la creació del teclat. S’indica clarament camps de text per indicar el nom del teclat i el input i un camp visual/desplegable per escollir la posició a l’eix X i Y on l’usuari prefereix tenir el polze. Aquest input podrà ser el path del fitxer txt, una llista de paraules amb la freqüència amb la que s’usen o un desplegable per escollir l’idioma. Si la creació es fa incorrectament o el nom del teclat ja existeix, saltarà una excepció.

**Descripció dels atributs:**

* TitolCrearTeclat: JLabel amb el títol de la vista VistaCrearTeclat.
* nomTeclatL: JLabel amb el text “Nom del teclat”.
* frameCrearTeclat: JFrame de la vista VistaCrearTeclat.
* nomTeclatTA: JTextArea on l’usuari intoduirà el nom del teclat que vol crear.
* inputCrear: JComboBox amb es imputs des dels que es vol crear el teclat.
* panelPath: JPanel amb tots els objectes relacionats amb el path.
* pathLabel: JLabel amb el text “Path del text”.
* pathText: JTextArea on l’usuari introduirà el path del seu text.
* panelSeleccionarIdioma: JPanel amb tots els objectes relacionats amb seleccionar l’idioma.
* idiomaLabel: JLabel amb el text “Idioma del teclat”.
* idiomaChoice: JComboBox amb els idiomes que l’usuari pot seleccionar pel seu teclat.
* panelParaulesFreq: JPanel amb tots els objectes relacionats amb les paraules freqüents.
* paraulesFreqLabel: JLabel amb el text “Intodueix paraules frequents”.
* paraulesFreqTextArea: JTextArea on l’usuari introduirà les paraules freqüents.
* posPolXL: JLabel amb el text “Posicio del polze en X”.
* posPolYL: JLabel amb el text “Posicio del polze en Y”.
* posPolXTA: JTextArea on l’usuari introduirà la seva posició del polze en x.
* posPolYTA: JTextArea on l’usuari introduirà la seva posició del polze en y.
* ampladaTeclatL: JLabel amb el text “Amplada del teclat”.
* alturaTeclatL: JLabel amb el text “Altura del teclat”.
* alturaTeclatTA: JTextArea on l’usuari introduirà la altura del seu teclat.
* ampladaTeclatTA: JTextArea on l’usuari introduirà l’amplada del seu teclat.
* algoritmeLabel: JLabel amb el text “Algoritme per crear teclat”.
* algoritmeChoice: JComboBox amb els algoritmes que l’usuari pot seleccionar pel seu teclat.
* crearBtn: JButton que crearà el teclat.

## 4.VistaLlistarTeclats

**Breu descripció de la classe:** La VistaLlistarTeclats s’encarrega de llistar els teclats associats a un usuari específic.

**Descripció dels atributs:**

* VistaLlistarTeclats: JLabel amb el títol de VistaLListarTeclats.
* frameLlistarTeclats: JFrame de la vista VistaLlistarTeclats.
* comboTeclats:JCombo amb tots els teclats.
* ConsultarTeclatsBtn: JButton per a consultar el teclat seleccionat a comboTeclats.

## 5.VistaConsultarEstadistiques

**Breu descripció de la classe:** La classe VistaConsultarEstadistiques s’encarrega de mostrar les estadístiques d’un teclat on es mostra la distància mitjana que hi ha entre les lletres adjacents, el percentatge de lletres adjacents que es toquen al teclat i la distància mitjana utilitzada per teclejar les paraules freqüents.

**Descripció dels atributs:**

* TitolEstadistiques: JLabel que mostra “Mostrar estadistiques d’un teclat”.
* NomTeclat: JLabel que mostra el nom del teclat seleccionat.
* DMLletres: JLabel que mostra les estadístiques de la distància mitja entre lletres.
* DMParaules: JLabel que mostra les estadístiques de la distància mitja entre les paraules.
* PercLletresAdjacents: JLabel que mostra el percentatge de lletres adjacents.
* TeclatVisualitzar: JTextArea on l’usuari introduirà el nom del teclat del que vol visualitzar les estadístiques.
* frameConsultarEstadistiques: JFrame de la vista VistaConsultarEstadistiques.
* panelEstadistiques: JPanel que on hi haurà tota la informació del teclat al que se l’hi estan consultant les estadístiques.
* panelSelecciona: JPanel on hi haurà tot el necessari per a poder seleccionar el teclat del que es volen consultar les estadístiques.

## 6.VistaConsultarTeclat

**Breu descripció de la classe:** La classe VistaConsultarTeclar mostra un teclat i ofereix les opcions de borrar-lo o de modificar-ne la distribució.

**Descripció dels atributs:**

* TitolConsutarTeclat: JLabel amb el títol de VistaConsultarTeclat.
* frameConsultarTeclat: JFrame de la vista VistaConsultarTeclat.
* Teclat: JLabel amb la distribució del teclat.
* EliminarTeclat: JButton per a borrar el teclat.
* ModificarTeclat: JButton per a modificar el teclat.

## 7.VistaModificarPerfil

**Breu descripció de la classe:** VistaModificarPerfil s’encarrega de modificar els atributs maDominant i Password del perfil d’usuari per uns atributs nous.

**Descripció dels atributs:**

* modificar: JButton per a modificar un perfil.
* ctlpresenta:instancia del CtlPresentacio.
* frameModificarPerfil: JFrame que representa la VistaModificarPerfil.
* atriutModificar: JLabel que mostra “Atribut a modificar:”.
* novaDominant: JLabel que mostra “Nova ma dominant”.
* novaContra: JLabel que mostra “Nova contrasenya”.
* comboma: JComboBox per a poder elegir quina mà es vol fer servir.
* Password: JTextArea on l’usuari introduirà la seva nova contrasenya.

## 8.VistaModificarTeclat

**Breu descripció de la classe:** La classe VistaModficarTeclat modifica la distribució del teclat passantli una nova altura i amplada del teclat que siguin compatibles, és a dir, que el producte de de les files i columnes és menor que el nombre de caràcters.

**Descripció dels atributs:**

* TitolModificar: JLabel amb el títol de la vista VistaModificarTeclat.
* NovaColumnaL: JLabel amb “Nou nombre de columnes del teclat”.
* NovaFila: JLabel amb “Nou nombre de files del teclat”.
* nouAlgoritmeL: JLabel amb “Seleccionar nou algoritme per crear el teclat”.
* nouAlgoritme: JComboBox amb els algoritmes que l’usuari pot seleccionar per modificar el seu teclat.
* frameModificarTeclat: JFrame de la vista VistaModificarTeclat.
* NovaColumnaTA : JTextArea on l’usuari intoduirà el nou nombre de columnes que vol.
* NovaFilaTA: JTextArea on l’usuari intoduirà el nou nombre de files que vol.
* modificarTeclatBtn: JButton per a modificar el teclat.

## 9.CtrlPresentacio

**Breu descripció de la classe:** La classe CtrlPresentacio actua com a controlador de la capa de presentació en la nostra aplicació. S’encarrega de gestionar la interacció entre les vistes de l’interfaç de l’usuari i la capa del domini.

**Descripció dels atributs:**

* ControladorDomini (tipus: CtrlDomini): Un atributo estático que representa una instancia del controlador de dominio CtrlDomini. Permet la comunicació entre la capa de presentació i la capa de domini.

**Descripció dels mètodes:**

* exec(): Inicialitza el control presentació i crida a la VistaPrincipal.
* signupPerfil(String user, String password, Perfil.ma ma): crea un nou perfil amb els atributs que ha introduit l’usuari i crida a la VistaPerfil i tanca la VistaPrincipal.
* logginPerfil(String user, String password): inicia sessió amb el user i contrasenya password que ha introduït l’usuari i crida a la VistaPerfil i tanca la VistaPrincipal.
* modificarPerfil(String user, String password, Perfil.ma ma): modifica els atributs del Perfil amb nom user que desitja l’usuari i tanca la VistaModificarPerfil.
* eliminarPerfil(String user): s’elimina el perfil amb nom user i tanca VistaPerfil i crida a VistaPrincipal.
* tancarSessio(): es tanca la sessió de l'usuari i tanca VistaPerfil i crida a VistaPrincipal.
* crearTeclat(String username, String nomTeclat, Perfil.ma ma, int pulgar\_x, intpulgar\_y, int altura\_inicial, int amplada\_inicial, String idioma, String palabrasF, String text): crea un teclat nou amb els paràmetres especificats i realitza validacions d'entrada. Tanca VistaCrearTeclat.
* obtenirEstadistiquesUltimaDistribucio(String username, String nomTeclat): obté estadístiques de la darrera distribució d'un teclat ja existent de l’usuari. Tanca VistaEstadistiquesTeclat.
* modificarTeclat(String username, String nomTeclat, int nova\_altura, int nova\_amplada): modifica la distribució d'un teclat ja existent de l’usuari canviant-ne l'alçada i l'amplada. Tanca VistaModificarTeclat, VistaConsultarTeclat i VistaLlistarTeclats.
* eliminar\_teclat(String username, String nomTeclat): elimina el teclat de l’usuari. Tanca VistaConsultarTeclat i VistaLlistarTeclats.
* getTeclat(String username, String nomTeclat): retorna un teclat ja existent i identificat per a un usuari i nomTeclat determinat.
* obtenirTeclatsUsuari(String username): recupera una llista de teclats associats a l’usuari.
* consultarUltimaDistribucio(String username, String nomTeclat): retorna l'última distribució d'un teclat nomTeclat específic que pertany a un usuari. Tanca VistaConsultarTeclat i VistaLlistarTeclats
* exit(): es tanca el programa i totes les vistes.