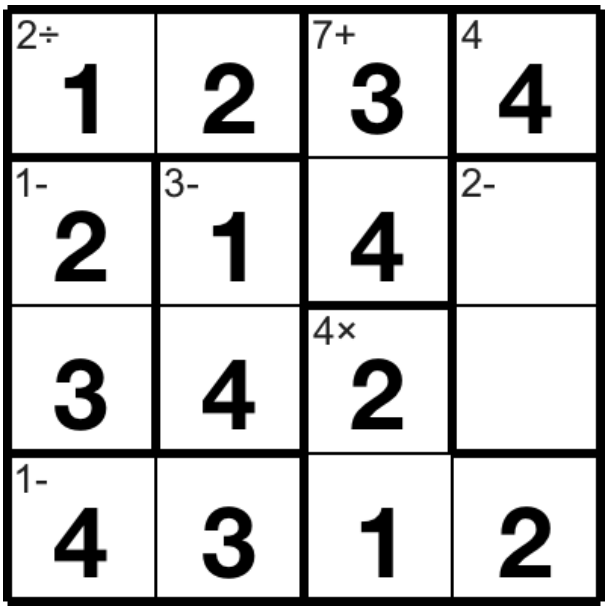
UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

KENKEN - PROP



GERARD FERRER ORIOL

ALBERT BETORZ LÓPEZ

SERGI PLA CASAMITJANA

BART SEBASTIAAN ZANEN

Grup: 12.3

Entrega 1 / Projectes de Programació

Tutor: CARLES ARNAL CASTELLO (Departament de Ciències de la Computació)

Grau: Grau en Enginyeria Informàtica (Computer Engineering)

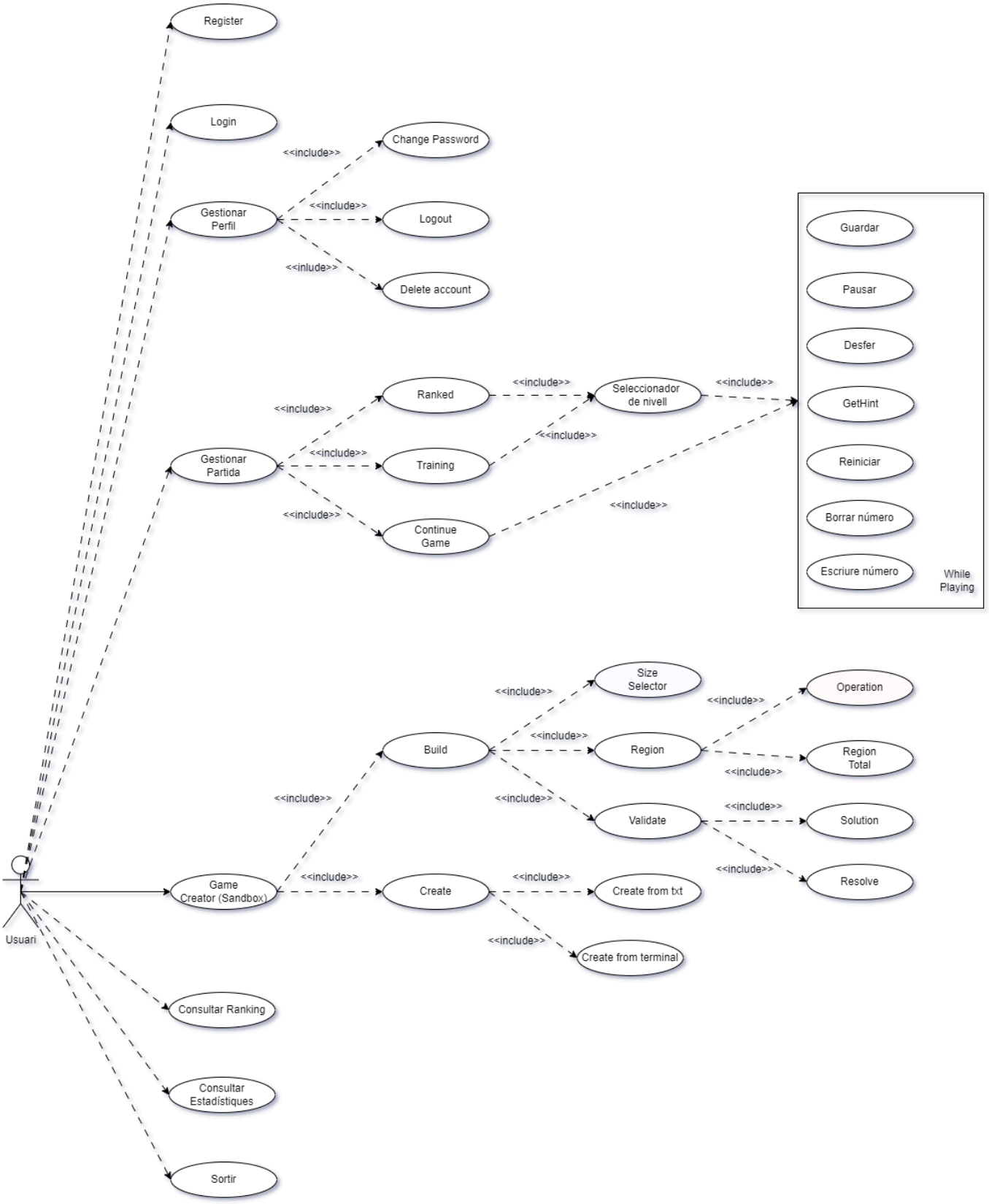
Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

ÍNDEX

1. DIAGRAMA DE CASOS D'ÚS

1.1 Diagrama



1.2 Descripció dels casos

Nom: UC001 - Register

Actor: Usuari

Precondició: L'usuari no està registrat.

Activador: L'usuari indica al sistema que vol crear un perfil.

Escenari principal:

- El sistema mostra per pantalla un formulari per introduir el nom d'usuari: [*username*], la contrasenya: [*password*] i la confirmació de la contrasenya: [*repeat password*].
- L'usuari omple els camps i prem el botó "Create".
- El sistema crea un nou perfil i guarda les dades d'aquest.

Errors possibles i camins alternatius:

- Si ja existeix un usuari [*username]*, el sistema informa amb l'error: "El nom d'usuari escollit ja existeix"
- Si la contrasenya [*password*] i la confirmació de la contrasenya [*repeat password*] no coincideixen, el sistema informa amb l'error: "Les contrasenyes no coincideixen".

Nom: UC002 - Login

Actor: Usuari

Precondició: L'usuari està registrat i no loguejat.

Activador: L'usuari indica al sistema que vol carregar un perfil.

Escenari principal:

- El sistema mostra per pantalla un formulari per introduir el nom d'usuari: [*username*] i la contrasenya: [*password*].
- L'usuari omple els camps i prem el botó "Login".
- El sistema valida els valors i carrega el perfil.

Errors possibles i camins alternatius:

- Si no existeix cap usuari amb el nom d'usuari [*username*] i la contrasenya [*password*], el sistema informa amb l'error: "Nom d'usuari o contrasenya incorrectes".

Nom: UC003 - Gestionar Perfil

Actor: Usuari

Precondició: L'usuari està registrat i loguejat.

Activador: L'usuari indica al sistema que vol gestionar el perfil.

Escenari principal:

- El sistema mostra per pantalla les opcions de gestió del perfil que té.

Errors possibles i camins alternatius: -

Nom: UC004 - Change password

Actor: Usuari

Precondició: L'usuari està registrat i loguejat

Activador: L'usuari indica al sistema que vol canviar la contrasenya

Comportament:

- El sistema mostra per pantalla un formulari per introduir la contrasenya [*current password*] del perfil del qual es vol canviar la contrasenya, la nova contrasenya [*new password*] i la confirmació [*repeat password*] d'aquesta nova contrasenya.
- L'usuari omple els camps i prem el botó "Change".
- El sistema guarda la nova contrasenya i elimina la contrasenya antiga.

Errors possibles i camins alternatius:

- Si la contrasenya actual del perfil [*current password*] és errònia, el sistema informa amb l'error: "La contrasenya actual introduïda no és correcte".
- Si la contrasenya [*new password*] i la confirmació de la contrasenya [*repeat password*] no coincideixen, el sistema informa amb l'error: "Les contrasenyes no coincideixen".

Nom: UC005 - Logout

Actor: Usuari

Precondició: L'usuari està registrat i loguejat.

Activador: L'usuari indica al sistema que vol tancar la sessió.

Escenari principal:

- El sistema mostra per pantalla dos botons, un botó "Logout" i un botó "Cancel".
- Si l'usuari prem el botó "Cancel", l'usuari seguirà loguejat, en canvi, si es prem el botó "Logout", l'usuari deixarà d'estar loguejat i veurà la pantalla inicial.

Errors possibles i camins alternatius: -

Nom: UC006 - Delete user

Actor: Usuari

Precondició: L'usuari està registrat i loguejat.

Activador: L'usuari indica al sistema que vol esborrar el perfil.

Escenari principal:

- El sistema mostra per pantalla un formulari per introduir la contrasenya del perfil que es vol esborrar.
- L'usuari introdueix la contrasenya i prem el botó "Delete". El sistema comprova que la contrasenya sigui vàlida per a aquell usuari.
- El sistema mostra un missatge de confirmació: "Are you sure you want to delete [*username]?"* amb dos botons, un botó "Yes" i un botó "No".
- Si l'usuari prem el botó "No", l'usuari seguirà registrat i loguejat, però en cas que es premi el botó "Yes", l'usuari deixarà d'estar registrat en el sistema.

Errors possibles i camins alternatius:

- Si la contrasenya [*password*] no és vàlida per aquell nom d'usuari [*username*], el sistema informa amb l'error: "La contrasenya introduïda no és correcta".

Nom: UC007 - Gestionar partida

Actor: Usuari

Precondició: L'usuari està registrat i loguejat.

Activador: L'usuari indica al sistema que vol gestionar una partida.

Comportament:

- El sistema mostra les opcions de gestió de partida que té

Errors possibles i camins alternatius: -

Nom: UC008 - Ranked

Actor: Usuari

Precondició: L'usuari està registrat i loguejat.

Activador: L'usuari indica que vol jugar una partida del mode ranked. Aquest mode és el que conta per les puntuacions i és sobre els nivells predeterminats.

Comportament:

- El sistema mostra els taulers disponibles del mode ranked, a l'usuari.

Errors possibles i camins alternatius: -

Nom: UC009 - Continue Game

Actor: Usuari

Precondició: L'usuari està registrat i loguejat

Activador: L'usuari indica que vol continuar una partida ja començada.

Comportament:

- El sistema mostra els taulers disponibles que l'usuari ha començat en algun moment però no estan acabats.
- L'usuari tria un dels taulers disponibles per continuar la partida.

Errors possibles i camins alternatius:

- Si no hi ha cap partida guardada, el sistema informa amb l'error: "No hi ha partides disponibles".

Nom: UC010 - Level Selector

Actor: Usuari

Precondició: L'usuari està registrat i loguejat

Activador: L'usuari ha indicat que vol jugar una partida de tipus ranked o training.

Comportament:

- El sistema mostra els nivells disponibles per cada un dels modes de joc.
- L'usuari tria un dels nivells disponibles per tal de seleccionar un tauler.

Errors possibles i camins alternatius: -

Nom: UC011 - Desfer

Actor: Usuari

Precondició: L'usuari està registrat i loguejat, a més està jugant una partida de qualsevol tipus.

Activador: L'usuari indica que vol desfer l'última jugada que ha realitzat.

Comportament:

- El sistema desfà l'última acció feta per l'usuari.
- L'usuari torna a veure el tauler tal i com estava just abans de la última operació feta.
- L'acció en partides de tipus ranked té un impacte en la puntuació obtinguda al completar el tauler.

Errors possibles i camins alternatius:

- En el cas en que no s'hagi fet cap jugada anterior, el sistema no farà res.

Nom: UC012 - Pausar

Actor: Usuari

Precondició: L'usuari està registrat i loguejat, a més, ha triat que vol jugar una partida, ja sigui de rànquing o d'entrenament.

Activador: L'usuari indica que vol pausar la partida.

Comportament:

- El sistema pausa la partida, i en conseqüència, atura el temps.

Errors possibles i camins alternatius: -

Nom: UC013 - Reiniciar

Actor: Usuari

Precondició: L'usuari està registrat i loguejat, a més, ha triat que vol jugar una partida, ja sigui de rànquing o d'entrenament.

Activador: L'usuari indica que vol reiniciar la partida.

Comportament:

- El sistema atura el temps i mostra per pantalla una confirmació de que es vol reiniciar la partida: "Do you want to restart the game? Yes/No".
- Si l'usuari prem el botó "No", l'usuari seguirà jugant la partida que havia començat i el temps seguirà corrent, però en cas que es premi el botó "Yes", l'usuari veurà el tauler tal i com estava al principi i es reiniciarà el temps.

Errors possibles i camins alternatius: -

Nom: UC014 - Escriure número

Actor: Usuari

Precondició: L'usuari està registrat, loguejat i jugant en una partida seleccionada.

Activador: L'usuari indica que vol escriure un número en una cel·la

Comportament:

- El sistema introdueix a la cel·la el nombre indicat per l'usuari.

Errors possibles i camins alternatius:

- Si el número introduït no compleix amb les regles del joc, el sistema canvia el color dels números afectats a vermell.
- Si ja hi ha un número en la cel·la, el sistema cambia el número per el nou introduït.

Nom: UC015 - Borrar número

Actor: Usuari

Precondició: L'usuari està registrat, loguejat i jugant en una partida seleccionada.

Activador: L'usuari indica que vol borrar el número de la cel·la seleccionada

Comportament:

- El sistema borra el número de la cel·la seleccionada.

Errors possibles i camins alternatius:

- En el cas en que a la casella seleccionada no hi hagi cap número, el sistema no farà res.

Nom: UC016 - Guardar

Actor: Usuari

Precondició: L'usuari està registrat, loguejat i jugant en una partida seleccionada.

Activador: L'usuari indica que vol guardar la partida actual.

Comportament:

- El sistema atura el temps i mostra per pantalla una confirmació de que es vol guardar la partida: "Do you want to save the current game? Yes/No".
- Si l'usuari prem el botó "No", l'usuari seguirà jugant la partida que havia començat i el temps seguirà corrent, però en cas que es premi el botó "Yes", el tauler es guardarà tal i com estava en el moment de prémer el botó de guardar (amb un temps i nombre de números escrits determinat). S'afegeix al vector savedGames.

Errors possibles i camins alternatius: -

Nom: UC017 - Game Creator (Sandbox)

Actor: Usuari

Precondició: L'usuari està registrat i loguejat.

Activador: L'usuari indica que vol crear un kenken (mode Game Creator)

Comportament:

- El sistema mostra les diferents opcions de creació de kenkens disponibles.

Errors possibles i camins alternatius: -

Nom: UC018 - Consultar ranking

Actor: Usuari

Precondició: L'usuari ha d'estar registrat i loguejat.

Activador: L'usuari indica que vol consultar el rànquing.

Comportament:

- El sistema mostra les opcions de rànquing que té.
- L'usuari escull una d'aquestes opcions.
- El sistema mostra el rànquing de l'opció demanada per l'usuari.

Opcions de rànquing:

- 1. ordenat per punts
- 2. ordenat per nom
- 3. ordenat per temps
- 4. ordenat per dificultat
- 5. ordenat per nº de kenkens solucionats

Errors possibles i camins alternatius: -

Nom: UC019 - Create

Actor: Usuari

Precondició: L'usuari està registrat i loguejat.

Activador: L'usuari indica que vol pujar un tauler creat.

Comportament: El sistema mostrarà les dues opcions disponibles per pujar taulers fets per l'usuari.

Errors possibles i camins alternatius: -

Nom: UC020 - Create from text

Actor: Usuari

Precondició: L'usuari està registrat i loguejat.

Activador: L'usuari indica que vol introduir un nou tauler en format txt.

Comportament:

- L'usuari introdueix el path del fitxer de text.
- En cas que tingui solució, es mostraran les opcions de jugar o de mostrar-lo solucionat.

Errors possibles i camins alternatius:

- En cas que el tauler no tingui solució es mostrarà el missatge: "El tauler no té solució".

Nom: UC021 - Create from terminal

Actor: Usuari

Precondició: L'usuari està registrat i loguejat.

Activador: L'usuari indica que vol introduir un nou tauler mitjançant la terminal.

Comportament:

- Comença l'algorisme d'entrada que llegeix els valors introduïts per l'usuari.
- Es demana la mida del kenken, el número de regions i es procedeixen a llegir tots els valors.

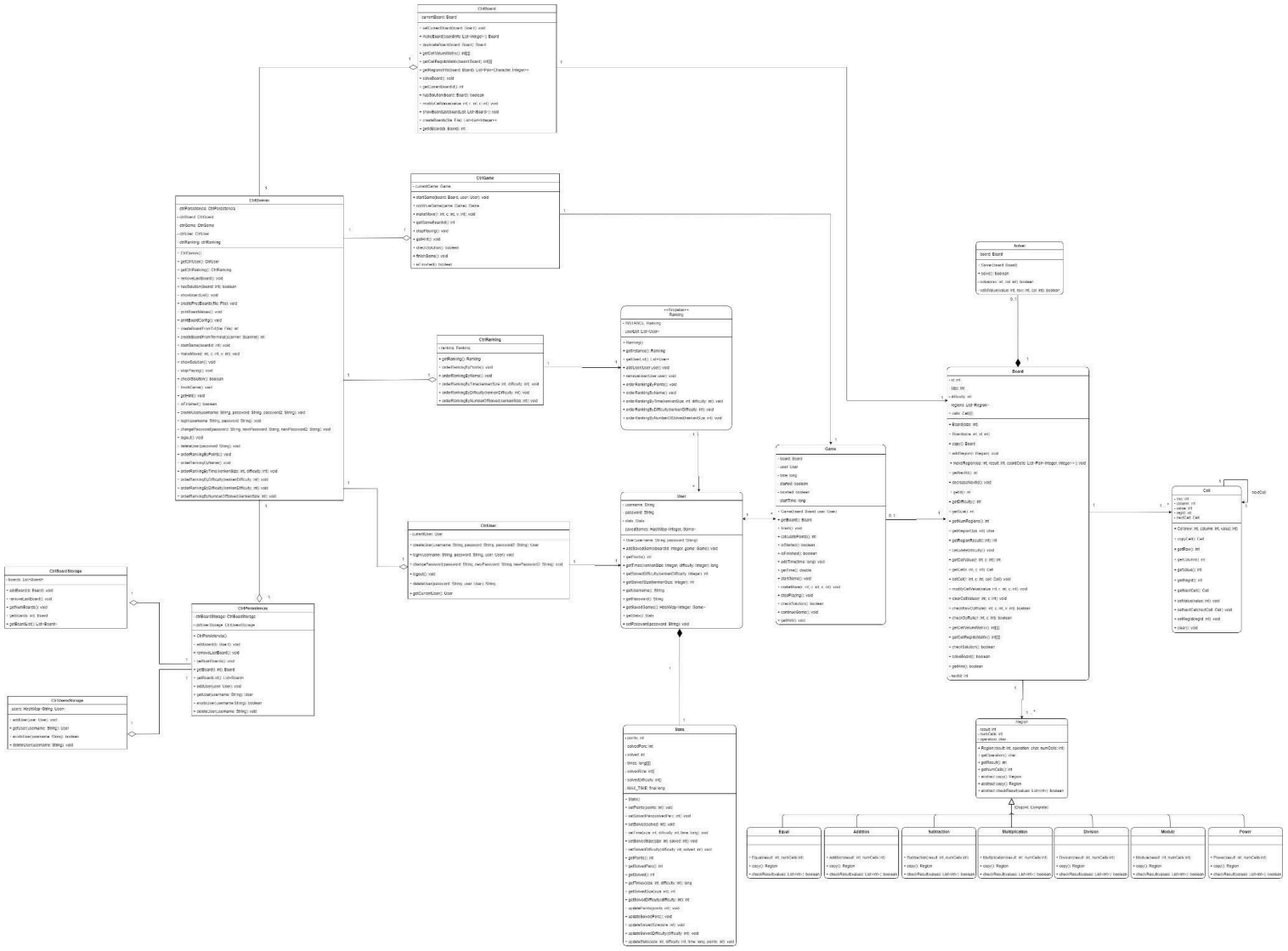
Errors possibles i camins alternatius:

- Si la mida del tauler és incorrecte es mostra el missatge: "No correct size".
- Si el nombre de regions és 0, negatiu o superior a N*N sent N la mida del tauler, es mostra el missatge: "No correct number of regions".

Els casos d'ús no mostrats son els pendents a implementar i per tant, seran comentats en posteriors entregues.

2. DIAGRAMA DEL MODEL CONCEPTUAL DE DADES

2.1 Diagrama



2.2 Descripció de les classes (atributs i mètodes)

2.2.1. Cell

La classe **Cell** representa una cel·la individual en el tauler del joc. Cada cel·la té una fila, "*row*", i una columna, "*column*", per saber la seva posició dins del tauler, un valor, "*value*", per guardar el valor actual de la cel·la, un identificador de la regió, "*regId*", per saber a quina regió pertany, i una cel·la, "*nextCell*", per saber quina és la següent cel·la dins de la regió a la qual pertany.

Entre les seves funcionalitats, aquesta classe ofereix un mètode per fer una còpia d'ella mateixa, "*copyCell*", per tal de poder fer còpies de taulers. El mètode *"clear"* esborra el valor actual de la cel·la.

2.2.2 Region

La classe abstracta **Region** representa una regió en el tauler del joc, amb atributs "*result*" per emmagatzemar el resultat, "*operation*" per identificar el símbol de l'operació (=, +, -, *, /, %, ^), i el nombre de cel·les, "*numCells*", de la regió.

Consta de dos mètodes abstractes, "*copy*" per crear una còpia de la regió i "*checkResult*" per verificar si un llistat de valors compleix l'operació esperada de la regió.

Hi ha 7 subclasses (**Equal**, **Addition**, **Subtraction**, **Multiplication**, **Division**, **Module** i **Power**) per poder representar totes les operacions que ofereixen els taulers del nostre joc KenKen.

2.2.3 Board

La classe **Board** és el centre del joc KenKen, amb tota la lògica i estructura del tauler. Aquesta classe consta d'un identificador, "*id*", per diferenciar els taulers, una mida, "*size*", que determina les dimensions del tauler, un nivell de dificultat, "*difficulty*", per saber la dificultat del tauler (easy, medium, hard o expert), un llistat de regions, "*regions*", per saber totes les regions en les que està dividit el tauler, i una matriu de cel·les, "*cells*", que conté totes les cel·les del tauler.

Els constructors inicialitzen el tauler amb cel·les buides, per després acabar de definir les cel·les i les regions mitjançant els mètodes "*addRegion*" i "*makeRegion*". També ofereix una funció per calcular i assignar la dificultat del tauler un cop creat, i un mètode per fer una còpia del tauler.

Per tal de verificar si es compleixen les regles del joc KenKen, la classe board ofereix dos mètodes, "*checkRowColRule*" i "*checkOpRule*", i també té un mètode, "*checkResult*" per comprovar si l'estat actual del tauler és una solució correcta.

Además interacciona amb la classe Solver per proporcionar la solució del tauler mitjançant el mètode "*solveBoard*", o obtenir una pista amb "*getHint*".

2.2.4 Pair

La classe **Pair<X, Y>** és una estructura de dades genèrica que permet emmagatzemar i utilitzar un parell de valors o objectes, on "*X*" i "*Y*" representen els tipus de dades del parell d'elements.

2.2.5 Solver

La classe Solver és l'encarregada de resoldre taulers del joc KenKen, mantenin una referència al tauler "*board*" que es vol resoldre. A través del constructor s'assigna el tauler que es vol solucionar.

El mètode principal d'aquesta classe, "*solve*", inicia el procés de solució recorrent totes les cel·les del tauler intentant assignar-los un valor vàlid segons les regles del joc. La recursivitat junt amb el backtracking permeten explorar totes les possibles combinacions de valors per tal de trobar una solució, si hi ha.

2.2.6 Game

La classe **Game** conté tota la informació i lògica necessària per gestionar una partida de KenKen. Aquesta classe vincula un tauler "*board*" amb un usuari "*user*" per fer el seguiment de la partida.

S'inicia la partida amb el mètode "*startGame*", que enregistra el temps d'inici de la partida. Ofereix diferents accions un cop començada la partida, fer un moviment, "*makeMove*", per introduir un valor a una cel·la específica, pausar la partida, "*stopPlaying*", per guardar la partida en l'estat actual per tal de continuar-la en un altre moment si es desitja amb el mètode "*continueGame*", i obtenir una pista, "'getHint", que introdueix un valor de la solució d'una cel·la. El mètode 'finish' permet determinar que la partida s'ha acabat, i s'encarrega de calcular els punts amb "*calculatePoints*" i proporcionar tots els valors necessaris per mantenir les estadístiques de l'usuari que ha jugat la partida.

2.2.7 Ranking

La classe **Ranking** és un singleton, per tant, és una única instància dins de tota l'aplicació que representa el ranking dels usuaris registrats al sistema. L'estructura d'aquesta classe és una llista d'usuaris, denominada "*userList*".

Els mètodes a destacar de la classe Ranking són els que ens permeten ordenar la classificació d'usuaris segons diferents criteris. El mètode "*orderRankingByPoints*()" ordena la llista pels punts acumulats pels usuaris, "*orderRankingByName*()" l'ordena

per ordre alfabètic dels noms dels usuaris, "*orderRankingByTime(int kenkenSize, int difficulty)*" l'ordena pel temps de resolució d'un KenKen en específic (identificat per mida i dificultat), i finalment "*orderRankingByDifficulty(int kenkenDifficulty)*" que ordena el ranking pel nombre de KenKens resolts en una determinada dificultat.

2.2.8 Stats

Aquesta classe és l'encarregada de representar les estadístiques de cada usuari. Els valors enregistrats a les estadístiques son: el número de punts que s'han aconseguit jugant els KenKens predeterminats "*points*" i el número de KenKens solucionats "*solved*". A més, hi ha dos vectors "*solvedSize[]*" i "*solvedDifficulty[]*" per tenir un recompte més concret de quins nivells ha resolt l'usuari separats per la mida i la dificultat respectivament. Els temps en els que s'ha resolt cada KenKen estan guardats a la matriu "*times[size][difficulty]*". Per últim hi ha "*solvedPerc*" que representa el % de KenKens predeterminats que l'usuari hagi solucionat. Quan un usuari hagi completat una partida es cridarà a "*updateStats(size, difficulty, time, points)*". Se li passarà la mida, dificultat, temps i punts obtinguts del KenKen resolt i s'actualitzaran totes les estadístiques correctament.

2.2.9 User

La classe **User** és la que representa els usuaris en el sistema. Cada usuari té un "*username*" que l'identifica i un "password" per iniciar sessió. També conté l'atribut "*stats*" de tipus Stats on hi guardem les estadístiques de l'usuari i un HashMap<Integer, Game> "*savedGames*" que conté les partides guardades per l'usuari.

Els mètodes de la classe ens permeten realitzar diverses operacions sobre l'usuari. Amb "*changeUsername()*" i "*changePassword()*" podem canviar el nom d'usuari i la contrasenya. Amb *getPoints()*, *getTimes(size, difficulty)*, *getSolvedDifficulty(difficulty)* i *getSolvedSize(size, solved)* podem accedir a les estadístiques de l'usuari per obtenir els següents valors: punts aconseguits, temps en resoldre un determinat KenKen (identificat per mida i dificultat), el número de KenKens solucionats d'una determinada dificultat i el número de KenKens solucionats d'una certa mida.

2.2.10 CtrlBoard

El **controlador de Board** gestiona els taulers "*boards*" en el joc, encarregant-se de manipular el tauler actual, crear nous taulers "*createBoards(file)*", duplicar taulers "*duplicateBoard(board)*", i també solucionar-los "*solveBoard()*". El controlador té un atribut "*currentBoard*" per tal de saber sobre quin tauler s'han d'aplicar els diferents mètodes de la classe.

2.2.11 CtrlGame

El **controlador de Game** gestiona les partides "*game*" dins del joc KenKen permetent iniciar "*startGame(board, user)*", continuar "*continueGame(game)*", pausar "*stopPlaying*()", fer moviments "*makeMove(r, c, v)"*, comprovar solucions "*checkSolution()*", obtenir pistes "*getHint()*" i finalitzar partides "*finishGame()*". També té la partida actual, "*currentGame*", permetent la manipulació d'aquesta partida.

2.2.12 CtrlRanking

El **controlador de Ranking**, a partir de la instància única de "*Ranking*", gestiona i ordena el rànquing de jugadors del joc KenKen segons diverses opcions com

- punts "*orderRankingByPoints()*",
- nom "*ordenarRankingByName()*",
- temps "*ordenarRankingByTime(kenkenSize,difficulty)*",
- dificultat "*orderRankingByTime(kenkenDifficulty)*"
- nombre de KenKens resolts "*orderRankingByNumberOfSolved(kenkenSize)"*.

2.2.13 CtrlUser

El **controlador de User** gestiona les accions dels usuaris en el joc KenKen. Ofereix la creació d'usuaris "*createUser(username, password, password2)*, l'eliminació d'usuaris "*deleteUser(password)*" el "login" "*login(username, password, user)*," i "log out" "*logout()*" de la sessió i el canvi de contrasenya. També manté l'usuari actual, "*currentUser*", per permetre la gestió de les accions de l'usuari actual.

2.2.14 CtrlDomini

Aquest **controlador de Domini** integra diversos subcontroladors, incloent "*CtrlPersistencia*" per la registió de les dades persistents, "*CtrlBoard*" per operacions relacionades amb els taulers, "*CtrlGame*" per la gestió de partides, "*CtrlsUser*" per operacions d'usuari, i "*CtrlRanking*" per utilitzar les classificacions dels usuaris.

2.2.15 CtrlPersistencia

La classe **controlador de Persistència** és l'encarregada de gestionar la persistència dels taulers i dels usuaris.
Aquesta classe conté dos atributs dels stubs de taulers "*ctrlBoardStorage*" i "*ctrlUsersStorage*".
Respecte els taulers, aquesta classe conté mètodes per afegir i eliminar-ne (*"addBoard(board)"* i *"removeLastBoard()"*), per obtenir el número de taulers que hi ha al sistema *"getNumBoards()"* i per obtenir taulers a partir del seu índex *"getBoard(i)"*. De manera semblant a la gestió de taulers, aquesta classe també conté funcions per gestionar els usuaris del sistema. Entre elles tenim *"addUser(user)"* que ens permet afegir un usuari al sistema d'emmagatzematge dels usuaris, *"getUser(username)"* per obtenir un usuari donat el seu username, "*existsUser(username)"* per saber si un nom d'usuari existeix i *"deleteUser(username)"* per eliminar un usuari del sistema.

2.2.16 CtrlBoardStorage

La classe **'CtrlBoardStorage'** representa un controlador per administrar la col·lecció de taulers del joc.
Consisteix en una llista de taulers List<Board> anomenada "*boards*". Els mètodes d'aquesta classe ens permeten afegir *"addBoard(board)"*, eliminar taulers *"removeLastBoard()"*, així com obtenir-los a partir de l'índex *"getBoard(i)"*, obtenir el número de taulers del sistema *"getNumBoards()"* i obtenir la llista sencera de taulers *"getBoardList()"*.

2.2.17 CtrlUsersStorage

La classe **'CtrlUsersStorage**' representa el mecanisme d'emmagatzematge per gestionar les dades dels usuaris en el sistema. Consisteix en un HashMap on la clau és el nom d'usuari i el valor és l'objecte usuari.
Ens proporciona mètodes per afegir *"addUser(user)"*, obtenir *"getUser(username)"*, verificar si existeixen *"existsUser(username)*" i eliminar usuaris *"deleteUser(username)"*.

3. RELACIÓ DE LES CLASSES IMPLEMENTADES PER CADA MEMBRE

ALBERT	SERGI	GERARD	BART
Board	Stats	User	Ranking
CtrlBoard	Cell	CtrlUser	CtrlRanking
Solver	Region	DriverUser	DriverRanking
Game	Addition	ExceptionUser	ExceptionBoard
CtrlGame	Substraction	ExceptionRankingEmpty	ExceptionGame
DriverGame	Multiplication	CtrlPersistence	Game
Pair	Division	CtrlUserStorage	CtrlGame
CtrlDomini	Module	CtrlDomini	DriverGame
CtrlBoardStorage	Equal	CtrlUserTest	CtrlDomini
SolverTest	Power	UserTest	GameTest
BoardTest			
CellTest			
RegionTest			

4. DESCRIPCIÓ D'ALGORISMES I ESTRUCTURES DE DADES

4.1 Estructures de dades

4.1.1 List<Region> + List<User> + List<Board>

Per emmagatzema les regions d'un tauler i els usuaris del ranking hem triat utilitzar llistes, i per implementar el stub que simula la persistencia dels taulers hem fet servir una llista de taulers. Les llistes ens proporcionen un accés seqüencial als seus elements. Això ens permet recórrer les posicions en el cas de les regions de manera senzilla. Ens afavoreix en la implementació d'algorismes que requereixen processar totes les regions del tauler. També ens aporta flexibilitat i facilitat per afegir, eliminar i modificar elements segons les necessitats.
En quant el cost, la inserció i eliminació d'elements té normalment un cost constant Θ(1) però en els el pitjor dels casos pot arribar a tenir un cost lineal Θ(n) sent n el nombre de regions de la llista o d'usuaris en l'altre cas.

4.1.2 Cel[][]

Aquesta matriu bidimensional representa el tauler del joc, on cada element representa una cel·la. L'ús d'una matriu ens facilita el accès directa a cel·les especifiques segons la seva posició en el tauler (fila i columna). La matriu ocupa Θ(n²), on n és la dimensió del tauler, i l'accés a cada element es realitza en temps constant Θ(1).

4.1.3 HashMap<Integer, Game> + HashMap<String, User>

Hem utilitzat dos HashMap per la nostre implementació, un per guardar les partides que té iniciades un usuari, i un altre HashMap en el stub que simula la persistencia d'usuaris, per poder guardar els usuaris que es registren en el joc.
Aquests diccionaris ens permeten associar un identificador (int o string) únic de cada tauler o usuari amb una instancia de partida (Game) o usuari (User). El cost d'espai dependrà del nombre de partides guardades i del nombre d'usuaris, per tant tindran un cost mitjà de Θ(n), per altre banda l'ús de hashMap ens asegura que el cost mitjà, a l'hora d'accedir, inserir i eliminar dades, serà de Θ(1).

4.1.4 Pair<X, Y>

És una estructura de dades genérica que permet emmagatzemar i manipular un parell de valors de qualsevol tipus. Ens ha sigut útil per mantenir coordenades

mitjançant parelles d'enters. El cost d'espai és de Θ(2) degut a que guarda dos valors, i el cost constant d'accés i modificació de valors és de Θ(1).

4.2 Algorismes

4.2.1 Solve

Aquest algorisme està implementat a la classe Solver i es basa en el backtracking per resoldre els KenKen. Comença per la primera cel·la i continua recursivament, provant totes les possibles assignacions de valors. A cada pas, l'algoritme verifica si s'ha arribat al final del board. En cas que sí, es comprova la solució mitjançant la crida al mètode "board.checkSolution()". Si la solució no és vàlida, l'algorisme selecciona la següent cel·la a explorar i li assigna valors. Utilitza un bucle que recorre tots els possibles valors entre 1 i la mida del tauler i verifica la validesa de cada valor per la cel·la cridant a "validValue". Aquest mètode modifica temporalment el valor de la cel·la actual i verifica que encaixi amb les restricions de la fila i columna. En cas que el valor sigui valid l'algorisme continua recursivament amb la següent cel·la i si no, tira enrere i prova el següent valor a la cel·la anterior.
L'algorisme continua assignant valors i retrocedint fins trobar una solució vàlida o fins que s'hagin provat totes les combinacions.

En el pitjor dels casos l'algorisme ha de provar tots els valors posibles (n) en totes les cel·les del tauler (n x n), per tant hi ha n^n^2 possibilitats. Això fa que el cost temporal d'aquest escenari sigui O(n^n^2).
Per altre banda, en el millor cas, les cel·les ja tenen els valors correctes i per tant, només cal fer un recorregut del tauler (n x n) per verificar que cada valor és correcte. Aquest escenari té un cost lineal de O(n^2).

4.2.2 MakeBoard

L'algorisme "makeBoard" de la classe "CtrlBoard" serveix per crear un nou tauler, a partir d'una llista d'enters que conté tota la informació necessària per crear un tauler, incloent-hi la mida, el nombre de regions, i per a cada regió la seva operació, el resultat i les coordenades de totes les cel·les que pertanyen a la regió.
A partir de la llista d'enters anomenada "boardInfo" que rep l'algorisme, s'extreu la mida del tauler i el nombre de regions, després es fa tantes iteracions com regions té per extreure tota la informació necessària per crear cadascuna de les regions mitjançant "makeRegion". Finalment, es fa el càlcul de la dificultat del tauler amb "calculateDifficulty".

El cost d'aquest algorisme depèn de la mida de la llista d'informació del tauler (n) i del nombre total de cel·les en totes les regions, li direm (m). Dins del bucle principal que recorre tota la llista es van fent operacions bàsiques d'assignació i accés als elements que tenen cost O(1). Per cada regió es recorre la llista de cel·les. Si el nombre de cel·les és m llavors el cost de recórrer-les totes seria de O(m).
En el pitjor dels casos, el cost d'aquest algorisme és proporcional a la suma dels dos valors, per tant, O(n+m).

4.2.3 CreateBoardFromTerminal

Aquest algorisme crea un tauler a partir de valors que entra l'usuari a través de la terminal. Comença demanant a l'usuari la mida del tauler i el número de regions que tindrà. El cost d'aquesta part és O(1) ja que només es realitzen operacions d'entrada i comprovacions.

A continuació es passa a l'entrada dels continguts de cada regió. Per cada una, l'algorisme demana l'operació, el resultat i les cel·les. El seu cost s'establirà en funció del número de regions i la quantitat de cel·les que tingui cada regió. Com que és un bucle extern que s'executa per N regions i un altre intern que s'executa M vegades, sent M el màxim número de cel·les en una regió, el cost total serà O(N*M).

Una vegada s'ha obtingut la informació de totes les cel·les i regions del tauler, es crida a makeBoard. A continuació l'analitzarem en detall:

La primera part d'aquesta funció té un cost constant O(1). S'assignen valors i es creen estructures.
El bucle principal itera sobre cada regió del tauler. Dins del bucle:

- S'agafen valors (cost O(1)).
- Es crea la llista cells que tindrà un cost O(c) sent c el número de cel·les de la regió.
- Es crida a makeRegion. El seu cost dependrà del número de cel·les de la regió actual.

Un cop fora del bucle principal, es crida a "calculateDifficulty()" que itera sobre totes les regions del tauler i realitza operacions constants. Aleshores, el cost d'aquesta funció depèn del número total de regions, O(r).
Per tant el cost de makeBoard sería la suma dels costos de totes les operacions que s'han comentat, és a dir: O(1) + O(r * (1+c)) + O(r). Recordem que r és el número total de regions del taulell i c és el número de cel·les.

L'últim pas de l'algorisme és retornar l'id, això ens sumara un cost constant O(1).
Cost final de "CreateBoardFromTerminal" → O(N*M) sent N el número de regions i M el número màxim de cel·les per regió.

5. JOCS DE PROVA

A la carpeta Primera Entrega/DOCS/Jocs de Proves, es troben les diferents funcionalitats dels diferents drivers. Cada carpeta representa un driver diferent que conté els .txt amb les funcionalitats d'aquell propi driver

Jocs de prova Ranking

- **OrderRankingByDifficulty**: Ordena els usuaris generats aleatoris, pel número de kenkens resolts de la dificultat indicada.
- **OrderRankingByPoints**: Ordena els usuaris generats aleatoris, pels punts dels usuaris.

- **OrderRankingByName**: Ordena els usuaris generats aleatoris, pel username dels usuaris per ordre alfabètic.
- **OrderRankingByNumberOfSolved**: Ordena els usuaris generats aleatoris, pel número de kenkens resolts del size indicat.
- **OrderRankingByTime**: Ordena els usuaris generats aleatoris, pel temps amb el size i dificultat indicat.

Jocs de prova User:

- **createUser**: crea un usuari a partir d'un username, un password i una confirmació d'aquest password escrit per l'usuari a través de la terminal.
- **login**: logueja un usuari dins del sistema a partir d'un username i un password escrit per l'usuari a través
- **changePassword**: canvia la contrasenya d'un usuari loguejat en el sistema. Es necessita la contrasenya actual de l'usuari per verificar que és l'usuari actual el que utilitza el sistema, i la nova contrasenya amb la seva confirmació per canviar-la.
- **logout:** deslogueja un usuari dins del sistema sense necessitar cap atribut.
- **deleteUser**: s'elimina un usuari del sistema. Es necessita la contrasenya de l'usuari actual loguejat al sistema.

Jocs de prova de Game:

- **createBoardTerminal(getHint):** es crea un tauler a partir d'unes dades introduïdes per la terminal (size, num regions, i les cel·les pertinents), i es resol un kenken a través d'escollir l'opció getHint tota l'estona.
- **createBoardTerminal(playGame):** es crea un tauler a partir d'unes dades introduïdes per la terminal (size, num regions i les cel·les pertinents), i es resol un kenken a través d'escollir l'opció playGame i afegint per la terminal la posició i el valor que es vol introduir en el tauler.
- **createBoardTerminal(solveBoard):** es crea un tauler a partir d'unes dades introduïdes per la terminal (size, num regions, i les cel·les pertinents), i es resol un kenken a través d'escollir l'opció solveBoard, que resol el kenken directament.
- **createBoardTxt(getHint):** es crea un tauler a partir d'un fitxer de text introduït per l'usuari. Aquest tauler es resol a partir d'anar demanant pistes.
- **createBoardTxt(playGame):** es crea un tauler a partir d'un fitxer de text introduït per l'usuari. Es resol el nivell creat amb l'opció playGame que a partir dels inputs de posició i valor introduïts va omplint el tauler.
- **createBoardTxt(solveBoard):** es crea un tauler a partir d'un fitxer de text introduït per l'usuari. Aquest tauler es resol mitjançant la funció solveBoard que mostra la solució final del KenKen directament.
- **solveGame(getHint):** es resol un tauler a partir de l'ús de les pistes per completar tots els valors del tauler
- **solveGame(makeMove):** es crea una partida i es fan diferents moviments dins del tauler fins arribar a resoldre'l.
- **startGame:** s'inicia una partida amb el tauler identificat amb número 0.