PROP: Scrabble

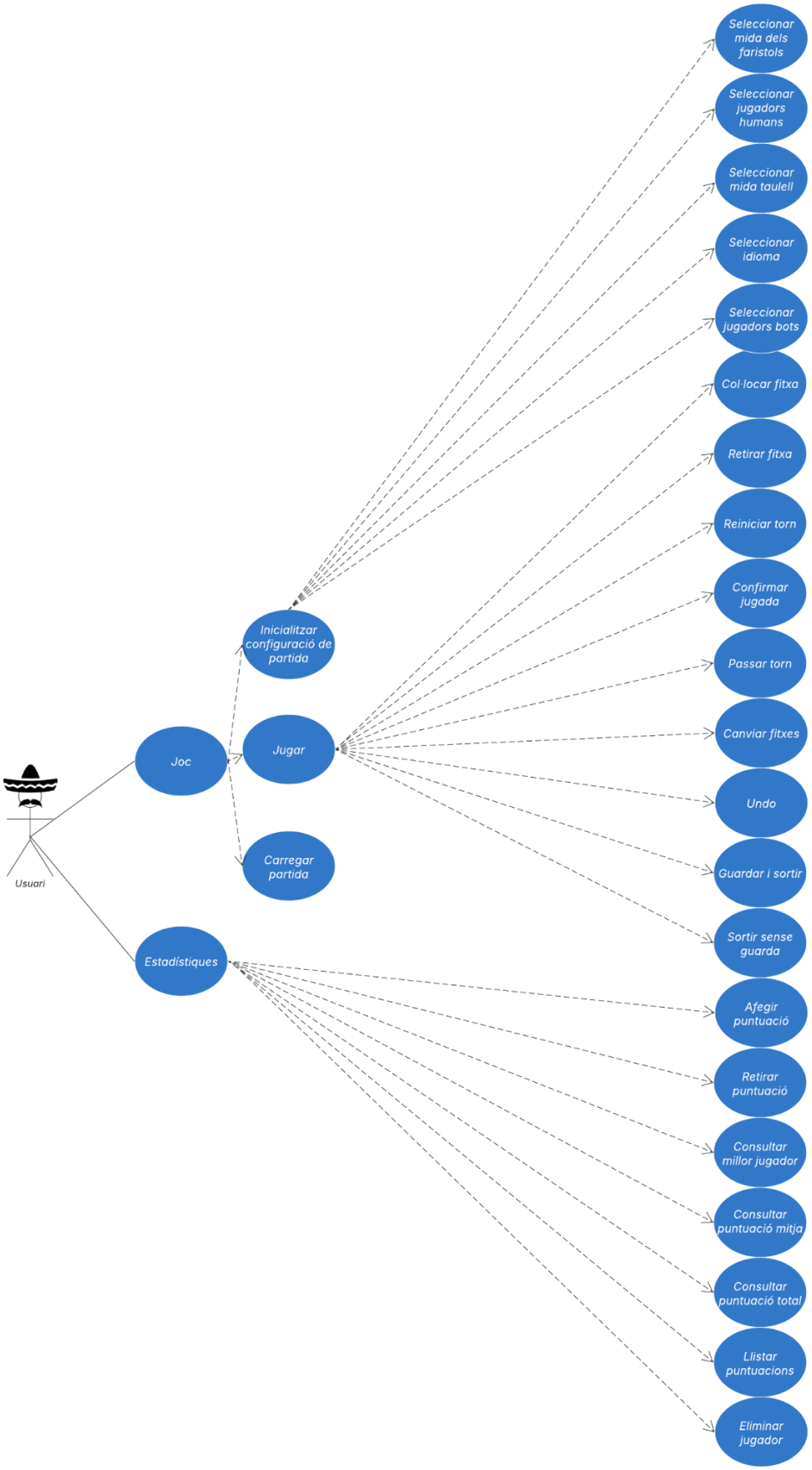
Grup 13.4

Entrega 2.0

Quadrimestre de primavera, curs 24/25

Toni Gratacós Miralles | **ToniGratacosMiralles**
Jan Pruneda Marcet | **JanPru**
Camila Valeria Santos Cabrera | **camivsantos**
Roger Bitlloch Galceran | **MUX-enjoyer**

Casos d'ús


Usuari
Joc
Estadístiques
Inicialitzar configuració de partida
Jugar
Carregar partida
Seleccionar mida dels faristols
Seleccionar jugadors humans
Seleccionar mida taulell
Seleccionar idioma
Seleccionar jugadors bots
Col·locar fitxa
Retirar fitxa
Reiniciar torn
Confirmar jugada
Passar torn
Canviar fitxes
Undo
Guardar i sortir
Sortir sense guarda
Afegir puntuació
Retirar puntuació
Consultar millor jugador
Consultar puntuació mitja
Consultar puntuació total
Llistar puntuacions
Eliminar jugador

Partida

Nom	Inicialitzar configuració de partida
Actor	Usuari
Comportament	1) L'usuari selecciona "Nova Partida" en el menu principal 2) El sistema mostra una pestanya amb totes les opcions de configuració pas a pas 3) Per cada pas: - Si el valor introduït és vàlid, es continua al següent. - Si el valor és invàlid, es segueix el Flux alternatiu 1 - Si l'usuari cancel·la en qualsevol moment, es segueix el Flux alternatiu 2.
Errors possibles i cursos alternatius	1) El sistema mostra un missatge d'error especific del camp que s'estava entrant. Seguidament es torna a demanar la dada corresponent. 2) El sistema retorna l'usuari a la pantalla principal.

Nom	Seleccionar mida del taulell
Actor	Usuari
Comportament	1) El sistema demana la mida del taulell 2) L'usuari introdueix un valor numèric 3) El sistema valida la mida i l'accepta, si no és vàlida es segueix el flux alternatiu 1
Errors possibles i cursos alternatius	1) Si la mida es invalida: - El sistema mostra un missatge: "La mida ha de ser un enter positiu" - Es torna a demanar la mida del taulell

Nom	Seleccionar mida del faristol
Actor	Usuari
Comportament	1) El sistema demana la mida del faristol 2) L'usuari introdueix un valor numèric 3) El sistema valida la mida i l'accepta, si no és vàlida es segueix el flux alternatiu 1
Errors possibles i cursos alternatius	1) Si la mida es invalida: - El sistema mostra un missatge: "La mida del faristol ha de ser un enter positiu superior o igual a 1" - Es torna a demanar el valor

Nom	Seleccionar idioma del diccionari
Actor	Usuari
Comportament	1) El sistema mostra els idiomes disponibles (català, castellà, anglés) 2) L'usuari selecciona un d'ells 3) El sistema valida i guarda l'acció, si no és vàlid es segueix el flux alternatiu 1
Errors possibles i cursos alternatius	1) Si l'idioma no és vàlid o el diccionari és inexistent: - El sistema mostra un missatge: "El diccionari seleccionat no està disponible" - Es torna a mostrar la llista d'idiomes perquè l'usuari pugui seleccionar-ne un altre

Nom	Seleccionar jugadors humans
Actor	Usuari
Comportament	1) El sistema solicita el número de jugadors humans 2) L'usuari introdueix un número 3) Si el número no és vàlid, es segueix el flux alternatiu 1 4) El sistema demana el nom de cada jugador 5) Si el nom no és vàlid es segueix el flux alternatiu 2
Errors possibles i cursos alternatius	1) Si el nombre de jugadors no és vàlid: - El sistema mostra un missatge: "Cal introduir un nombre enter i positiu de jugadors humans" - Es torna a demanar el valor. 2) Si el nom no és vàlid: - El sistema mostra un missatge: "El nom del jugador no pot estar buit ni repetir-se" - Es torna a demanar el nom incorrecte

Nom	Seleccionar jugadors bots
Actor	Usuari
Comportament	1) El sistema solicita el número de jugadors bots 2) L'usuari introdueix un número 3) Si el número no és vàlid, es segueix el flux alternatiu 1 4) El sistema demana la dificultat de cada bot - Fàcil - Nomal - Difícil
Errors possibles i cursos alternatius	1) Si el número no es valid: - El sistema mostra un missatge: "El nombre de bots ha de ser un enter positiu" - Es torna a demanar el valor

Nom	Carregar partida
Actor	Usuari
Comportament	1) L'usuari selecciona "Carregar Partida" en el menú principal 2) El sistema comprova si hi ha partides guardades, si no, es segueix el flux alternatiu 1 3) El sistema mostra les diferents partides guardades amb detalls com: - Data i hora de creació - Jugadors - Estat actual (torn, puntuacions...) 4) L'usuari selecciona la partida que vol continuar 5) El sistema redirigeix a l'usuari a la partida seleccionada 6) En qualsevol moment l'usuari pot cancelar tota la operació i es seguirà el flux alternatiu 2
Errors possibles i cursos alternatius	1) No hi ha cap partida guardada: - El sistema mostra un missatge: "No hi ha cap partida disponible per carregar" - Es retorna al menú principal 2) Cancelar: - El sistema mostra un missatge de confirmació - Si l'usuari confirma, es retorna al menú principal

Nom	Jugar una partida
Actor	Usuari
Comportament	1) El sistema mostra de qui és el torn actual 2) Si es un bot realitza una jugada automàtica 3) Si es un humà, se li mostren totes les opcions d'acció: - Jugar - Passar Torn - Canviar Fitxes - Undo - Guardar i Sortir - Sortir sense guardar 4) L'usuari selecciona una de les opcions, si no és vàlida, es segueix el flux alternatiu 1 5) Un cop finalitzada l'acció es passa el torn i es repeteix el procés
Errors possibles i cursos alternatius	1) Si la opció no és vàlida: - El sistema mostra un missatge: "L'opció seleccionada no és vàlida" - Es tornen a mostrar les opcions disponibles

Nom	Col·locar fitxa
Actor	Usuari
Comportament	1) L'usuari indica la lletra de la fitxa que vol jugar 2) Si la lletra no és vàlida, es segueix el flux alternatiu 1 3) El sistema solicita la fila i la columna on col·locar-la 4) Si la posició no és vàlida, es segueix el flux alternatiu 2 5) Si es un comodín, el sistema demana per quina lletra es vol substituir 6) El sistema actualitza el taulell
Errors possibles i cursos alternatius	1) Fitxa no vàlida: - Es mostra un missatge: "La fitxa seleccionada no es troba al teu faristol" - Es torna a demanar la fitxa 2) Posició no vàlida: - Es mostra un missatge: "La posició seleccionada no és vàlida per col·locar una fitxa" - Es torna a demanar fila i columna

Nom	Retirar fitxa
Actor	Usuari
Comportament	1) L'usuari indica la fila i la columna de la fitxa que vol retirar 2) Si la posició no és vàlida, es segueix el flux alternatiu 1 3) El sistema retira la fitxa i retorna el taulell
Errors possibles i cursos alternatius	1) Posició no vàlida: - Es mostra un missatge: "No has col·locat cap fitxa en aquesta casella en aquesta jugada." - Es demana una altra posició o es cancel·la l'acció

Nom	Reiniciar torn
Actor	Usuari
Comportament	1) L'usuari indica l'opció de reiniciar el torn 2) El sistema comprova si s'han realitzat accions que es poden desfer (com col·locar fitxes) 3) El sistema desfà totes les accions realitzades durant el torn en qüestió 4) Si no es pot reiniciar es segueix el flux alternatiu 1
Errors possibles i cursos alternatius	1) No es pot reiniciar: - El sistema mostra un missatge: "No s'han realitzat accions durant aquest torn" - L'usuari torna al menú d'opcions del torn

Nom	Confirmar jugada
Actor	Usuari
Comportament	1) L'usuari col·loca les fitxes desitjades 2) El sistema valida la jugada, mostra la paraula formada i la puntuació 3) Si la jugada és vàlida, el sistema mostra la opciò de confirmar la jugada, si no és vàlida es segueix el flux alternatiu 1 4) L'usuari selecciona l'opció de confirmar la jugada 5) El sistema assigna els punts i recarrega el taulell
Errors possibles i cursos alternatius	1) Jugada invàlida: - El sistema mostra un missatge com: "La paraula formada no és vàlida" o "Les fitxes no formen una línia contínua" - Es manté l'estat anterior i es permet a l'usuari modificar o reiniciar la jugada

Nom	Passar torn
Actor	Usuari
Comportament	1) L'usuari decideix no jugar i selecciona l'opció de passar el torn 2) El sistema avança al següent jugador.
Errors possibles i cursos alternatius	Cap

Nom	Canviar fitxes
Actor	Usuari
Comportament	1) L'usuari indica quantes fitxes vol canviar 2) El sistema solicita les lletres de les fitxes a intercanviar 3) L'usuari especifica les lletres de les fitxes que vol intercanviar. Si no ho estan, es segueix el flux alternatiu 1. 4) El sistema valida les fitxes i realitza el canvi amb fitxes aleatòries del sac 5) Si no hi ha prou fitxes al sac, es segueix el flux alternatiu 2
Errors possibles i cursos alternatius	1) Fitxes no disponibles: - Es mostra un missatge: "Algunes de les fitxes seleccionades no són al teu faristol" - Es torna a demanar la selecció 2) Fitxes insuficients al sac: - Es mostra un missatge: "No hi ha prou fitxes al sac per fer el canvi" - L'usuari pot triar una altra acció

Nom	Undo
Actor	Usuari
Comportament	3) L'usuari selecciona l'opció Undo (Desfer) 4) El sistema comprova si hi ha una acció previa que es pugui desfer (Ex: Col·locar fitxes) 5) Si no és possible desfer, es segueix el flux alternatiu 1 6) El sistema reverteix l'última acció realitzada
Errors possibles i cursos alternatius	1) No hi ha accions a desfer: - Es mostra un missatge: "No es pot desfer cap acció. El torn actual no conté accions actives" - Es torna al menú d'accions

Nom	Guardar i sortir
Actor	Usuari
Comportament	1) L'usuari selecciona l'opció de guardar i sortir 2) El sistema guarda el estat actual de la partida 3) Si es guarda correctament, el sistema tanca el joc 4) Si no es guarda correctament, es segueix el flux alternatiu 1 5) El sistema tanca el joc
Errors possibles i cursos alternatius	1) No es pot guardar: - Es mostra un missatge: "No s'ha pogut desar la partida" - L'usuari torna al joc amb la possibilitat de tornar-ho a intentar

Nom	Sortir sense guardar
Actor	Usuari
Comportament	1) L'usuari selecciona l'opció de sortir sense guardar 2) El sistema tanca la partida sense guardar els canvis
Errors possibles i cursos alternatius	Cap

Estadístiques

Els casos d'ús de Afegir puntuació i Retirar puntuació es canviaran per indicar que els fa el sistema automàticament, no obstant ara els fa l'usuari ja que no tenim capa de persistencia.

Nom	Afegir puntuació
Actor	Usuari
Comportament	1) L'usuari indica el nom del jugador i la puntuació 2) Si el nom del jugador o la puntuació no son vàlids, es segueix el flux alternatiu 1 3) El sistema afegeix la puntuació al seu historial
Errors possibles i cursos alternatius	1) Nom o puntuació no vàlids: - Es mostra un missatge: "El nom no pot estar buit i la puntuació ha de ser un enter positiu" - Es torna a demanar la informació

Nom	Retirar puntuació
Actor	Usuari
Comportament	1) L'usuari indica el nom del jugador i la puntuació a retirar 2) Si el nom del jugador o la puntuació no son vàlids, es segueix el flux alternatiu 1 3) El sistema elimina aquella puntuació
Errors possibles i cursos alternatius	1) Si la puntuació no existeix: - Es mostra un missatge: "No s'ha trobat cap puntuació amb aquests valors pel jugador indicat" - Es demana repetir l'acció.

Nom	Consultar millor jugador
Actor	Usuari
Comportament	1) L'usuari selecciona l'opció de veure el millor jugador 2) El sistema consulta les puntuacions registrades 3) Si no hi ha una puntuació registrada, es segueix el flux alternatiu 1 4) El sistema mostra el jugador amb major puntuació amb la seva puntuació associada
Errors possibles i cursos alternatius	1) No hi ha puntuacions guardades: - Es mostra un missatge: "Encara no s'ha registrat cap puntuació".

Nom	Consultar puntuació mitja
Actor	Usuari
Comportament	1) L'usuari selecciona l'opció de veure la puntuació mitja 2) El sistema consulta les puntuacions registrades 3) Si no hi ha una puntuació registrada, es segueix el flux alternatiu 1 4) El sistema mostra la mitja de totes les puntuacions de tots els jugadors registrats
Errors possibles i cursos alternatius	1) No hi ha puntuacions guardades: - Es mostra un missatge: "Encara no s'ha registrat cap puntuació".

Nom	Consultar puntuació total
Actor	Usuari
Comportament	1) L'usuari selecciona l'opció de veure la puntuació total 2) El sistema consulta les puntuacions registrades 3) Si no hi ha una puntuació registrada, es segueix el flux alternatiu 1 4) El sistema mostra la suma de totes les puntuacions de tots els jugadors registrats
Errors possibles i cursos alternatius	1) No hi ha puntuacions guardades: - Es mostra un missatge: "Encara no s'ha registrat cap puntuació".

Nom	Llistar puntuacions
Actor	Usuari
Comportament	1) L'usuari selecciona l'opció de veure les puntuacions 2) El sistema consulta les puntuacions registrades 3) Si no hi ha una puntuació registrada, es segueix el flux alternatiu 1 4) El sistema mostra totes les puntuacions, ordenades de major a menor
Errors possibles i cursos alternatius	1) No hi ha puntuacions guardades: - Es mostra un missatge: "Encara no s'ha registrat cap puntuació".

Nom	Eliminar jugador
Actor	Usuari

Comportament	1) L'usuari selecciona l'opció d'eliminar un jugador 2) L'usuari indica el nom del jugador a eliminar 3) Si el nom no és vàlid, es segueix el flux alternatiu 1 4) Si el nom és vàlid, el sistema elimina totes les puntuacions registrades d'aquest jugador
Errors possibles i cursos alternatius	1) Nom del jugador invàlid: - Es mostra un missatge: "El jugador indicat no existeix al sistema" - Es demana introduir un nom vàlid

Capa de Presentació

Preàmbul

La capa de presentació és l'encarregada de gestionar la interfície gràfica del joc i d'interactuar amb l'usuari final. Està dissenyada seguint el patró MVC (Model-Vista-Controlador), separant clarament la representació visual de la lògica del joc. Aquesta capa tradueix l'estat intern del model en una representació gràfica coherent i interactiva, facilitant l'experiència de joc i la manipulació dels elements del tauler i del faristol.

Aquesta estructura es divideix principalment en dos grans blocs:

- Vistes de components del joc, que representen elements individuals com fitxes, caselles, faristols o jugadors, oferint funcionalitats específiques per a cada un d'ells.
- Pantalles completes, que integren diversos components visuals per formar les escenes principals del joc, com ara la pantalla d'inici, la de personalització de partida o la vista completa d'una partida en curs.

Cada classe visual s'encarrega de la seva pròpia representació gràfica i comportament interactiu, utilitzant tecnologies de la biblioteca Swing i aplicant estils consistents mitjançant recursos comuns (colors, fonts i estils). A més, totes les vistes mantenen sincronització amb l'estat del model i interactuen amb el controlador de presentació per a gestionar les accions de l'usuari.

Vistes | Components del joc

FitxaVista
FitxaVista representa visualment una fitxa del joc de Scrabble. Mostra la lletra de la fitxa al centre i els punts de valor a la cantonada inferior dreta, utilitzant fonts personalitzades i un fons acolorit. El disseny incorpora detalls gràfics com un contorn definit i un efecte visual per indicar si la fitxa està seleccionada.

Aquesta classe permet seleccionar i desseleccionar la fitxa mitjançant clics de ratolí, canviant el seu color de fons per reflectir l'estat actual. També pot adaptar-se a diferents mides i mides de font, la qual cosa la fa útil tant per representar fitxes a la mà del jugador com al tauler. Aquesta interacció visual facilita la construcció de paraules i altres accions relacionades amb el joc.

FaristolVista
La classe FaristolVista mostra gràficament el faristol del jugador amb totes les fitxes disponibles, organitzades visualment en un panell. Inclou un botó per barrejar les fitxes, que crida el mètode corresponent del model i actualitza automàticament la visualització. En mode normal, permet seleccionar una única fitxa per col·locar-la al tauler; aquesta selecció es destaca visualment i es pot obtenir mitjançant un mètode específic. En canvi, si s'activa el mode de canvi de fitxes, l'usuari pot

seleccionar-ne diverses per intercanviar-les. La classe proporciona mètodes per obtenir les fitxes seleccionades, desseleccionar-les, i canviar el mode de selecció. També es responsabilitza d'actualitzar la vista cada cop que es produeix un canvi, assegurant que l'estat gràfic reflecteixi l'estat intern del model Faristol.

JugadorVista

La classe JugadorVista s'encarrega de representar visualment un jugador dins la interfície gràfica, mostrant el seu nom, la seva puntuació i el seu faristol. Està construïda mitjançant diversos panells Swing personalitzats, amb estils carregats des de FontLoader i ColorLoader, per mantenir una coherència visual amb la resta de l'aplicació. Aquesta vista integra el component FaristolVista, que reflecteix les fitxes disponibles del jugador, i s'actualitza automàticament quan canvia l'estat del jugador.

A més, JugadorVista disposa de funcionalitats per actualitzar dinàmicament la informació mostrada i per destacar el jugador actiu visualment, modificant-ne la vora i el títol. Proporciona accés a les fitxes seleccionades o marcades per canvi, permet desseleccionar fitxes, i activa o desactiva el mode de canvi segons convingui. Amb això, actua com a pont entre la lògica del joc i la seva representació visual, mantenint sincronitzats estat i interfície.

CasellaVista

La classe CasellaVista és un component visual que representa una casella individual del tauler de joc, mostrant tant el tipus de casella (normal, multiplicador de lletra, multiplicador de paraula) com el contingut, és a dir, la fitxa que hi pugui haver col·locada. Aquesta vista hereta de JPanel i utilitza els colors definits a ColorLoader i les fonts de FontLoader per garantir una coherència gràfica amb la resta de l'aplicació. A través d'un MouseListener, permet la interacció directa amb l'usuari per seleccionar o desseleccionar caselles, canviant-ne visualment el color de fons.

El comportament visual de la casella s'adapta segons el seu estat: si està buida, es mostra el tipus de multiplicador amb una lletra indicadora ("L" o "P") i el valor corresponent (x2 o x3); si conté una fitxa, es dibuixa un rectangle amb la lletra i la puntuació d'aquesta. El mètode paintComponent personalitza la manera com es renderitza cada element visual, centrant el text i aplicant estils que recorden les fitxes físiques d'un joc de Scrabble.

TaulellVista

La classe TaulellVista representa visualment el tauler de joc del Scrabble. Mostra totes les caselles en una graella quadrada, inicialitzada a partir del model Taulell, amb una casella per cada posició lògica. Cada casella es visualitza mitjançant una instància de CasellaVista, i s'organitzen mitjançant un GridLayout.

Aquest component permet la interacció directa amb les caselles mitjançant clics del ratolí. En cada clic, es marca visualment la casella seleccionada, es desa la posició seleccionada com a parell de coordenades, i s'emet una notificació a través d'un listener de selecció (SeleccioListener). També destaca la casella central amb un contorn groc, fent-la fàcilment identificable.

Vistes | Pantalles

PantallaIniciVista
La classe PantallaIniciVista implementa la pantalla inicial del joc de Scrabble, donant accés a les funcionalitats principals del joc. Mostra un missatge de benvinguda "Benvingut a", seguit del logotip "SCRABBLE" format amb fitxes reals del joc.
Inclou quatre botons principals disposats verticalment:

- "*Jugar Nova Partida*": Inicia una partida nova.
- "*Continuar Partid*a": Mostra les partides guardades per continuar-ne una.
- "*Estadístiques*": Obre una finestra amb les estadístiques del jugador.
- "*Sortir*": Tanca completament l'aplicació.

En iniciar l'aplicació, l'usuari pot escollir entre començar una partida, continuar-ne una, consultar les estadístiques o sortir del joc, accedint fàcilment a totes les opcions principals.
Aquesta classe segueix el patró MVC i es comunica amb el CtrlPresentacio.

PantallaPersonalitzacioVista
La classe PantallaPersonalitzacioVista implementa la pantalla de configuració de partida del joc de Scrabble, permetent a l'usuari personalitzar tots els paràmetres necessaris abans d'iniciar una nova partida. La pantalla permet configurar els paràmetres bàsics del joc, com la mida del taulell, que determina les dimensions del tauler de joc, la mida del faristol, que estableix quantes fitxes pot tenir cada jugador, i la selecció de l'idioma, que pot ser Català, English o Castellano. Aquests valors es validen per assegurar que siguin correctes abans de continuar amb la configuració.

A més, el sistema permet configurar bots automatitzats mitjançant un camp numèric on es pot especificar la quantitat desitjada. Quan s'aplica el nombre de bots, es genera dinàmicament una interfície que permet assignar un nom personalitzat a cada bot i seleccionar-ne el nivell de dificultat entre tres opcions: 1, 2 o 3. La interfície es reconstrueix automàticament si es canvia el nombre de bots, adaptant-se a la nova configuració.

De manera similar, també es pot indicar el nombre de jugadors humans que participaran a la partida. En aplicar aquest valor, la pantalla crea una interfície que demana el nom de cada jugador. El sistema valida que tots els jugadors tinguin un nom assignat abans de permetre continuar.
La pantalla implementa un sistema de validació complet que verifica que tots els camps estiguin degudament emplenats per a no fer saltar excepcions a la capa de domini. Es comprova que les mides del taulell i del faristol siguin valors positius, que hi hagi com a mínim un participant, i que tots els noms estiguin especificats. En cas d'error, es mostra un missatge informatiu indicant què cal corregir.

Per facilitar l'inici ràpid d'una partida, la classe ofereix una configuració predeterminada amb valors per defecte: un taulell de 15x15, un faristol de 7 fitxes, idioma català, un bot amb dificultat 1 anomenat "Bot 1" i un jugador anomenat "Loser". Aquesta configuració permet començar una partida amb un sol clic.

Un cop validada tota la configuració, la pantalla es comunica amb el controlador de presentació per inicialitzar la partida amb els paràmetres seleccionats. Això inclou les dimensions del taulell, la mida del faristol, l'idioma, els noms dels jugadors i la configuració dels bots amb els seus nivells de dificultat.

PartidaVista

La classe PartidaVista representa la vista principal d'una partida del nostre joc d'Scrabble. Funciona com a contenidor principal que integra i coordina tots els elements visuals necessaris per a l'experiència de joc explicats en l'apartat anterior, proporcionant una interfície completa perquè els jugadors interactuïn amb el taulell, gestionin les seves fitxes i controlin el desenvolupament de la partida.

La vista incorpora una representació gràfica del taulell mitjançant la classe TaulellVista, que permet als jugadors visualitzar l'estat actual de la partida de forma clara. El sistema gestiona la selecció de caselles amb esdeveniments de ratolí. La informació del jugador actiu es mostra a través de JugadorVista, on es poden veure i seleccionar les fitxes disponibles. El sistema gestiona el faristol del jugador actiu i permet activar el mode de canvi per intercanviar peces no desitjades quan calgui.

Un panell específic mostra les puntuacions dels jugadors, indicant clarament qui té el torn. Aquestes puntuacions s'actualitzen després de cada jugada i el panell s'adapta al nombre de participants, utilitzant un sistema de desplaçament si és necessari.

Els controls de joc consisteixen en els casos d'ús principals com: "Validar Jugada" per confirmar el moviment segons les regles, "Col·locar" per posicionar fitxes al taulell, "Canviar Fitxes" per activar l'intercanvi, "Passar Torn" per cedir-lo sense jugar, etc... A més, hi ha un botó destacat per desar la partida, que permet conservar-ne l'estat per continuar-la més endavant.

SelectorPartidesVista

La classe SelectorPartidesVista ofereix una interfície gràfica per seleccionar partides guardades, permetent navegar, seleccionar i carregar partides prèviament desades.

L'usuari accedeix al selector quan vol carregar una partida guardada, i la vista mostra la llista de partides disponibles o, si no n'hi ha cap, un missatge explicatiu. La selecció es confirma amb un clic, que habilita el botó per carregar la partida, i l'usuari pot triar entre carregar-la o cancel·lar i sortir en qualsevol moment.

La classe disposa de mètodes específics dedicats a cada part de la vista, com el títol, la llista i els botons. El contingut es gestiona de manera dinàmica segons les dades disponibles. Per a la interacció, s'utilitzen listeners que capturen esdeveniments que comuniquen amb el controlador.

GuanyadorVista

La classe GuanyadorVista s'encarrega de mostrar la pantalla final d'una partida de joc. La seva funció principal és presentar de manera visual i atractiva el resultat de la partida, destacant el guanyador i proporcionant un resum complet dels resultats.

Aquesta vista està estructurada en tres seccions principals distribuïdes estratègicament a la pantalla. A la part superior hi ha un títol que indica el final de la partida, juntament amb un panell que mostra el nom del guanyador i la seva puntuació final. El centre de la pantalla està ocupat per una representació del tauler final del joc.

La secció lateral dreta conté un rànquing complet de tots els jugadors participants. Aquest rànquing mostra cada jugador ordenat segons la seva puntuació final, indicant la seva posició, nom i punts obtinguts.

En termes de funcionalitat, la vista incorpora dos botons d'acció que permeten a l'usuari continuar el joc. El botó "Nova Partida" permet iniciar immediatament una nova partida amb els mateixos paràmetres, mentre que "Tornar al Menú" retorna l'usuari a la pantalla principal del joc. Ambdós botons inclouen efectes visuals d'hover que milloren l'experiència d'usuari.

EstadistiquesVista

La classe EstadistiquesVista és una finestra que presenta les estadístiques històriques de puntuació dels jugadors del joc. La seva funció és mostrar un leaderboard o taula de classificació on es poden consultar els resultats de partides anteriors de manera organitzada i clara.

Aquesta vista es presenta com una finestra independent de mida fixa que es posiciona automàticament al centre de la pantalla de l'usuari. La funcionalitat central de la classe consisteix a recuperar les puntuacions emmagatzemades a través del sistema singleton d'Estadistiques i processar-les per crear una classificació ordenada de major a menor puntuació.

Controlador Presentació

Justificació del patró Singleton aplicat a CtrlPresentacio

La classe CtrlPresentacio implementa el patró Singleton per assegurar que només existeixi una instància encarregada de gestionar la interfície gràfica del sistema. Aquesta decisió és fonamental per mantenir la coherència visual de l'aplicació, evitant la creació de múltiples finestres duplicades o inconsistents que podrien derivar en errors visuals o desincronitzacions amb l'estat del joc.
Tot i que la classe manté referències a diferents vistes (com PantallaIniciVista, PartidaVista, etc.), aquestes no necessiten una gestió personalitzada per instància. L'estat que controla és uniforme, i per tant no cal tenir diverses instàncies amb comportaments diferents. La centralització d'aquesta lògica de presentació en una sola instància facilita el control del flux de la interfície d'usuari de manera més eficient i segura.

A més, utilitzar CtrlPresentacio com a Singleton facilita l'accés global a la capa de presentació des d'altres parts del sistema, com ara CtrlDomini, sense haver de passar referències explícites entre capes. Això contribueix a un codi més net, modular i mantenible, on la separació de responsabilitats es manté clara i ben estructurada.

Secció	Contingut
Nom	CtrlPresentacio
Descripció	Classe encarregada de gestionar el flux visual del sistema. Administra les vistes gràfiques (pantalla inicial, personalització, partida, etc.) i sincronitza aquestes amb el model de domini (CtrlDomini). Encapsula la lògica d'interacció de l'usuari amb el sistema: col·locar fitxes, validar jugades, carregar/guardar partides, etc.
Atributs	- instance (CtrlPresentacio): instància única (patró Singleton) - ctrlDomini (CtrlDomini): controlador del domini - pantallaInici, pantallaPersonalitzacioVista, partidaVista: vistes de la interfície - framePartida (JFrame): finestra principal del joc - paraulaValida (boolean): indicador de validesa de la jugada actual
Mètodes destacats	**inicialitzarApp()**: Inicia l'aplicació mostrant la pantalla inicial **configurarPartida()**: Passa de la vista d'inici a la de configuració de la partida Excepcions: NullPointerException **inicialitzarPartida(...)**: Crida a CtrlDomini per iniciar una nova partida i crea la vista de joc Excepcions: bon dia toni IOException IllegalArgumentException **crearFramePartida()**: Prepara la finestra del joc (JFrame) i hi afegeix la vista de partida Excepcions: NullPointerException **passarTorn()**: Passa el torn al següent jugador i actualitza la vista Excepcions: IllegalStateException **retirarFitxa()**: Retira una fitxa prèviament col·locada en el tauler Excepcions:

	IndexOutOfBoundsException **colocarFitxa()**: Col·loca una fitxa a la posició seleccionada del tauler Excepcions: IllegalArgumentException **commitParaula()**: Valida i confirma la paraula col·locada pel jugador Excepcions: IllegalStateException InvalidWordException **gestionarComodi(Fitxa fitxa)**: Permet assignar una lletra a una fitxa comodí mitjançant un diàleg Excepcions: NullPointerException **canviarFitxes()**: Gestiona l'intercanvi de fitxes i actualitza la vista Excepcions: IllegalStateException EmptyBagException **actualitzarVistes()**: Recarrega el tauler i les dades dels jugadors a la vista actual Excepcions: (none) **guardarPartida()**: Desa l'estat actual de la partida amb ajuda de CtrlDomini Excepcions: IOException **mostrarFinalPartida(...)**: Mostra la vista de final de partida amb els resultats Excepcions: (none) **mostraSelectorPartides()**: Obre un diàleg per seleccionar i carregar una partida guardada Excepcions: IOException ClassNotFoundException
Punts forts	- Separació clara de responsabilitats - Bon ús del patró MVC - Gestió eficaç dels estats visuals - Interacció amigable amb l'usuari (JOptionPane) - Flexibilitat funcional: carregar partides, mostrar resultats, gestionar comodins, etc.

Possibles millores	- Modularització de mètodes llargs (canviarFitxes(), colocarFitxa()) - Desacoblament de Swing per facilitar canvis futurs de GUI - Gestió d'errors més centralitzada (encapsular crides a JOptionPane) - Ús coherent de lambda expressions en listeners

Capa de Domini

Preàmbul

Per desenvolupar el nostre projecte de Scrabble, hem pensat en una estructura basada en classes que representin els diferents elements reals del joc. L'objectiu ha estat mantenir una separació clara de responsabilitats i reflectir de la millor manera possible com funciona un Scrabble real. Hem volgut que el codi fos entenedor, fàcil de modificar i ampliable en cas que en el futur vulguem afegir més funcionalitats o millorar la intel·ligència del bot.

Per representar les fitxes, hem creat la classe Fitxa, que conté la informació essencial de cada peça del joc: la lletra (o dígraf, com ara "NY") i la puntuació que li correspon. Aquesta classe és simple però imprescindible, ja que les fitxes són els elements que es col·loquen al taulell i que acaben definint les jugades i els punts.

El taulell del nostre Scrabble està format per una graella de caselles, com en la versió clàssica del joc. Ara bé, no totes les caselles són iguals: algunes tenen efectes especials que afecten la puntuació, com per exemple multiplicar el valor d'una lletra o d'una paraula. Per gestionar aquesta varietat de comportaments de manera clara i flexible, hem fet servir una interfície anomenada Multiplicador.

La interfície EstratègiaMultiplicador defineix el comportament comú que poden tenir totes les caselles que modifiquen la puntuació. Les diferents classes de casella implementen aquesta interfície segons l'efecte que han de tenir. Així doncs, hem creat tres tipus concrets de caselles: CasellaNormal, MultiplicadorLletra i MultiplicadorParaula.

La classe CasellaNormal representa les caselles estàndard, que no apliquen cap modificador, i per tant implementen EstratègiaMultiplicador retornant sempre un valor neutre (és a dir, multiplicador 1). Les caselles de tipus MultiplicadorLletra implementen la interfície per retornar el factor pel qual s'ha de multiplicar la puntuació de la lletra col·locada (per exemple, x2 o x3). De manera similar, les caselles de tipus MultiplicadorParaula apliquen un multiplicador a tota la paraula formada quan una de les seves fitxes cau en aquesta casella.

Un aspecte a tenir en compte és que si la casella és de tipus multiplicador de paraula, aquest efecte no s'aplica en el mètode calcularPunts(), ja que no té sentit considerar-lo de manera aïllada. El multiplicador de paraula només entra en joc quan s'ha completat tota una paraula: és en aquest moment que es calcula la puntuació total sumant les aportacions de cada casella i, si alguna d'aquestes és un multiplicador de paraula, s'hi aplica el factor corresponent al resultat global.

El fet que cada casella tingui una sola estratègia ben definida ens ha permès mantenir el codi net, fàcil de llegir i molt extensible. En lloc de tenir una lògica complexa dins d'una única classe amb molts condicionals, hem separat el comportament en diferents classes que comparteixen una interfície comuna. Aquesta estratègia s'ha demostrat molt útil a l'hora de calcular la puntuació de les jugades, ja que només cal demanar a la casella quin multiplicador aplica segons el tipus que és.

Aquest enfocament també ens facilita molt les possibles ampliacions del joc. Si en el futur volguéssim afegir nous tipus de casella amb comportaments més avançats, només caldria crear una nova classe que implementi la interfície Multiplicador amb la lògica corresponent.

Els Jugadors són una part clau del joc, i per això tenen la seva pròpia classe. Cada jugador té un nom, una puntuació i un faristol, que és el conjunt de fitxes que pot utilitzar en cada torn. El jugador ha de poder veure les seves fitxes, col·locar paraules, passar torns o intercanviar fitxes. Per representar el faristol hem fet una classe pròpia, que facilita molt la gestió de les fitxes del jugador.

Un cas particular de jugador és el Bot, que hem implementat com una subclasse de Jugador. Això ens permet reutilitzar bona part del codi del jugador humà, però redefinint certs comportaments, com ara la manera de decidir quina paraula col·locar. El bot pot tenir una lògica senzilla o més avançada segons l'atribut del nivell de dificultat.

Finalment, hi ha el Sac de fitxes, que conté totes les fitxes disponibles al joc. Aquest sac s'utilitza per repartir les fitxes inicials als jugadors i per omplir els faristols després de cada torn. La classe gestiona quantes fitxes queden, i retorna fitxes de manera aleatòria, tal com passa en el joc real. També té un conjut de Strings amb les lletres "originals" del sac. Això ens permet saber si una lletra o dígraf pertany al sac o no. Per exemple, si el jugador inicialitza el sac en català quan vulgui intercanviar un comodí amb el dígraf CH se li prohibirà, ja que aquest no pertany a les lletres originals del sac.

Amb aquest conjunt de classes hem aconseguit una estructura clara i lògica que reflecteix molt bé les mecàniques del joc de Scrabble. Cada peça del joc està representada per un objecte, i el comportament es reparteix entre les classes de forma coherent. Això ens ha ajudat molt tant en el procés de desenvolupament com en les proves i l'extensió del codi.

La classe Estadístiques s'encarrega de gestionar les estadístiques globals del sistema de joc, com les puntuacions dels jugadors, el rànquing ordenat i la puntuació total acumulada. Aquesta classe segueix el patró de disseny Singleton, de manera que només existeix una única instància durant tota l'execució del programa. Aquesta decisió es justifica pel fet que les estadístiques han de ser globals i consistents: no tindria sentit tenir estadístiques duplicades o disperses, i el Singleton permet accedir fàcilment a la instància única des de qualsevol part del codi.

Atributs i mètodes de les classes

Nom	Fitxa
Descripció	Representa una fitxa del joc, amb una lletra, un valor en punts i un indicador de si és un dígraf.
Atributs	-String lletra

	-int punts -boolean dígraf
Mètodes	+Fitxa(String lletra, int punts) : Fitxa Crea una nova fitxa amb la lletra i els punts indicats. +Fitxa(Fitxa copiaFitxa) : Fitxa Inicialitza una fitxa com a còpia d'una altra. +esDigraf(): boolean Indica si la fitxa representa un dígraf. +setLletraComodi(String lletra) Assigna una nova lletra a una fitxa comodí. +esComodi() : boolean Comprova si una fitxa és un comodí. Un comodí és una fitxa amb un valor de punts igual a 0.

Nom	Sac
Descripció	Representa el sac de fitxes del joc. Conté totes les fitxes que encara no s'han utilitzat i la quantitat disponible de cadascuna. El sac permet extreure fitxes de manera aleatòria i controlar quantes queden.
Atributs	-Multiset<Fitxa> fitxes -Set<String> lletresOriginals;
Mètodes	+Sac() : Sac Crea un nou sac buit de fitxes. +Sac(Sac copiaSac) : Sac Inicialitza un sac com a còpia d'un altre. +obtenirLletresOriginals(): Set<String> Retorna les lletres inicials del sac. +setFitxesOriginals() Inicialitza les lletres originals del sac. +esFitxaOriginal(String lletra): boolean Comprova si una lletra és un fitxa original. +afegirFitxa(Fitxa f) Afegeix una fitxa al sac. Si la fitxa ja hi és, incrementa la seva quantitat. +agafarFitxa() : Fitxa Extreu una fitxa aleatòria del sac. Un cop seleccionada, la seva quantitat es redueix en una unitat. ExcepcioSacBuit +agafarFitxa(Fitxa fitxa) : Fitxa Extreu una instància concreta d'una fitxa del sac. Redueix en una unitat la seva quantitat disponible. ExcepcioSacNoConteLaFitxa +obtenirNumFitxes() : int Retorna el nombre total de fitxes disponibles al sac, sumant totes les quantitats de cada tipus de fitxa. +obtenirNumFitxes(Fitxa fitxa) : int Retorna la quantitat disponible d'una fitxa concreta al sac. +esBuit() : boolean Comprova si el sac està buit. +obtenirSac(): Multiset<Fitxes> Retorna el contingut intern del sac.

Nom	Casella
Descripció	Representa una casella del tauler de joc. Cada casella té una posició definida per coordenades (x,y), pot contenir una fitxa i té associada una estratègia de puntuació.
Atributs	-int x -int y -Fitxa fitxa -boolean casellaJugada -EstrategiaPuntuació estrategia
Mètodes	+Casella(int x, int y, int size) Crea una nova casella amb els paràmetres indicats. +Casella(Casella copia) Inicialitza una casella com a còpia d'una altra. +calcularPunts() : int Calcula els punts que aporta la fitxa en aquesta casella segons l'estratègia assignada. +jugarCasella() Marca la casella com a jugada. +esBuida() : booelan Comprova si la casella està buida (no té fitxa). +colocarFitxa(Fitxa fitxa) Col·loca una fitxa a la casella. ExcepcioCasellaOcupada si la casella ja conté una fitxa. +retirarFitxa() Retira la fitxa de la casella. ExcepcioCasellaBuida -assignarEstratègia(int i, int j, int size): EstrategiaPuntuacio Assigna una estratègia de puntuació a la casella indicada segons la seva posició

Nom	Taulell
Descripció	Representa el taulell de joc de Scrabble com una matriu de caselles. Cada casella pot contenir una fitxa i tenir una estratègia de puntuació associada.
Atributs	-int size -Casella[][] taulell
Mètodes	+Taulell(int size) Crea un nou taulell amb la mida especificada +Taulell(Taulell copiaTaulell)

	Inicialtiza aquest taulell com a copia d'un altre. +colocarFitxa(Fitxa fitxa, int fila, int columna) Mètode per assignar una fitxa a una casella del taulell. +retirarFitxa(int fila, int columna) Mètode per desassignar un fitxa d'una casella del taulell. +esBuit(): boolean Comprova si totes les caselles del taulell estan buides.

Nom	ValidadorJugada
Descripció	S'encarrega de validar si una jugada és correcta segons les regles de Scrabble. Revisa la posició de les fitxes, la validesa de la paraula principal i les paraules perpendiculars.
Mètodes	+esJugadaHoritzontal(List<Casella> caselles) : boolean Identifica si la paraula s'està jugant en direcció horitzontal. +casellaPertanyAJugada(int x, int y, List<Casella> jugada) : Casella Determina si una casella forma part de la jugada. Retorna la casella corresponent o null. +posicioValida(List<Casella> casellesJugades, Taulell taulell) : boolean Verifica si les caselles jugades estan alineades correctament i compleixen amb les regles del Scrabble. +construirParaula(List<Casella> casellesJugades, boolean horitzontal, Taulell taulell) : String Construeix la paraula principal utilitzant les fitxes jugades i el taulell, en la direcció indicada. +validarParaulesPerpendiculars(List<Casella> casellesJugades, DAWG dawg, boolean horitzontal, Taulell taulell) : boolean Valida si les paraules perpendiculars generades per la jugada són vàlides segons el diccionari DAWG. +validarJugada(List<Casella> casellesJugades, DAWG dawg, Taulell taulell) : boolean Verifica que la jugada completa sigui vàlida: posició, paraula principal i paraules perpendiculars.

Nom	CalculadorPuntuacio
Descripció	S'encarrega de calcular la puntuació d'una jugada al joc de Scrabble. Calcula la puntuació de la paraula principal i de les paraules perpendiculars formades.

Mètodes	+calcularPuntuacioJugada(List<Casella> casellesJugades, boolean horitzontal, Taulell taulell) : int Calcula la puntuació total d'una jugada (paraula principal + paraules perpendiculars). -calcularPuntuacioParaulaPrincipal(List<Casella> casellesJugades, boolean horitzontal, Taulell taulell) : int Calcula la puntuació de la paraula principal formada durant la jugada, aplicant els multiplicadors corresponents. -calcularPuntuacioPerpendiculars(List<Casella> casellesJugades, boolean horitzontal, Taulell taulell) : int Calcula la puntuació total de les paraules perpendiculars formades durant la jugada, només si tenen més d'una lletra.

Nom	Jugador
Descripció	Representa un jugador de la partida. Conté el nom del jugador, la seva puntuació acumulada i el seu faristol amb fitxes disponibles.
Atributs	-String nom -int punts -Faristol faristol
Mètodes	+Jugador(String nom, Faristol faristol) Crea un nou jugador amb el nom especificat i un faristol assignat. +Jugador(Jugador copiaJugador) Inicialitza un jugador com a còpia d'un altre. +faristolPle() : boolean Comprova si el faristol del jugador està ple. +barrejarFaristol() Barreja aleatòriament les fitxes del faristol del jugador. +afegirPunts(int nousPunts) Afegeix punts a la puntuació del jugador. +eliminarPunts(int nousPunts) Resta punts de la puntuació del jugador. +afegirFitxa(Fitxa fitxa) Afegeix una fitxa al faristol del jugador. +eliminarFitxa() Elimina una fitxa del faristol del jugador. +esBot() : boolean Indica si el jugador és un bot. Per defecte, retorna fals.

Nom	Faristol
Descripció	Representa el faristol d'un jugador, on es col·loquen les fitxes disponibles durant la partida. El faristol té una mida fixa i pot contenir diverses fitxes que el jugador pot utilitzar.
Atributs	-int size -ArrayList<Fitxa> fitxes
Mètodes	+Faristol(int size) Crea un nou faristol amb una mida especificada. +Faristol(Faristol copiaFaristol) Inicialitza aquest faristol com la copia d'un altre. +afegirFitxa(Fitxa fitxa) Afegeix una fitxa al faristol. ExcepcioFaristolPle si el faristol ja ha arribat a la seva capacitat màxima +eliminarFitxa(Fitxa fitxa) Elimina una fitxa concreta del faristol. ExcepcioFaristolNoConteLaFitxa si la fitxa no és present al faristol +barrejarFitxes() Barreja aleatòriament les fitxes del faristol. +esPle() : boolean Comprova si el faristol està ple +esBuit() : boolean Comprova si el faristol és buit

Nom	Bot
Descripció	Jugador però controlat per la màquina seguint un algorisme. Hereta de Jugador i té un nivell de dificultat del 1 al 3.
Atributs	-int nivellDificultat
Mètodes	+Bot(String nom, Faristol faristol, int nivellDificultat) Crea un bot amb els paràmetres indicats. +Bot(Bot copiaBot) Inicialitza aquest bot com a còpia d'un altre. +esBot() : boolean Retorna vertader.

Nom	Jugada
Descripció	Classe que representa una jugada al joc de Scrabble. Emmagatzema la paraula formada, les caselles on s'han col·locat fitxes noves i la puntuació total.
Atributs	-String paraulaFormada -List<Casella> casellesJugades -int puntuacio -boolean jugadaValida -boolean comodi
Mètodes	+Jugada(String paraulaFormada, List<Casella> casellesJugades, int puntuacio, boolean jugadaValida) Constructor explícit amb tots els valors coneguts (útil per desar o clonar jugades). +Jugada(List<Casella> casellesJugades, DAWG dawg, Taulell taulell) Constructor per jugades d'usuari. Valida la jugada, calcula la puntuació i determina la paraula formada. +Jugada(String paraulaFormada, List<Casella> casellesJugades, DAWG dawg, Taulell taulell, boolean horitzontal) Constructor per jugades d'un bot. Valida la posició i les paraules perpendiculars, i calcula la puntuació. +conteComodi() : boolean Retorna si la jugada conté almenys un comodí. -private conteComodi(List<Casella> casellesJugades) : boolean Mètode auxiliar per detectar si alguna de les fitxes jugades és un comodí.

Nom	Torn
Descripció	Representa un torn dins d'una partida de Scrabble. Conté la informació necessària per gestionar el torn actual, incloent el sac de fitxes, el taulell, els jugadors, el número de torn i si el torn ha acabat.
Atributs	-Sac sac -Taulell taulell -Jugador[] jugadors -int numTorn -boolean acabada -int tornsSenseCanvis -String idioma
Mètodes	+Torn(Sac sac, Taulell taulell, Jugador[] jugadors, int torn, boolean acabada, int tornsSenseCanvis, String idioma) Crea un nou torn amb els paràmetres indicats. +jugadorBot(int torn) : boolean Retorna *true* si el jugador del torn és un bot.

	+esAcabada() : boolean Indica si és l'últim torn de la partida +getIdioma() : String

Nom	HistorialJoc
Descripció	Representa l'historial d'una partida de joc. Emmagatzema la llista de torns realitzats durant la partida, així com la data en què es va jugar.
Atributs	-List<Torn> torns -Date dataJoc
Mètodes	+HistorialJoc(Date date) Crea un nou historial de joc amb la data especificada. +afegirTorn(Torn torn) Afegeix un torn a l'historial de joc +retirarTorn() Elimina l'últim torn afegit a l'historial de joc. +retirarTorns(int torn) Elimina els torns posteriors a un torn especific. +obtenirTorn(int index) : Torn Retorna el torn corresponent a una posició concreta de l'historial. La numeració dels torns comença per 1 (no per 0). +obtenirMida() : int Retorna el nombre total de torns registrats a l'historial.

Nom	EstrategiaPuntuacio [INTERFÍCIE]
Descripció	Interfície que defineix una estratègia de puntuació per a una casella del tauler. Les estratègies poden aplicar multiplicadors a lletres o paraules senceres, segons el tipus de casella.
Mètodes	+calcularPunts() : int Calcula els punts aportats per una fitxa segons l'estratègia de puntuació. +esMultiplicadorParaula() : boolean Indica si l'estratègia és de multiplicador de paraula.

Nom	EstrategiaNormal/ EstrategiaMultiplicadorLletra / EstrategiaMultiplicadorParaula
Descripció	Classes que implementen EstrategiaPuntuacio Estratègia normal (sense multiplicadors)

	Multiplicador de lletra (multiplica el valor de la lletra) Multiplicador de paraula (multiplica el valor de la paraula)
Atributs	-int multiplicador
Mètodes	+calcularPunts() : int Calcula els punts aportats per una fitxa segons l'estratègia de puntuació. +esMultiplicadorParaula() : boolean Indica si l'estratègia és de multiplicador de paraula.

Nom	Estadistiques [SINGLETON]
Descripció	Classe que gestiona les estadístiques d'una partida o conjunt de partides. Permet emmagatzemar les puntuacions dels jugadors, així com calcular la puntuació mínima, la puntuació total i la mitjana.
Atributs	- Map<String, Integer> puntuacions - TreeMap<Integer, Set<String>> ranking - int puntuacioTotal
Mètodes	+Estadistiques() Constructor per defecte que inicialitza les estadístiques. Estableix la puntuació total i mitjana a 0, i la mínima al valor màxim d'enter. Crea una cua de prioritat per emmagatzemar les puntuacions, ordenades de major a menor valor. +afegirPuntuacio(int puntuacio, String jugador) Afegeix una puntuació per a un jugador determinat. Si la puntuació és vàlida (no negativa), s'actualitzen la puntuació total, la mínima i s'afegeix la nova entrada a la cua de prioritats. +obtenirMillor() : Map.Entry<String, Integer> Retorna el jugador amb la puntuació més alta. +obtenirPitjor() : Map.Entry<String, Integer> Retorna el jugador amb la puntuació més baixa. +obtenirMitja() : float Calcula la puntuació mitjana dels jugadors i l'emmagatzema. La mitjana es calcula dividint la puntuació total entre el nombre de jugadors. Si no hi ha cap puntuació registrada, no es fa cap càlcul. +retirarPuntuacio(String jugador, int punts) Retira una quantitat de punts a un jugador i actualitza les estadístiques globals. Si la nova puntuació es menor o igual a 0, s'elimina al jugador del registre. +eliminarJugador(String jugador) Elimina un determinat jugador de totes les estadístiques. +carregarDes(Estadistiques altres) : void Substitueix les dades actuals per les d'un altre objecte `Estadistiques`.

Nom	Node
Descripció	És una classe interna de DAWG i representa cada un dels punts del graf. Cada node pot contenir diversos arcs sortints cap a altres nodes, identificats per tokens (lletres o dígrafs) i marca si és el final o no d'algun prefix.
Atributs	-Map<String, Node> fills -boolean esFinal
Mètodes	+getFill(String token) : Node +getFills() : Map<String, Node> +conteToken(String token) : boolean

Nom	DAWG
Descripció	DAWG (Directed Acyclic Word Graph) que representa un diccionari de paraules a on cada node té un conjunt de transicions etiquetades amb tokens (caràcters o dígrafs) cap a altres nodes tot formant paraules.
Atributs	-Node arrel -Map<Node, Node> nodesMinimitzats -List<String> tokens -List<Node> pathAnterior
Mètodes	+DAWG(List<String> tokens, List<String> paraules) Constructor. Crea i construeix el DAWG a partir de tokens i paraules donades. -afegirParaula(String paraula, String paraulaAnterior) Afegeix una nova paraula al DAWG comparant-la amb la paraula anterior per detectar el prefix comú i reutilitzar-lo. En cas que sigui diferent el prefix crearà una nova ruta amb un node diferent +conteParaula(String paraula) : boolean Comprova si una paraula completa existeix dins del DAWG -minimitzarSufix(int desDe) Minimitza els nodes de la ruta afegida recentment (pathAnterior) començant des de la posició 'desDe'. Aquest mètode substitueix subgràfics equivalents per versions ja existents usant el mapa de nodes minimitzats. Això assegura que el DAWG sigui minimal, reutilitzant subestructures comunes. -tokenitzar(String paraula) : List<String> Divideix una paraula en una llista de tokens (Caràcters/Dígrafs) segons els tokens càrregats al DAWG. +esPrefix(String prefix) : Node Comprova si un prefix donat existeix en l'estructura DAWG i retorna el node corresponent al final del prefix si existeix.

Controladors de domini

Controlador Domini

Descripció

Classe principal del domini que actua com a controlador general per gestionar les partides, els bots i les estadístiques del Scrabble. Coordina els tres controladors principals (CtrlEstadistica, CtrlPartida, CtrlJugadaBot) i ofereix els mètodes necessaris per a poder jugar una partida.

Atributs

- **ctrlEstadistica** (CtrlEstadistica): Controlador que gestiona totes les estadístiques del joc.
- **ctrlPartida** (CtrlPartida): Controlador que gestiona tot el funcionament d'una partida.
- **ctrlJugadaBot** (CtrlJugadaBot): Controlador que gestiona les jugades que efectuarà un bot.

Mètodes

- **obtenirPuntuacioMitjana**(): Retorna la puntuació mitjana de tots els jugadors registrats.
- **obtenirPuntuacioTotal**(): Retorna la suma de les puntuacions de tots els jugadors.
- **obtenirPuntuacioMaxima**(): Retorna la puntuació més alta registrada.
- **obtenirPuntuacioMinima**(): Retorna la puntuació més baixa registrada.
- **obtenirPuntuacions**(): Retorna un Mapa amb les puntuacions i els noms de tots els jugadors.
- **inicialitzarPartida**(int midaTaulell, int midaFaristol, String idioma, String[] nomsJugadors, int[] dificultatsBots): Inicialitza una nova partida amb els paràmetres indicats.
- **carregarPartida**(Torn torn): Carrega una partida a partir d'un estat de torn guardat. D'aquesta manera es pot reprendre una partida ja iniciada.
- **obtenirTorn**(): Retorna el número del torn actual.
- **obtenirTaulell**(): Retorna el taulell del joc actual.
- **obtenirJugadors**(): Retorna els jugadors participants de la partida.
- **obtenirJugadorActual**(): Retorna el jugador que ha de jugar el torn acutal.
- **obtenirSac**(): Retorna el sac de fitxes actual.
- **colocarFitxa**(String lletra, int i, int j): Coloca una fitxa del faristol del jugador actual en una posició indicada en els paràmetres.
- **retirarFitxa**(int i, int j): Retira la fitxa situada en la posició indicada.
- **ferUndo**(): Desfà l'última acció realitzada.
- **acabarPartida**(): Acaba la partida actual.
- **passarTorn**(): Passa el torn al següent jugador.
- **canviarFitxes**(String[] fitxesCanviades): Canvia un conjunt de fitxes del faristol del jugador actual per noves del sac.
- **commitParaula**(): Confirma la paraula formada en el taulell.
- **partidaAcabada**(): Comprova si la partida ha acabat.
- **esFinalDePartida**(): Comprova si es el final de la partida.
- **jugadaBot**(): Realitza la jugada del bot.
- **resetTorn**(): Es torna a l'estat inicial del torn.

- **setLletraComodi**(Fitxa fitxa, String lletra): Canvia la lletra comodí per la indicada en el paràmetre lletra.
- **getInstance**(): Retorna la instància singleton del controlador
- **afegirPuntuacio**(String jugador, int puntuacio): Afegeix una puntuació a les estadístiques
- **retirarPuntuacio**(String jugador, int puntuacio): Retira una puntuació de les estadístiques
- **obtenirMillorJugador**(): Retorna el millor jugador segons les estadístiques
- **eliminarJugador**(String nom): Elimina un jugador de les estadístiques
- **actualitzarEstadistiques**(Jugador[] jugadors): Actualitza les estadístiques amb els jugadors de la partida
- **esPotFerUndo**(): Comprova si es pot fer undo
- **guardarPartida**(): Guarda l'estat actual de la partida
- **carregarPartida**(String nomFitxer): Carrega una partida des d'un fitxer (diferent signatura que al document)
- **generarNomFitxerAmbData**(): Genera un nom de fitxer amb timestamp (mètode privat)
- **llistarPartidesGuardades**(): Retorna la llista de partides guardades
- **desarEstadistiques**(): Guarda les estadístiques
- **carregarEstadistiques**(): Carrega les estadístiques

Controlador Partida

Descripció

Controlador principal encarregat de gestionar el desenvolupament d'una partida de Scrabble. S'encarrega de la creació i inicialització del taulell, el sac de fitxes, el conjunt de jugadors (incloent bots), l'historial de jugades i el diccionari (DAWG). També gestiona els torns, controla l'estat del joc, permet desfer accions, i determina quan la partida ha d'acabar.

Atributs

- **instance** (CtrlPartida): Instància singleton del controlador de partida.
- **ctrlBot** (CtrlJugadaBot): Controlador encarregat de gestionar les jugades dels bots.
- **historial** (HistorialJoc): Historial de tots els torns jugats durant la partida.
- **sac** (Sac)**:** Sac de fitxes utilitzat per distribuir lletres als jugadors.
- **taulell** (Taulell): Taulell on es desenvolupen les jugades.
- **jugadors** (Jugador[]): Array de jugadors que participen en la partida.
- **dawg** (DAWG): Diccionari per validar paraules segons l'idioma.
- **acabada** (boolean): Indica si la partida ha finalitzat.
- **torn** (int): Número de torn actual.
- **casellesTorn** (List<Casella>): Caselles modificades durant el torn actual.
- **jugadaActual** (Jugada): Jugada en curs del torn actual.
- **tornsSenseCanvi** (int): Comptador de torns consecutius sense jugades que modifiquin el taulell.
- **idioma** (String): Emmagatzema l'idioma de la partida actual

Mètodes

- **getInstance**(): Retorna la instància singleton del controlador de partida.

- **inicialitzarPartida**(int midaTaulell, int midaFaristol, String idioma, String[] nomsJugadors, int[] dificultatsBots): Inicialitza una nova partida de Scrabble configurant tots els components essencials: reinicia l'estat del joc, crea el taulell i el sac de fitxes segons l'idioma, carrega el diccionari, genera els jugadors (incloent bots), assigna els faristols, crea l'historial de jugades i estableix el primer torn.
- **inicialitzarTaulell**(int midaTaulell): Crea i inicialitza el taulell amb la mida indicada.
- **inicialitzarSac**(String idioma): Llegeix el fitxer corresponent a l'idioma i omple el sac amb les fitxes configurades.
- **inicialitzarJugadors**(String[] nomsJugadors, int[] dificultatsBots, int midaFaristol): Crea els jugadors (humans i bots) i assigna els seus faristols inicials.
- **inicialitzarCtrlBot**(): Inicialitza el controlador de jugades dels bots.
- **inicialitzarDawg**(String idioma): Llegeix i carrega les paraules vàlides i símbols de l´idioma seleccionat per la generació del DAWG.
- **inicialitzarFaristol**(Jugador jugador): Omple el faristol del jugador amb fitxes del sac fins que estigui ple.
- **recuperarTorn**(Torn nouTorn): Recupera l'estat del joc a partir d'un torn prèviament guardat.
- **resetTorn**(): Restaura l'estat del torn actual a com estava a l'inici.
- **acabarPartida**(): Finalitza la partida i ordena els jugadors segons la seva puntuació.
- **passarTorn**(): Passa el torn al jugador següent i actualitza l'historial.
- **undo**(): Desfà l'últim torn fet per un jugador humà, si és possible.
- **esPotFerUndo**(): Comprova si es pot fer undo segons l'estat de la partida i el tipus de jugadors.
- **esFinalDePartida**(): Determina si es compleixen les condicions per finalitzar la partida.
- **acabada**(): Retorna si la partida ha finalitzat o no.
- **obtenirSac**(): Retorna el sac de fitxes actual.
- **obtenirTaulell**(): Retorna el taulell actual.
- **obtenirJugadors**(): Retorna els jugadors participants.
- **obtenirJugadorActual**(): Retorna el jugador al qual li toca jugar.
- **obtenirTorn**(): Retorna el número de torn actual.
- **agafarDelSac**(): Agafa una fitxa del sac i l'afegeix al faristol del jugador en curs.
- **ordenarJugadors**(): Ordena els jugadors de la partida en funció de la seva puntuació, de major a menor.
- **setLletraComodi**(Fitxa fitxa, String lletraComodi): Permet al jugador canviar la lletra d'una fitxa comodí per la que hagi escollit, sempre que sigui vàlida.
- **canviarFitxes**(String[] fitxesCanviades): Canvia les fitxes que el jugador ha escollit per noves fitxes aleatòries del sac, i passa el torn al següent jugador.
- **colocarFitxa**(String fitxa, int fila, int columna): Coloca una fitxa en una casella del taulell i registra la jugada.
- **retirarFitxa**(int fila, int columna): Retira una fitxa de la casella especificada del taulell, la retorna al faristol del jugador i registra la jugada .
- **rellenarFaristol**(): Reomple el faristol del jugador actual amb fitxes del sac fins que torni a estar ple o no quedin més fitxes disponibles.

- **commitParaula**(): Registra la jugada com a vàlida, suma els punts al jugador (incloent-hi 50 punts extres si ha utilitzat totes les fitxes), reomple el faristol, reinicia el comptador de torns sense canvi i avança al següent torn.
- **jugadaBot**(): Executa la jugada del bot en funció del seu nivell de dificultat, col·loca les fitxes al taulell, les elimina del faristol, i registra la jugada com a vàlida.
- **inicialitzarCasellasTorn**(): Inicialitza l'estructura que guarda les caselles que s'utilitzaran en el torn (tot i que és privat al codi, apareix com a funcionalitat important)
- **obtenirTornActual**(): Retorna el torn actual de l'historial
- **recuperarTornPersi**(Torn nouTorn): Versió alternativa per recuperar un torn sense modificar l'historial
- **recuperarTornDesDeFitxer**(Torn torn): Recupera una partida des d'un fitxer, reinicialitzant els components necessaris
- **passarTornIntern**(): Mètode intern que gestiona el pas de torn (diferent del públic passarTorn())

Controlador JugadaBot

Descripció

El controlador JugadaBot és l'encarregat de generar automàticament les jugades del jugador bot en el joc de Scrabble. El seu objectiu és calcular la millor/intermèdia/pitjor jugada possible a partir de l'estat actual del taulell i les fitxes disponibles d'un bot, tenint en compte el nivell de dificultat establert.

L'algorisme implementat es basa en backtracking amb poda mitjançant l'estructura DAWG. Per més informació vegeu algorisme Bot.

Mètodes

- **calcularJugada**(Taulell taulell, DAWG dawg, int nivellDificultat, List<Fitxa> fitxes)
  Calcula i retorna una jugada vàlida segons el nivell de dificultat.

- **generarJugades**(Taulell taulell, DAWG dawg, List<Fitxa> fitxes)
  Genera totes les jugades vàlides possibles des de qualsevol posició del taulell.

- **generarParaules**(String prefix, List<Fitxa> fitxes, List<Casella> caselles, DAWG.Node node, int fila, int col, boolean horitzontal, Taulell taulell, DAWG dawg)
  Genera recursivament paraules vàlides a partir d'un prefix.

- **processarFitxaIContinuar**(String prefix, Fitxa fitxa, List<Fitxa> fitxes, List<Casella> caselles, DAWG.Node node, int fila, int col, boolean horitzontal, Taulell taulell, DAWG dawg, List<Jugada> resultats)
  Afegeix una fitxa i continua generant paraules amb recursió.

Controlador Estadístiques

Descripció
Aquest és el controlador encarregat de gestionar les operacions relacionades amb les estadístiques del sistema. Permet donar d'alta, eliminar i consultar jugadors i les seves respectives puntuacions, així com consultar estadístiques globals. Tot això es fa mitjançant el maneig dels atributs de la classe Estadistiques.

Atributs

- **estadistiques**(Estadistiques): Objecte que encapsula i gestiona les estadístiques del sistema.
- **instance** (CtrEstadistica): Instància singleton del controlador. Permet garantir que només hi hagi una instància de CtrEstadistica.

Mètodes

- **afegirPuntuacio**(String jugador, int puntuacio): Afegeix una puntuació al jugador especificat i actualitza les estadístiques globals.
- **retirarPuntuacio**(String jugador, int punts): Resta una quantitat de punts a un jugador i actualitza les estadístiques globals.
- **eliminarJugador**(String jugador): Elimina totes les estadístiques associades al jugador especificat.
- **obtenirMillorJugador**(): Retorna el nom del jugador amb la puntuació més alta.
- **obtenirPitjorJugador**(): Retorna el nom del jugador amb la puntuació més baixa.
- **obtenirPuntuacioTotal**(): Retorna la suma de les puntuacions de tots els jugadors.
- **obtenirPuntuacioMaxima**(): Retorna la puntuació més alta registrada.
- **obtenirPuntuacioMinima**(): Retorna la puntuació més baixa registrada.
- **obtenirPuntuacions**(): Retorna un Mapa amb les puntuacions i els noms de tots els jugadors.
- **getInstance**(): Retorna la instància única del controlador. S'utilitza per aplicar el patró Singleton.
- **desarEstadistiques**(): Desa les estadístiques actuals a disc utilitzant el CtrlPersistencia.
- **carregarEstadistiques**(): Carrega les estadístiques des del disc utilitzant el CtrlPersistencia.
- **obtenirPuntuacioMitjana**(): Retorna la mitjana de les puntuacions de tots els jugadors.

Estructures de dades i algoritmes

Estructures de les classes

Sac com a Multiset<Fitxa>

Per representar el sac de fitxes, hem optat per utilitzar un Multiset<Fitxa>, una estructura que ens permet tenir múltiples còpies de la mateixa fitxa i fer operacions sobre elles de manera eficient i natural. Aquesta tria és coherent tant amb la naturalesa del joc com amb els costos i el tipus d'operacions que hi fem habitualment.

En primer lloc, cal tenir en compte que el sac no és un conjunt de fitxes úniques, sinó que conté diverses còpies d'una mateixa lletra amb puntuacions associades. Un Multiset ens permet modelar això de forma directa, ja que internament manté un recompte de quantes vegades apareix cada element.

Pel que fa als costos, amb un Multiset, afegir o treure fitxes és una operació d'ordre O(1) en la mitjana, igual que consultar la quantitat d'un element concret. A més, iterar sobre totes les fitxes (tenint en compte les repeticions) és lineal respecte al total d'elements, no només al nombre de claus diferents.

Si en lloc d'un Multiset haguéssim utilitzat un Map<Fitxa, Integer>, també podríem haver representat la quantitat de cada fitxa, però la gestió hauria estat més feixuga i menys intuïtiva. Tot i que operacions com afegir o consultar una fitxa concreta serien igualment ràpides (O(1) en mitjana amb un HashMap), eliminar una fitxa o reduir-ne el comptador implicaria escriure més lògica manual. A més, hauríem d'estar pendents de quan una fitxa arriba a zero per eliminar-la explícitament del mapa.

Per tant, el Multiset és no només més natural des del punt de vista conceptual, sinó també més eficient i més senzill de gestionar pel tipus d'operacions que fem: seleccions aleatòries, insercions, eliminacions i consultes sobre quantes fitxes queden.

Taulell com a matriu de Caselles

La representació del taulell com una matriu de caselles és una elecció molt natural i coherent, tant des del punt de vista conceptual com pel que fa a l'eficiència i simplicitat del codi.

D'entrada, el taulell del Scrabble té una estructura clarament bidimensional i de mida fixa on cada posició és una casella que pot contenir o no una fitxa, i que pot tenir propietats especials (com multiplicadors de lletra o de paraula). Utilitzar una matriu ens permet reflectir fidelment aquesta organització: cada casella queda identificada per unes coordenades (fila, columna) que corresponen de manera directa a la seva posició real al taulell.

Des del punt de vista d'implementació, accedir a una casella concreta amb una matriu és extremadament eficient: tenim accés constant O(1) a qualsevol posició, la qual cosa és important quan es vol comprovar ràpidament si una casella està buida, aplicar una jugada, o calcular la

puntuació d'una paraula col·locada al taulell. Aquest accés directe també facilita la validació de moviments, la cerca de paraules formades, i la visualització general del taulell.

A més, com que la mida del taulell és coneguda i fixa, no cal cap estructura dinàmica ni cap optimització especial per gestionar l'espai. La matriu cobreix totes les necessitats de manera simple i clara, i facilita molt la comprensió del codi i el seu manteniment.

Estadísiques com a Map i TreeMap

Pel que fa al cost computacional de les operacions més habituals, afegir o retirar puntuació d'un jugador té un cost de O(log n), degut al fet que cal modificar el TreeMap que manté l'ordre del rànquing. Consultar la puntuació mitjana o accedir a la millor o pitjor puntuació és molt eficient, ja que es pot fer en temps constant o logarítmic segons el cas. Aquesta gestió eficient i ordenada de les estadístiques, facilita el manteniment del codi i assegura una bona escalabilitat si el nombre de jugadors augmenta.

La separació entre dues estructures —el HashMap<String, Integer> puntuacions i el TreeMap<Integer, Set<String>> ranking— és totalment intencionada i té diversos motius importants, tant pel rendiment com per la claredat del codi.

D'una banda, el HashMap permet consultar ràpidament la puntuació d'un jugador concret a partir del seu nom. Aquesta operació és molt freqüent (per exemple, quan vols afegir punts o restar-ne a un jugador concret), i per això interessa que sigui molt eficient: el HashMap ofereix accés en temps constant, O(1) de mitjana.

D'altra banda, el TreeMap manté el rànquing ordenat de puntuacions. És a dir, ens permet saber en tot moment quin jugador (o jugadors) tenen la puntuació més alta o més baixa, sense haver de recórrer tota la col·lecció. El TreeMap, en ordenar automàticament les claus, permet accedir directament al millor (firstEntry()) o pitjor (lastEntry()) en temps logarítmic, O(log n).

També es manté una variable sencera (int puntuacioTotal) per acumular la suma de totes les puntuacions. Això evita haver de recórrer el mapa sencer cada vegada que es vol obtenir la mitjana, cosa que milloraria el rendiment especialment en sistemes amb molts jugadors.

Quan s'afegeix una puntuació amb el mètode afegirPuntuacio, si el jugador ja existeix, primer s'elimina la seva puntuació antiga del rànquing i es resta de la puntuació total. A continuació, es calcula la nova puntuació, s'actualitza al HashMap i es torna a inserir al TreeMap segons la seva nova puntuació. Si el jugador no existia, simplement s'afegeix amb la seva puntuació inicial a ambdues estructures i s'actualitza la puntuació total. Així, les dues estructures es mantenen sincronitzades.

Quan s'elimina un jugador amb eliminarJugador, es comprova si existeix; si és així, es treu del HashMap, es resta la seva puntuació de la total i també s'elimina del conjunt corresponent del TreeMap. Si aquell conjunt de jugadors queda buit, es retira també la clau corresponent. Tot plegat assegura que les estadístiques siguin precises i que no quedin dades residuals.

Estructura DAWG

El DAWG (Directed Acyclic Word Graph) és una estructura de graf dirigit acíclic dissenyada per representar un conjunt de paraules vàlides de manera eficient. En aquest projecte, hem dividit la seva implementació en dues parts principals: la classe externa DAWG, que gestiona la construcció i navegació del graf, i la classe interna Node, que representa els estats individuals dins d'aquest graf.

Classe Node:

Cada instància de Node representa un estat parcial d'una paraula. Conté les següents estructures:

- **Map<String, Node> fills**
  Aquest mapa defineix les transicions sortints des del node actual. Cada clau és un token (pot ser una lletra o un dígraf), i el valor associat és el node fill corresponent. Hem utilitzat un HashMap ja que ens cal garantir un accés ràpid (O(1) de mitjana) a les transicions, ja que cal consultar amb freqüència si existeix una connexió concreta a partir d'un token determinat.
- **boolean esFinal**
  Indica si el camí des de l'arrel fins aquest node forma una paraula vàlida completa del diccionari. Aquesta marca és fonamental per la validació de jugades.

DAWG:

La classe externa DAWG s'encarrega de construir i gestionar tot el graf. Les seves estructures principals són:

- **Node arrel**
  Representa el node inicial del graf. Totes les cerques i insercions comencen a partir d'aquest node.
- **Map<Node, Node> nodesMinimitzats**
  Aquest mapa s'utilitza durant la construcció del DAWG per detectar i reutilitzar subgràfics equivalents ja existents.
  Usem un Map (i no un Set) perquè, un cop detectat un node equivalent, cal recuperar-lo explícitament per substituir l'original. Un Set en canvi permetria saber si ja existeix, però no accedir-hi de manera directa.
- **List<String> tokens**
  Conté els tokens vàlids (lletres o dígrafs) per a la tokenització de paraules.
  Hem optat per una List en lloc d'un Set perquè l'ordre d'aquests tokens és rellevant. Durant la tokenització, s'intenta trobar el primer token que coincideixi amb l'inici de la cadena, i per això cal preservar un ordre personalitzat (per exemple, prioritzar "ny" abans que "n").
- **List<Node> pathAnterior**
  Guarda el camí complet (els nodes) corresponent a l'última paraula afegida al graf.
  Aquest camí és necessari per a la minimització de sufixos, ja que permet identificar el punt de divergència entre dues paraules consecutives i aplicar l'optimització només on cal.

Torn i HistorialJoc

La classe Torn modela l'estat complet d'una partida de Scrabble en un moment determinat. Emmagatzema còpies independents del sac de fitxes, el taulell, els jugadors i informació com el número del torn, si la partida ha acabat i el nombre de torns consecutius sense canvis. Això permet preservar l'estat exacte de la partida en aquell instant, evitant efectes col·laterals, i facilita operacions com desfer accions o analitzar el desenvolupament del joc. La classe també permet consultar si el jugador actual és un bot i accedir als elements clau del torn.

La classe HistorialJoc serveix per registrar l'evolució cronològica d'una partida mitjançant una llista de torns. A cada acció del jugador, es pot afegir un nou torn, eliminant-se o modificant-se posteriorment si cal. També inclou la data de la partida per a finalitats de registre o visualització. Aquesta estructura és essencial per recuperar l'estat del joc en un punt concret, permetent funcionalitats com desfer accions, revisar partides jugades o implementar modes d'anàlisi i repetició.

Jugada

La classe Jugada concentra tota la informació rellevant d'una acció al taulell en un únic objecte, evitant haver de passar múltiples paràmetres entre mètodes i facilitant tant la llegibilitat com el manteniment del codi.

Conté la paraula formada, per mostrar-la a l'usuari per pantalla, una llista amb les caselles jugades, la seva puntuació corresponent i finalment un booleà que indica si la jugada és vàlida.

Algorisme creació DAWG

Avantatges d'usar un DAWG

Per tal d'emmagatzemar totes les paraules vàlides d'un diccionari o temàtica, de forma compacta i eficient hem utilitzat un **DAWG** (Directed Acyclic Word Graph), tal i com marcava l'enunciat del projecte.

Un DAWG està format per nodes connectats mitjançant arcs dirigits, on cada camí des de l'arrel fins a un node final forma una paraula del diccionari. L'objectiu del DAWG és aprofitar aquelles paraules que comparteixen una part comuna perquè comparteixin nodes dins del graf i optimitzar tant l'espai com el temps de consulta.

A diferència d'un trie (arbre de caràcters, que forma paraules), un DAWG aplica un procés de minimització per detectar i reutilitzar subgràfics idèntics. Això permet evitar duplicació de sufixos, reduint dràsticament la memòria necessària i accelerant les operacions de cerca.

El DAWG també ens permet saber donat un prefix, totes aquelles lletres que al ajuntar-les amb el prefix formen nous prefixos de paraules vàlides, d'aquesta manera el bot en comptes de generar totes les combinacions possibles de lletres, pot navegar pel DAWG i només explorar aquells camins que porten a una paraula vàlida.

Construcció del DAWG

El procés de construcció es duu a terme de manera directa, és a dir, no es genera un trie per després minimitzar-lo, sinó que el graf es construeix ja minimitzat en temps real.

Per construir el DAWG tan sols necessitem:

- Una llista de **tokens** (caràcters o dígrafs vàlids).
- Una llista de **paraules vàlides**, **ordenades alfabèticament** com a precondició.

Per cada una de les paraules vàlides farem els següents passos:

1. **Tokenitzar la paraula:** Es divideix la paraula en tokens utilitzant la llista de dígrafs/caràcters.
    - Tokens: ["NY", "L·L", "A", "B", "C", ...]
    - Paraula: "NYANYO" → ["NY", "A", "NY", "O"]

    L'algorisme intenta emparellar primer els tokens més llargs, per assegurar que es prioritzen els dígrafs.

2. **Buscar el node divergent:** Es compara la paraula actual amb la paraula anterior per trobar el punt en què difereixen.

Paraula Completa		Token 0	Token 1	Token 2
Paraula anterior:	NYAUFO	NY	A	U
Nova paraula:	NYANYO	NY	A	NY
Iguals		✅	✅	❌

    No ens cal revisar altres tokens amb paraules anteriors ja que sabem que cap paraula reusarà sufixos d'una que no sigui l'anterior si la llista de paraules està ordenada.

3. **Minimitzar el sufix:** A partir del punt de divergència, es minimitzen els nodes anteriors per identificar subgràfics equivalents.

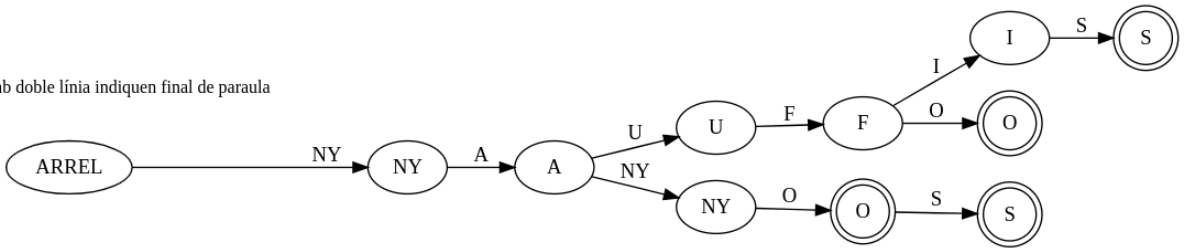
    Si ja existeix un node equivalent, (mateixa estructura i esFinal) es reutilitza. Això evita crear sufixos redundants i garanteix que el DAWG és minimal.

4. **Afegir els nous nodes**: A partir del punt de divergència, es crea un nou camí de nodes, un per cada token restant de la paraula. Aquests nodes s'enllacen amb l'anterior i s'afegeixen al camí actual.

5. **Marcar el node final:** L'últim node afegit es marca com a final. Això indica que el camí des de l'arrel fins aquest node representa una paraula vàlida del diccionari.

DAWG format amb les paraules “NYAUFIS”, “NYAUFO”, “NYANYO” i “NYANYOS”

Nota: els nodes amb doble línia indiquen final de paraula



Cercar paraula al DAWG

Per cercar si un prefix de paraula, o una paraula forma part del DAWG, cal primer tokenitzar el text. Aquest procés té un cost en el pitjor dels casos de **O(w · T)**. On:

- w representa la longitud de la paraula/prefix.
- T representa el nombre de tokens possibles (petit i acotat).

La segona part consisteix en recórrer els nodes del DAWG consultant si cada token de la paraula correspon amb les arestes del dawg. Com el dawg està implementat amb un hashmap, tardem log m temps en obtenir el següent node del dawg.
Aquest procés té un cost de **O(n * log m)** on:

- n representa el nombre de nodes de la paraula.
- m representa el nombre d’arestes que té cada node.

Per tant, el cost total de la cerca és, en el pitjor cas, **O(w * T + n * log m)** on:

- w és la longitud de la paraula o prefix,
- T és el nombre màxim de tokens possibles (petit i acotat),
- n = nombre de tokens obtinguts (n ≤ w),
- m = nombre d’arestes/fills d’un node (m ≤ T).

Ús de memòria i vida útil del DAWG

El DAWG es crea una únicament quan es carrega el diccionari de la partida. Com que té un Cost temporal: O(Σ |w|), lineal en la longitud total de totes les paraules, perquè cada token es visita exactament un cop i cada node es minimitza com a molt un cop.

A la pràctica, el diccionari del llenguatge català (582.020 paraules) és redueix al DAWG i passa a tenir tan sols 39.743 nodes i a ocupar tan sols en memòria uns 4MB. Per altre banda el DAWG tarda en generar-se uns ≈6000ms (depenent de la màquina).

Com que el diccionari (i per tant el DAWG) pot canviar segons la partida, ocupa poc espai a memòria i es genera en poc temps, hem pensat que la millor opció que el DAWG es generi un cop s'inicia la partida i que es mantindrà a memòria fins que: o bé s’acabi la partida, o és tanqui l’aplicació.

Algorisme Jugades Bot

Per a implementar la intel·ligència del jugador automàtic (bot), s'ha dissenyat un sistema que pot adaptar-se a diferents nivells de dificultat. El bot pot escollir la millor jugada possible (nivell 3), una jugada intermedia (nivell 2) o la pitjor jugada entre les vàlides (nivell 1). La selecció es basa en la puntuació obtinguda d'entre totes les jugades vàlides possibles.

Algorisme de Backtracking

Per tal de generar totes les jugades possibles, hem utilitzat un algorisme de **backtracking**. Aquest algorisme consisteix en cercar totes les combinacions possibles que compleixin certes condicions. En el nostre cas, l'objectiu serà cercar totes les **jugades vàlides** que pot fer un jugador sabent l'estat actual del taulell i les fitxes que té a la seva disposició per jugar.

L'algorisme de backtracking recorre el taulell casella per casella, intentant construir totes les paraules (primer horitzontalment i després verticalment).

Per cada posició inicial construeix el prefix existent a partir de les fitxes que ja es troben col·locades en el taulell abans de la posició actual.

A continuació, explora totes les combinacions possibles de fitxes del faristol que poden continuar aquest prefix.

Si la casella està buida, per a cada fitxa disponible al faristol s'hi col·loca una fitxa temporalment. Es construeix un nou prefix a partir de:

- Si la casella està buida amb una fitxa disponible del faristol.
- Si la casella està ocupada amb s'afegeix la fitxa que l'ocupa

Es comprova si aquest nou prefix pot conduir a una paraula vàlida fent ús del DAWG (vegeu secció següent).

Si el prefix és vàlid, es continua recursivament col·locant més fitxes.

Quan s'acaba l'exploració d'aquella branca, es retira la fitxa del faristol i es restaura l'estat anterior per poder provar amb una altra fitxa.

Ús del DAWG per Podar la Recerca

Per evitar generar combinacions que no poden formar cap paraula vàlida, entra en joc el nostre DAWG. Com hem explicat anteriorment aquest conté el conjunt de paraules vàlides del diccionari i ens permet comprovar totes les ramificacions possibles per cada prefix.
Cada vegada que s'afegeix una nova fitxa a una combinació parcial, es comprova si el prefix resultant existeix al DAWG. Si no és així, es **poda** (és a dir, s'abandona aquella possible combinació i es va a provar una altre), estalviant temps i evitant recorreguts inútils. Això millora significativament l'eficiència de l'algorisme.

Validació i Classificació de Jugades

Un cop es genera una combinació que forma una paraula present al DAWG, es valida que compleixi amb les regles del joc Scrabble:

- Estar col·locada al centre si és la primera jugada i estar connectada amb alguna fitxa existent del taulell si ja hi han paraules jugades.
- Formar paraules vàlides amb tots els seus encreuaments tant verticals com horitzontals.
- Si la jugada és vàlida, se'n calcula la puntuació tenint en compte els multiplicadors del taulell i bonificacions com jugar totes les fitxes.

Un cop generades, les jugades vàlides s'agrupen en una llista i s'ordenen seguint aquests criteris, per ordre de prioritat:

1. **Puntuació obtinguda:** Factor més important
2. **Ús de comodins:** en cas d'empat de punts, tenen preferència les jugades que no utilitzen cap comodí.
3. **Longitud de la paraula:** si persisteix l'empat, s'escull la jugada més curta per reduir al màxim l'avantatge del rival, ja que li deixa menys lletres en joc amb les quals combinar lletres.

Finalment segons el nivell de dificultat del bot s'agafa:

- **Nivell 1 (Fàcil)**: S'agafa la última jugada de la llista i per tant la jugada que otorga menys puntuació, amb més comodins i més llarga possible.
- **Nivell 2 (Intermedi)**: S'agafa la jugada del mig de la llista,
- **Nivell 3 (Difícil)**: S'agafa la primera jugada de la llista i per tant la que otorga més punts, amb menys comodins i el més curta possible.

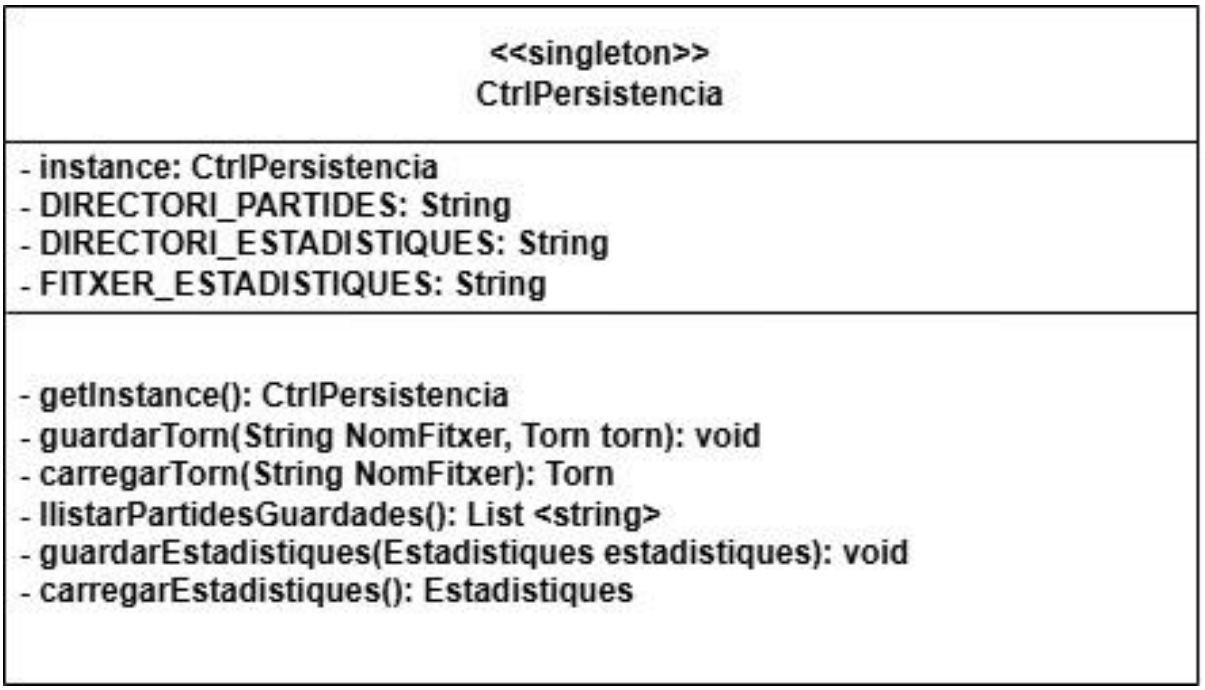
Capa de Persistència

Preàmbul

Aquest apartat descriu el funcionament de la capa de persistència del sistema, centrant-se en la classe CtrlPersistencia, encarregada de gestionar el guardat i la càrrega de les dades rellevants del joc, com ara les partides i les estadístiques globals. La seva arquitectura es fonamenta en el patró Singleton, que garanteix una instància única i un accés centralitzat, afavorint així la consistència i simplicitat del sistema.

A continuació s'explica la justificació de l'ús d'aquest patró de disseny, detallant els avantatges que aporta en el context del nostre projecte. A més, es descriuen les principals operacions que permet aquesta classe, els atributs implicats i com es fa efectiva la persistència mitjançant la serialització d'objectes Java. Finalment, s'explica com s'ha optat per reutilitzar objectes del domini (com Torn i Estadistiques) per simplificar la gestió i garantir una recuperació fidel de l'estat del joc.

Controlador Persistència



Justificació del patró Singleton

- Instància única i control d'accés centralitzat: La classe de persistència actua com a únic punt d'accés a les operacions de guardat i carrega de dades relacionades amb les partides i les estadístiques dels jugadors. No és necessari, ni vam considerar adient, tenir múltiples instàncies d'aquesta classe, ja que podria provocar inconsistencies i duplicació de responsabilitats.

- Absència d'estat intern: Com que el CtrlPersistencia no te atributs que necessitin ser personalitzats per a cada una de les seves instàncies, les seves funcions de exportar i importar son bàsicament operatives. Com que no cal guardar diferents estats de persistència, no hi ha necessitat de tenir múltiples instàncies, de manera que el patró singleton es el més adient.

- Facilitat d'accés: El patró singleton permet que l'accés a la instància de la classe sigui directe i senzill desde qualsevol part del sistema, especialment desde on es crida mes, el CtrlDomini. Això permet tenir un codi net i coherent sense la necessitat de passar objectes de persitència com paràmetres.

Descripció del controlador de la capa de persistència

Nom	CtrlPersistencia
Descripció	Aquesta classe es la responsable de la gestió de persistència de dades del projecte. Encapsula les operacions per guardar i carregar tant l'estat de la partida (Torn) com de les estadìstiques globals del joc (Estadistiques). Les dades es guarden en fitxers binaris utilitzant la serialització de Java, la qual cosa garanteix que el estat de l'objecte persisteix de forma segura.
Atributs	**-instance (CtrlPersistencia):** Instància única de la classe per implementar el patró singleton **-DIRECTORI_PARTIDES (String):** Ruta relativa on es guarden les partides **-DIRECTORI_ESTADISTIQUES (String):** Ruta relativa on es guarden les estadìstiques **-FITXER_ESTADISTIQUES (String):** Fitxer concret on es guarda l'objecte de estadìstiques
Mètodes	**- getInstance():** Retorna la instància única de ctrlPersistencia **- guardarTorn (String nomFitxer, Torn torn):** Guarda una instància de Torn com un fitxer binari Excepcions: ExcepcioLectura ExcepcioEscriptura

	- carregarTorn (String nomFitxer): TORN Recupera una instància de Torn guardada en un fitxer binari Excepcions: ExcepcioLectura ExcepcioEscriptura **- llistarPartidesGuardades(): LIST<STRING>** Retorna una llista amb els noms de les partides guardades **- guardarEstadistiques (Estadistiques estadistiques):** Guarda les estadistiques globals del usuari Excepcions: ExcepcioLectura ExcepcioEscriptura **- carregarEstadistiques(): ESTADISTIQUES** Carrega les estadistiques guardades o crea una nova instància buida si no existeix cap fitxer. Excepcions: ExcepcioLectura ExcepcioEscriptura

Funcionament de la serialització

El nostres sistema de persistència es basa en fitxers binaris mitjançant la serialització d'objectes en java. Per això, les classes Torn (i subclasses que conté, com per exemple Taulell, Faristol...) i Estadistiques implementen la interfície Serializable, la qual permet convertir una instància d'objecte en una sequencia de bytes per guardar-la i, posteriorment, recuperarla exactament amb el mateix estat intern.

No s'ha creat una classe específica que representi el que es guarda perquè es reutilitzen directament objectes del model de domini (Torn i Estadistiques). Aquest enfocament simplifica la persistencia, ja que els objectes que encapsulen tot el estat del joc i les mètriques es poden escriure i llegir directament, sempre que mantinguin compatibilitat de serializtacio.

Respecte a les altres opcions que vam considerar aquest enfocament te diversos avantatges:

- Evita la necessitat de crear estructures duplicades per a la persitència
- Permet guardar tot el estat amb una sola operació
- Millora el rendiment al no haver de parsejar ni transformar el contingut a formats com el JSON o el CSV
- El manteniment és senzill, només cal assegurar-se que les classes implicades siguin Serializable i compatibles entre versions