

---

# COFFEE CLICKER

---

Projecte final semestre 2



26/MAIG/2024

ID GRUP: COFFEE CLICKER 3

Marina Miranda Riaza – marina.miranda

Pol Monné Parera – pol.monne

Ricard Morales Prat – Ricard.morales

Daniel Soler Fontanet – daniel.soler

David Susagna Holgado – david.susagna

Disseny i Programació Orientat a Objectes– Daniel Gil

## INDEX

1. Resum del projecte.....	3
2. Explicació del disseny implementat (GUI) .....	4
2.2 Estructura principal.....	4
2.2 Home View.....	6
2.3 Vista registre usuari .....	8
2.4 Vista iniciar sessió .....	9
2.5 Vista iniciar partida .....	11
2.6 Vista Continuar partida .....	12
2.7 Vista gestió de compte.....	13
2.8 Vista veure estadístiques .....	15
2.9 Vista partida .....	18
2.10 Vista Recuperar contrasenya (optatiu) .....	20
3. Diagrama de classes .....	22
3.1 Arquitectura MVC .....	22
3.2 Bloc presentation .....	23
3.3 Bloc business.....	28
3.4 Bloc persistence .....	29
3.5 Diagrama final .....	32
4. Metodologia de desenvolupament .....	33
4.1 Primera fase .....	33
4.2 Segona fase .....	34
4.3 Tercera fase.....	35
4.3 Eines desenvolupament.....	35

4.4. Disseny de la base de dades.....	36
5. Dedicació .....	39
6. Conclusions.....	41
7. Bibliografia.....	42
8. Annexos.....	43

## 1. RESUM DEL PROJECTE

El projecte es centra en la realització del disseny i la programació d'un joc de generació de recursos. En aquesta pràctica en concret, es tracta d'un generador de cafès, "CoffeeClicker". El joc es desenvolupa en una única vista on es troben els generadors de recursos, les millores que se'ls hi poden aplicar a aquests, juntament amb una taula que mostra les estadístiques principals de la partida, permetent així que l'usuari tingui un major control del joc.

A més, l'usuari comptarà amb una botiga de millores que li oferirà la possibilitat d'escol·lir una estratègia per obtenir el major nombre de recursos en el menor temps possible. Aquestes millores estan dissenyades per optimitzar l'experiència de joc i proporcionar una varietat d'opcions estratègiques.

El disseny del joc es caracteritza per una temàtica desenfadada, amb diferents tipus de cafès i personatges que accompanyen l'usuari durant la partida. Aquests personatges es poden trobar en els diferents missatges que apareixen al llarg del joc, aportant una dimensió lúdica i entretinguda a l'experiència. Això ho veiem per exemple quan hi ha algun error a l'hora d'inserir dades, en aquest cas doncs, salta una pantalla pop up amb un dels personatges i on s'indica l'error a l'usuari.

Pel que fa a la dinàmica del joc, els usuaris tenen l'opció de registrar-se o iniciar sessió si ja s'han registrat prèviament. Un cop dins del joc amb el compte corresponent, poden continuar una partida ja començada o iniciar-ne una de nova. Aquesta funcionalitat assegura que els jugadors puguin mantenir el progrés de les seves partides i seguir desenvolupant les seves estratègies per assolir els objectius del joc.

Per finalitzar, cal destacar que totes les dades de les partides de cada jugador s'emmagatzemen en una base de dades. D'aquesta forma, el jugador podrà tenir un seguiment de la seva evolució en les partides, i poder continuar jugant aquella partida que havia deixat oberta.

## 2. EXPLICACIÓ DEL DISSENY IMPLEMENTAT (GUI)

El disseny implementat en el projecte es distingeix per tenir dos tipus de vistes diferents. La primera està relacionada amb les accions prèvies a la partida, com ara la funcionalitat de registrar-se, la gestió de l'usuari i la gestió de les partides. En aquestes vistes es pot observar que totes segueixen la mateixa estructura: el logotip del joc, juntament amb el títol de la vista, i sota aquests elements es presenten les opcions corresponents a cada vista.

D'altra banda, la vista de la partida presenta un disseny diferent, orientat específicament a la jugabilitat. Aquest disseny està enfocat a proporcionar una experiència immersiva i interactiva durant la partida.

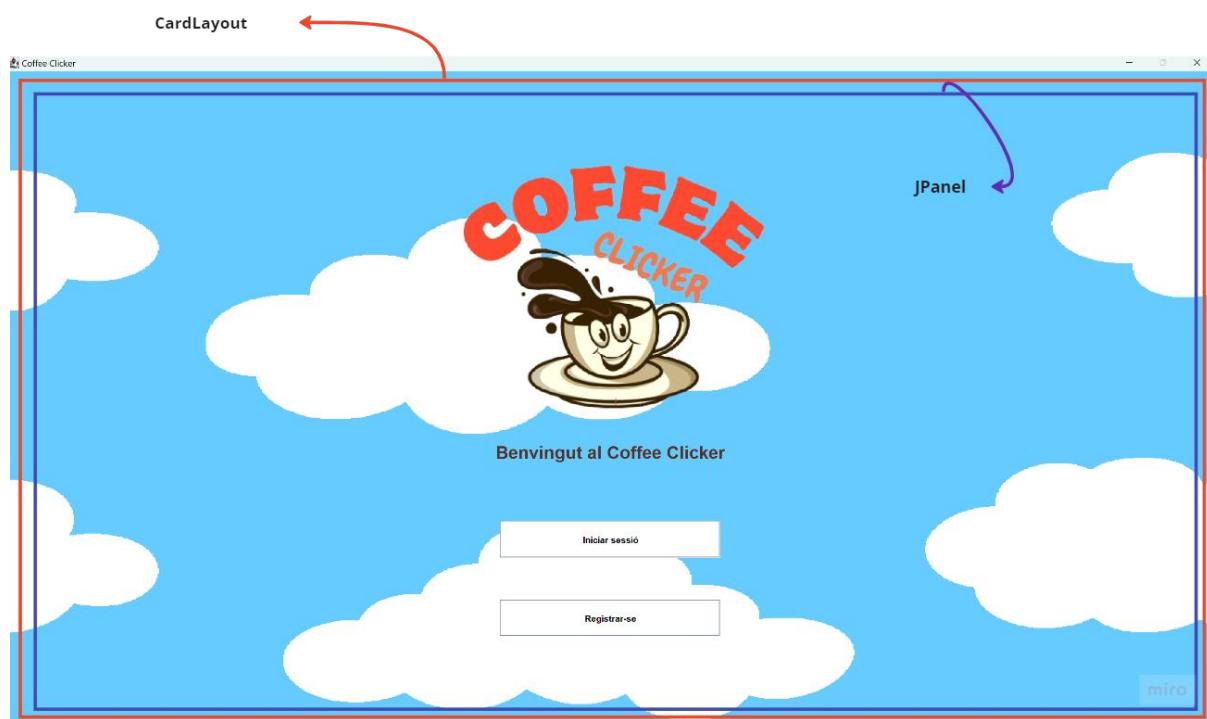
És important destacar que totes les vistes es descriuen com a panells, que s'organitzen en un *Card Layout*. Mitjançant la lògica del programa, es mostra un panell o un altre segons la situació. Aquesta estructura permet una navegació fluida i intuïtiva entre les diferents fases del joc, assegurant que l'usuari tingui una experiència coherent i ben organitzada.

### 2.2 ESTRUCTURA PRINCIPAL

Amb la intenció de fer el programa senzill d'utilitzar per part de l'usuari i assegurar que no es modifiqui independentment de l'ordinador en el qual s'executi, s'ha optat per crear una finestra amb la mida de la resolució de la pantalla, la qual s'obre a pantalla completa, tal com es veu en molts jocs d'ordinador.

Aquesta estructura inicial de finestra ens permet crear un panell principal que ocupa tota la pantalla, on es mostraran les diferents vistes del joc. Per aquest motiu, cada vista només retorna un panell amb tota la informació que es vol mostrar. Aquesta estructura millora la fluïdesa en els temps de càrrega entre vistes.

Per últim, cal destacar que durant tota l'execució del joc, s'està reproduint una música de fons que proporciona ambientació i immersió en la temàtica de la partida. Aquesta música està lligada amb la finestra de manera que, independentment del canvi de vista, la música segueix el seu ritme sense interrupcions. També cal destacar que en el cas del gif de fons que es veu durant l'execució, aquest manté una continuïtat amb els canvis de vista, ja que es troba darrere dels panells que canvien, sent independent de la seva posició en la interfície de l'usuari durant l'ús del programa, GIF que només es veurà present a les vistes prèvies al joc. Això permet una experiència més fluida i immersiva per a l'usuari.



Tal com es pot veure en la captura, l'estructura principal destaca per ser un CardLayout, el qual es troba en la classe MainFrame.java. Dins d'aquest, es carreguen totes les vistes utilitzant un JPanel. Aquesta configuració permet una navegació fluida i organitzada entre les diferents vistes del programa, facilitant la gestió i el canvi de panells segons les accions de l'usuari.

## 2.2 HOME VIEW

La vista inicial és la primera que es mostra en executar el programa. En aquesta vista, es presenta el logotip del joc, el qual es troba dins d'un panell amb una imatge incorporada. Aquesta estructura permet una adaptació més precisa amb la resta de components de la vista.

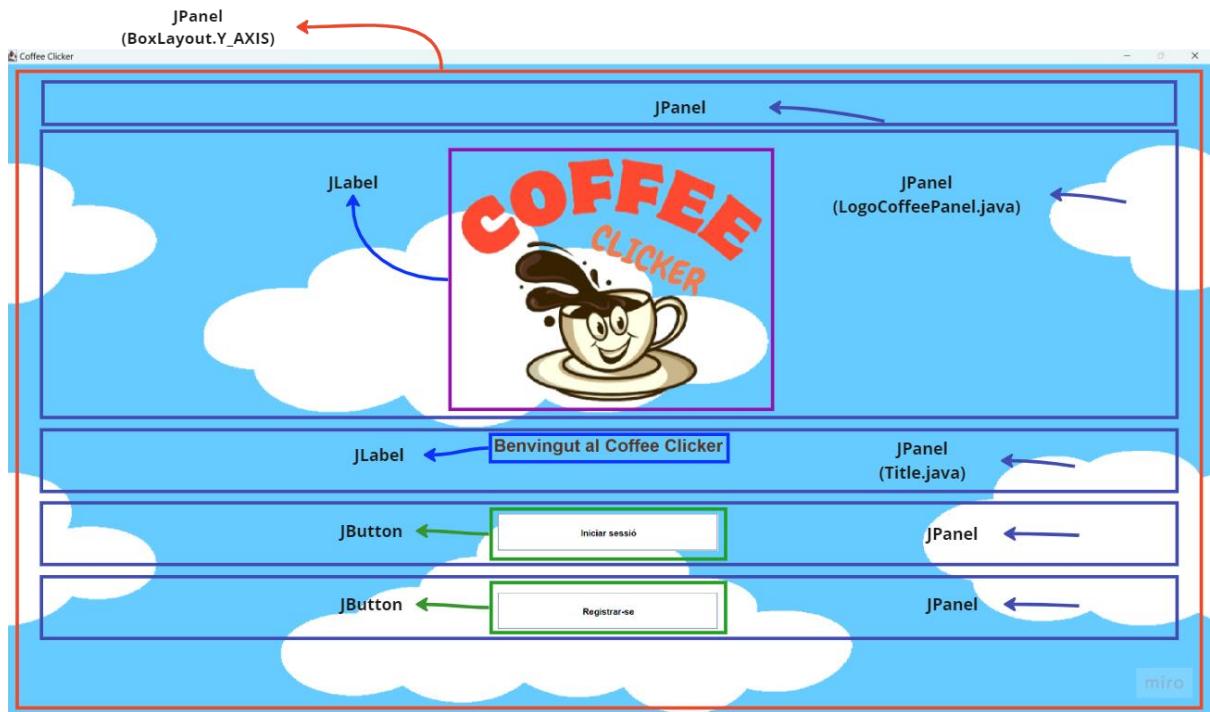
Donat que aquest panell es mostra en diferents vistes del joc, es va optar per crear una classe auxiliar encarregada de retornar el panell amb totes les seves característiques. Aquesta abstracció del codi permet una major reutilització, fent el disseny més eficient i mantenible. La classe auxiliar simplifica la gestió del logotip a través de les diferents vistes, assegurant que es mantingui una coherència visual i funcional al llarg de tota l'experiència de l'usuari.

El segon element que trobem en aquesta vista és el títol. Aquest apartat també es repeteix nombroses vegades al llarg de diferents vistes. Per aquest motiu, s'ha creat una classe que permet la seva reutilització en cadascuna d'elles, reduint així el volum de codi.

Aquesta classe proporciona una manera estandardizada i eficient de mostrar el títol en les diverses vistes, assegurant que es mantingui una aparença i funcionalitat coherents. Això no només millora la llegibilitat del codi, sinó que també facilita el manteniment i l'extensió futura del projecte. Mitjançant aquesta abstracció, s'optimitza el procés de desenvolupament i es garanteix una experiència d'usuari uniforme i professional.

Per últim, es pot observar la part més personalizable en cadascun dels panells. Per aquest motiu, tot i que es deixi com un panell que es pot editar en cada una de les vistes, en aquest cas ens trobem amb les dues opcions inicials: iniciar sessió i registrar-se.

Cada una d'aquestes opcions s'executa mitjançant un botó, el qual canvia la vista del panell principal en cada vista. Això significa que, en prémer un dels botons, el panell principal es modificarà per mostrar la vista corresponent, ja sigui la d'inici de sessió o la de registre. Aquesta estructura permet una interacció fluida i intuitiva per part de l'usuari, facilitant-ne la navegació pel joc.

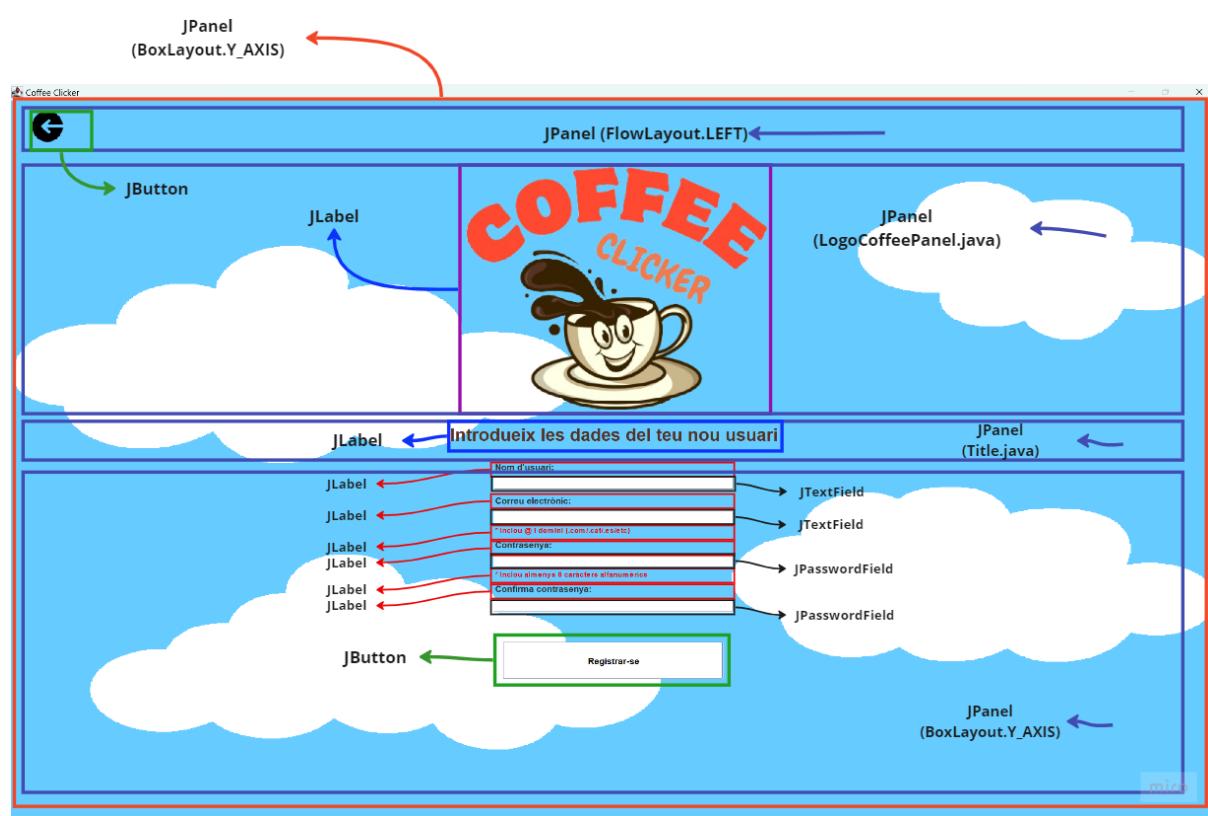


En aquesta vista es pot veure com la vista es troba dins d'un panell ordenat de forma vertical, on s'introdueixen els subpanells. El primer panell és un espai reservat per a la barra superior de les altres vistes que venen a continuació. El segon panell que s'afegeix a la vista és el que prové de la classe on es genera el logotip. El tercer panell representa el títol, el qual es retorna de la classe Title.java. El quart i el cinquè són panells on es troben els botons corresponents a la vista.

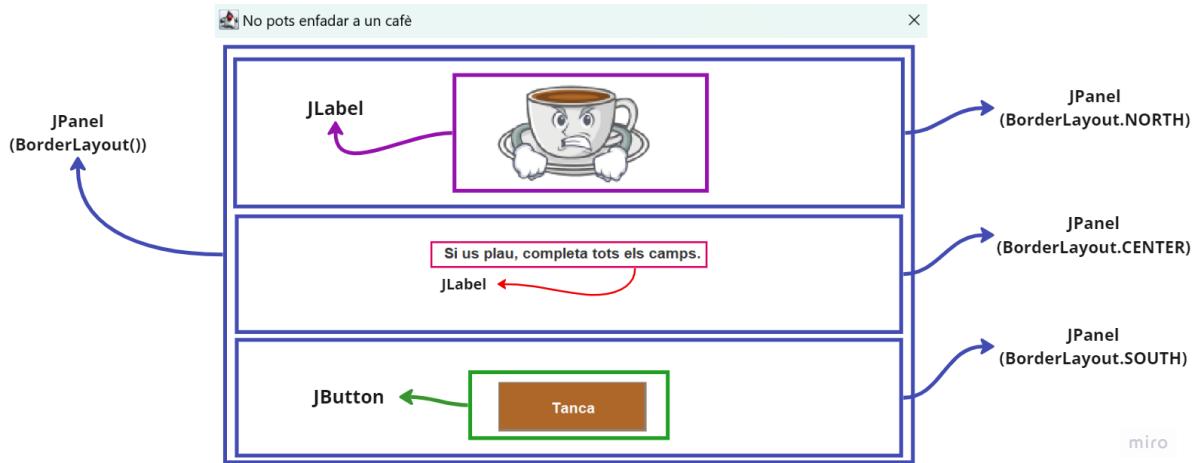
## 2.3 VISTA REGISTRE USUARI

Com s'ha comentat en l'apartat anterior, aquesta és una de les vistes prèvies a la partida que segueix l'estruccura inicial, on es troben el logotip, el títol i la part fonamental de la vista.

En aquesta vista, es poden observar una sèrie de camps editables on l'usuari pot introduir la informació pertinent, així com un botó per executar la comanda de registre. Un cop es prem aquest botó, s'executa la lògica del programa. En el cas que tots els camps estiguin complets de manera correcta, la vista canvia, permetent a l'usuari iniciar la partida i accedir a altres opcions.



En cas que l'usuari cometi algun error o el programa trobi alguna inconsistència, apareixerà un *pop-up*. Aquest *pop-up* desapareixerà de manera automàtica després d'uns segons, o l'usuari podrà tancar-lo manualment per corregir els errors en la introducció de les dades. Aquesta funcionalitat millora la usabilitat del programa, ja que proporciona feedback immediat a l'usuari sobre l'estat de les seves accions i els possibles errors que pugui haver comès.

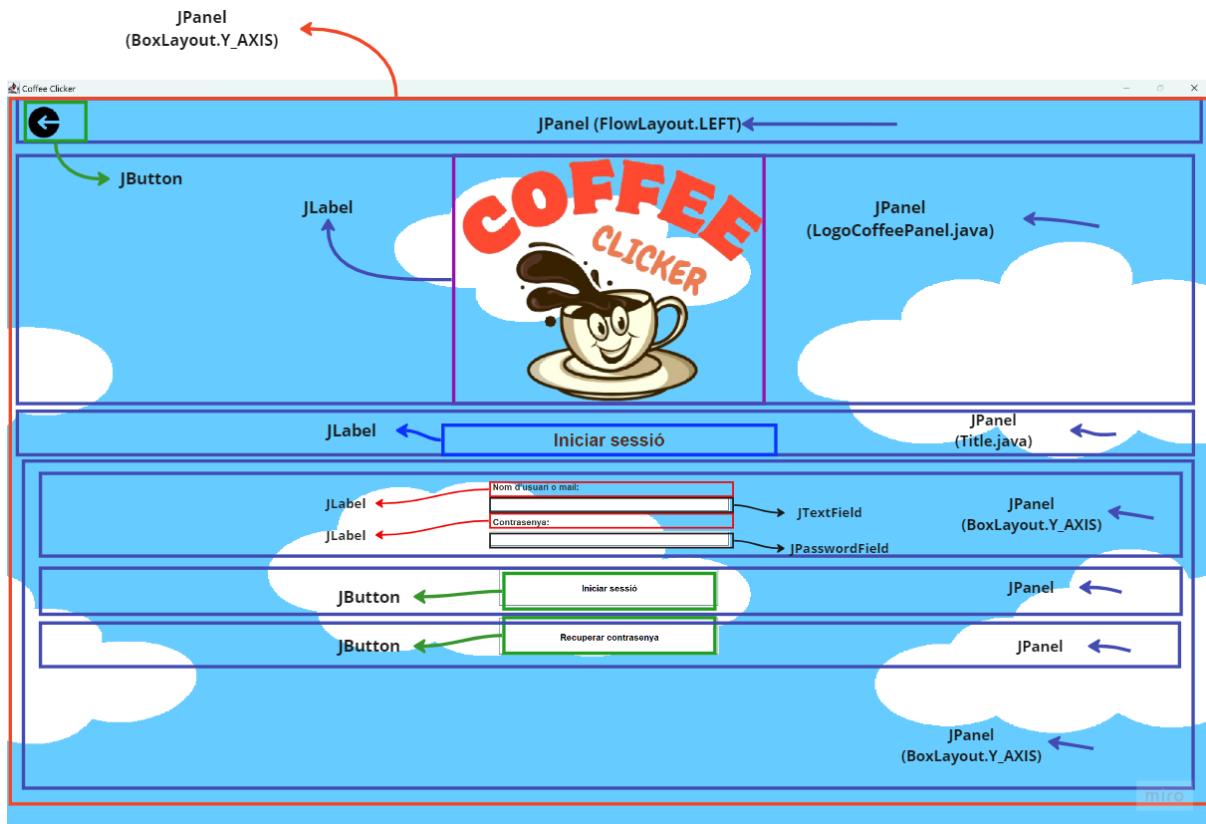


En aquesta vista, destaca per ser la primera en la qual es mostra la barra superior. En futures vistes, aquesta barra inclourà el botó de tornar enrere i l'opció d'accedir al panell de l'usuari, el qual permetrà iniciar sessió o eliminar el compte. En el cas de la vista actual, només es mostra el botó de tornar enrere.

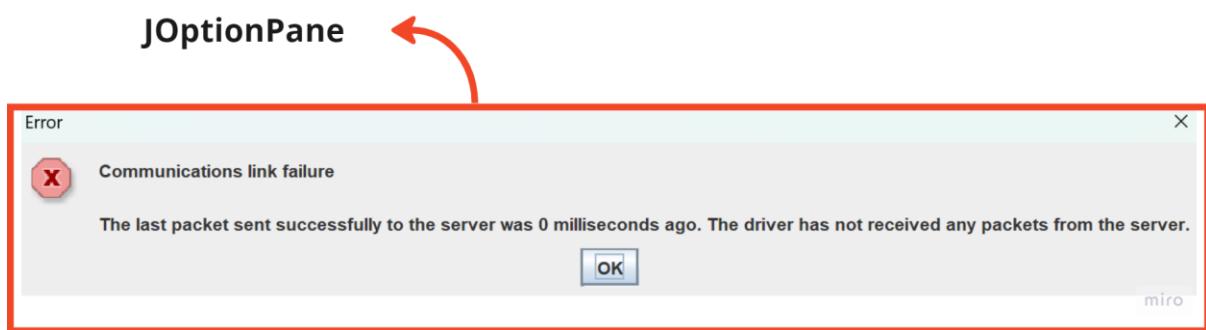
Aquest botó de tornar enrere està programat de tal manera que, en el moment que es canvia de vista, aquesta vista d'origen s'emmagatzema a l'etiqueta en la memòria RAM. D'aquesta manera, el botó de tornar enrere crida a un canvi de vista per a la vista d'origen, de manera que el botó sempre redirigeix a la vista des d'on es va arribar, independentment del camí que es va seguir per arribar a la vista actual.

## 2.4 VISTA INICIAR SESSIÓ

La vista per entrar a la partida manté el format de les dues vistes anteriors, és a dir, el format de les vistes prepartida, amb el logotip i el títol. En aquest cas, la informació del panell es modifica amb dos camps que l'usuari ha de completar: un per introduir el nom d'usuari o el correu electrònic i un altre per introduir la contrasenya. Aquest darrer camp es destaca per no mostrar el que s'està escrivint, sinó que només mostra una sèrie de punts per cada caràcter introduït, cosa que permet una introducció de la contrasenya més segura.



En cas que l'usuari cometí algun error, es mostrerà el mateix *pop-up* d'error amb el missatge corresponent. No obstant això, si es produeix algun problema en l'execució, ja sigui per una crida a la base de dades o una excepció del programa, es mostrerà un nou *pop-up* amb la informació de l'excepció.



D'altra banda, a l'últim apartat de la vista es pot veure el botó per recuperar la contrasenya. Aquest botó permet canviar de vista perquè l'usuari pugui recuperar la contrasenya en cas que no la recordi.

Quan es prem aquest botó, es redirigeix a l'usuari a una nova vista específica per a la recuperació de la contrasenya. Aquesta vista sol·licitarà informació addicional per verificar la identitat de l'usuari, com ara el correu electrònic associat al compte. Aquest procés assegura

que l'usuari pugui restablir la seva contrasenya de manera segura i continuar accedint al joc sense problemes.

## 2.5 VISTA INICIAR PARTIDA

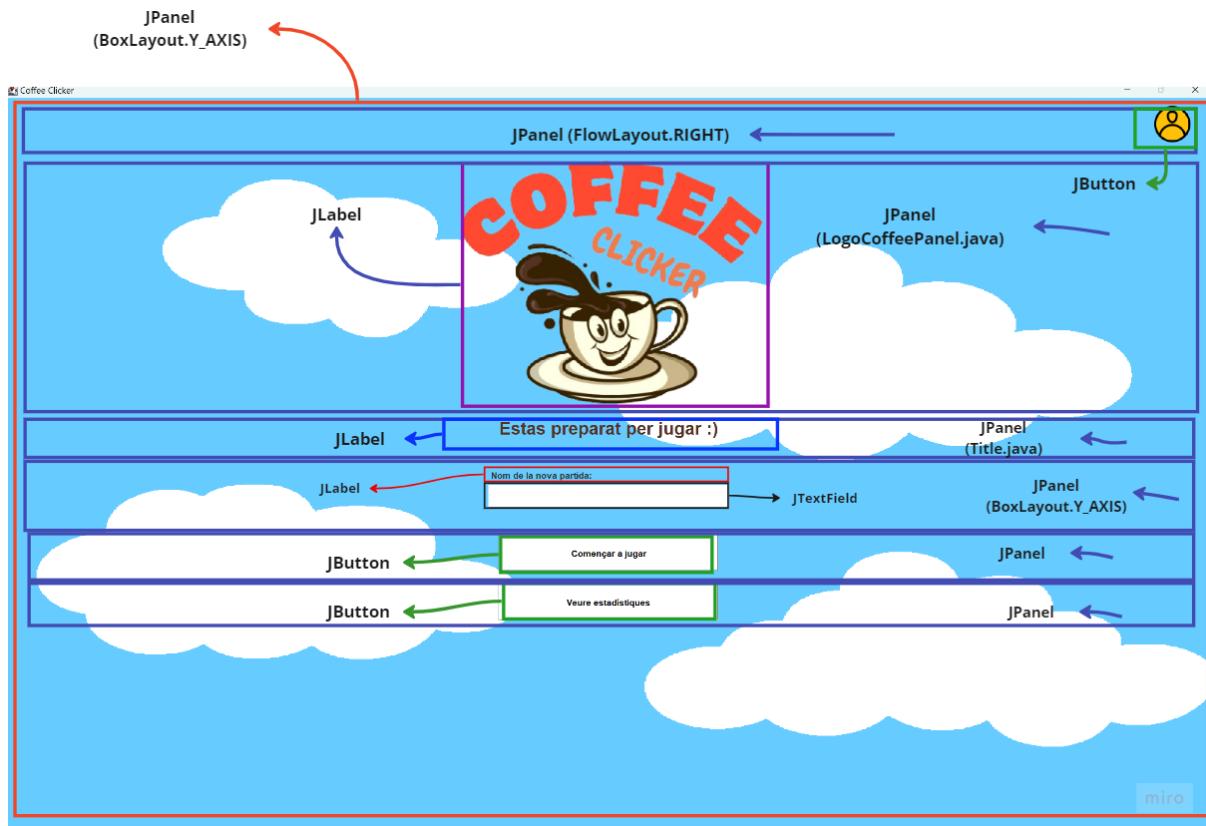
Un cop l'usuari es troba dins del seu compte, té dues opcions: iniciar una partida nova o continuar una partida ja començada. En aquest apartat s'explica la primera opció. Cal destacar que la lògica del joc i l'enunciat defineixen que un usuari no pot tenir més d'una partida iniciada al mateix temps, de manera que aquesta vista només es mostra en cas que l'usuari no tingui cap partida oberta.

Pel que fa a la vista, cal destacar que manté la mateixa estructura que les vistes anteriors, modificant només la part personalizable corresponent. En aquesta vista es troba un camp per omplir amb la informació per part de l'usuari amb el nom de la partida. Aquest nom ha de ser únic per a totes les partides guardades a la base de dades, independentment de l'usuari. Per aquest motiu, si el nom de la partida no és vàlid, el programa mostrarà un missatge d'error mitjançant el *pop-up* d'error mencionat en apartats anteriors.

D'altra banda, la vista incorpora dos botons més: un de confirmació per a la creació del joc i inici de la partida, i un segon botó relacionat amb les estadístiques de les partides finalitzades. Cada un d'aquests botons canvia el panell per mostrar la vista corresponent a cada funció.

El botó de confirmació permet a l'usuari crear una nova partida amb el nom proporcionat i iniciar-la immediatament, sempre que el nom sigui vàlid i únic. Aquesta acció assegura que l'usuari pugui començar a jugar de seguida, mantenint la fluïdesa del procés.

El segon botó porta l'usuari a una vista on es poden consultar les estadístiques de les partides finalitzades. Aquesta vista proporciona una visió detallada del rendiment de l'usuari en les seves partides anteriors, oferint dades valioses per a l'anàlisi i millora de les seves estratègies de joc.

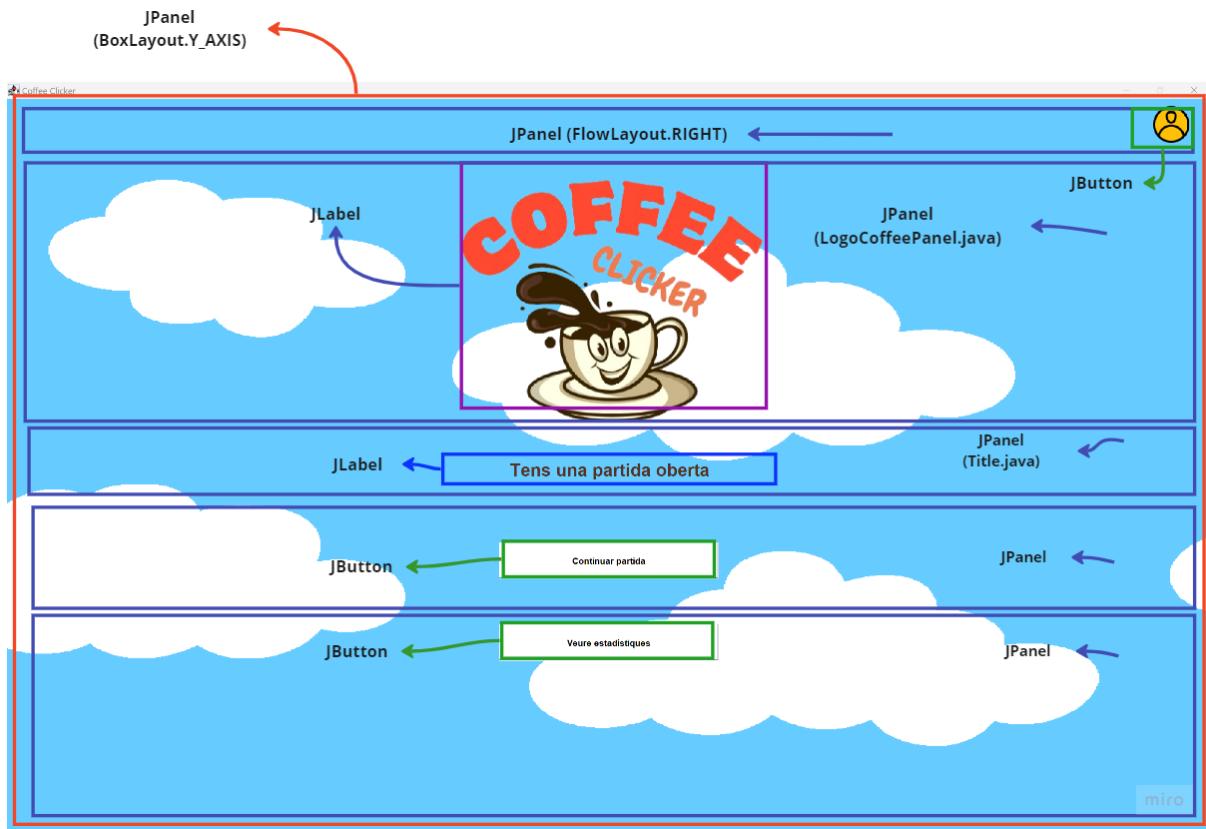


## 2.6 VISTA CONTINUAR PARTIDA

Com en la vista anterior, aquesta vista només és accessible un cop l'usuari ha iniciat sessió. No obstant això, aquesta vista està dedicada als usuaris que ja tenen una partida iniciada. Per tant, només es poden observar dos botons.

El botó de continuar partida, aquest permet a l'usuari passar directament a la vista del joc, on podrà reprendre la partida que havia deixat en pausa. Aquesta funcionalitat assegura que l'usuari pugui continuar el seu joc sense interrupcions.

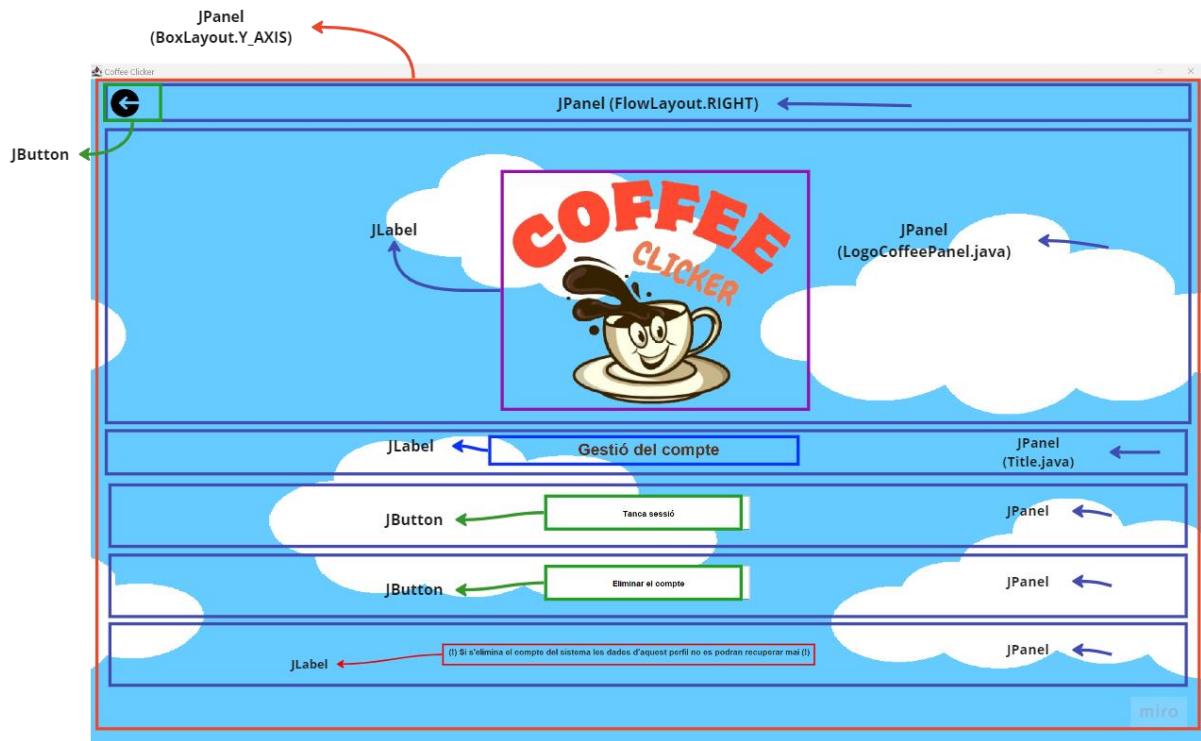
I el botó per veure les estadístiques: que permet canviar a la vista on es poden consultar les estadístiques de les partides finalitzades. Aquesta vista proporciona a l'usuari informació detallada sobre el seu rendiment en partides anteriors, permetent-li analitzar i millorar les seves estratègies de joc.



## 2.7 VISTA GESTIÓ DE COMPTE

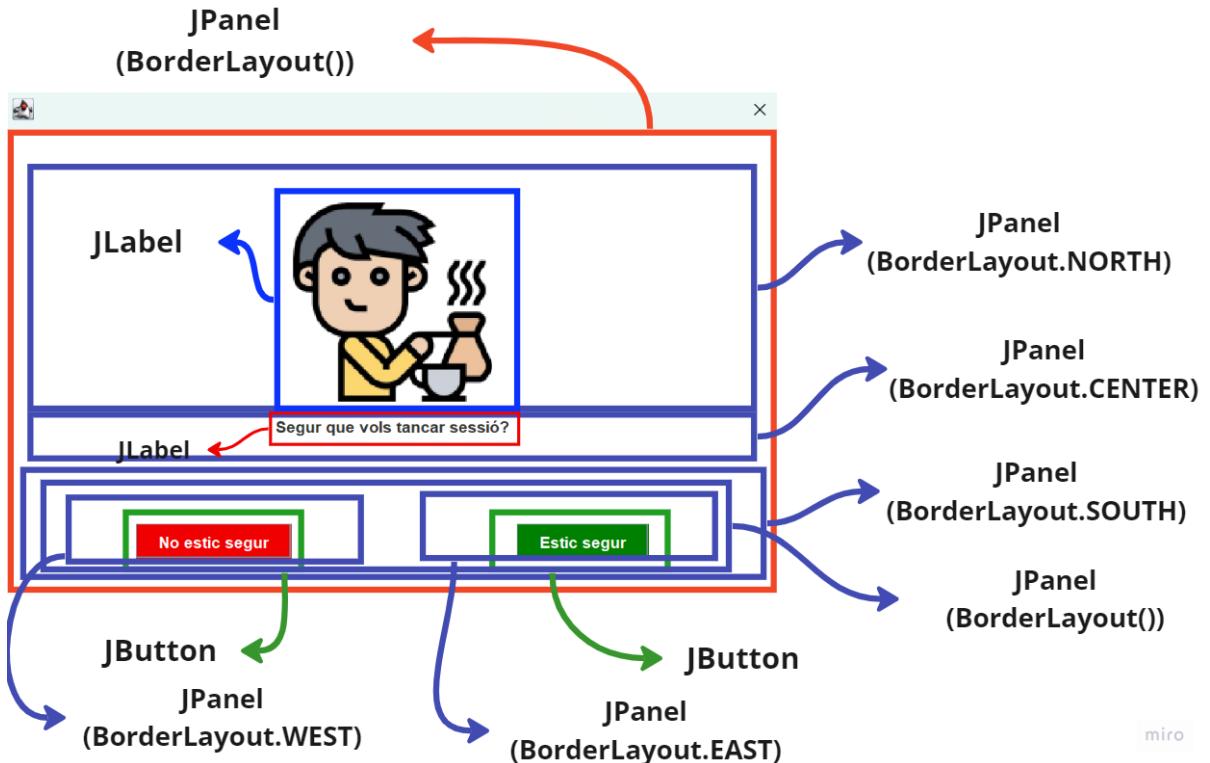
La vista per gestionar el compte és l'última que es troba amb el format previ a la vista de la partida. La intencionalitat d'aquesta vista es centra en la gestió del compte de l'usuari, on es poden trobar les funcionalitats relacionades amb l'eliminació del compte, com ara el tancament de sessió temporal. Per accedir a aquesta vista, només es pot fer mitjançant el botó de la barra superior disponible des de la resta de vistes quan l'usuari té la sessió iniciada.

En aquesta vista, l'usuari pot realitzar diverses accions relacionades amb la gestió del seu compte, com ara canviar la contrasenya, actualitzar la informació personal o eliminar el compte permanentment. Aquesta funcionalitat proporciona a l'usuari el control total sobre el seu compte i la seva privacitat.



En el cas del botó en què es pot tancar la sessió, el programa elimina de la memòria RAM el nom d'usuari que està iniciada la sessió. No obstant això, per garantir la funcionalitat i prevenir possibles errors per part de l'usuari, aquest botó inicia un *pop-up* on es mostra un missatge de confirmació per tancar la sessió o no.

Pel que fa al segon botó, està relacionat amb l'eliminació del compte. Aquesta opció elimina tot el rastre de l'usuari dins el programa i la base de dades, incloent-hi les partides i el propi usuari, deixant lliure aquest nom d'usuari per a futurs nous usuaris. Igual que amb el botó anterior, es lanza un *pop-up* amb el missatge de confirmació i una advertència a l'usuari sobre les conseqüències d'eliminar el compte.

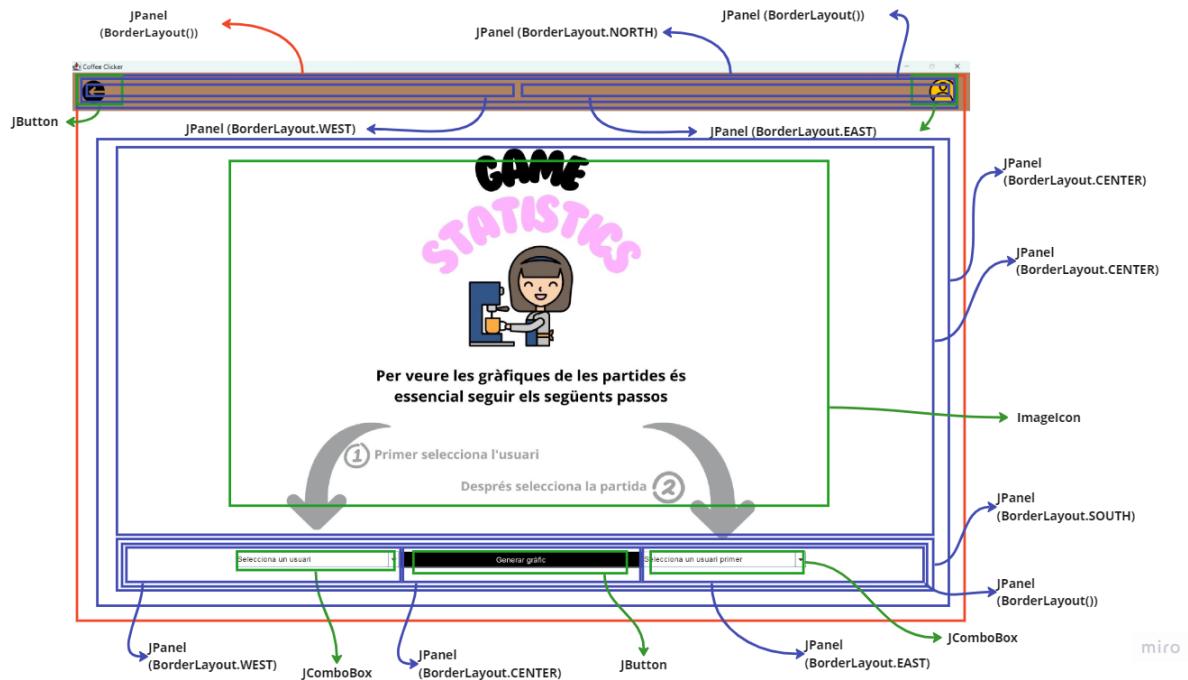


## 2.8 VISTA VEURE ESTADÍSTIQUES

Aquest panell té un format diferent dels panells exposats fins al moment. En aquest cas, es manté la barra superior amb el botó de retrocedir i per accedir al menú de l'usuari.

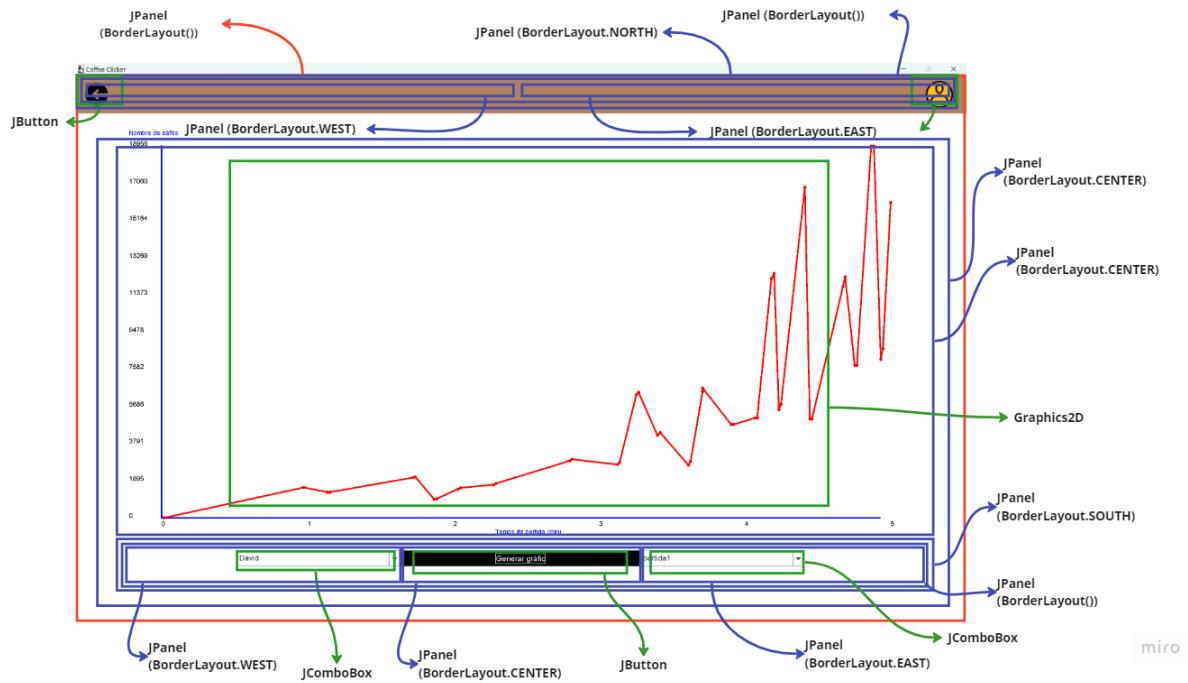
La vista està composta per una imatge inicial que ocupa la major part de la pantalla. En aquesta imatge es mostren les instruccions per a l'usuari per poder veure les estadístiques.

A la part inferior de la pantalla es pot veure un menú format per dos menús desplegables i un botó central. En el primer menú desplegable es mostra la llista dels usuaris registrats al programa. El segon menú desplegable es troba buit en un moment inicial, fins que se selecciona un usuari. En aquest moment es mostren les partides finalitzades relacionades amb el jugador seleccionat.

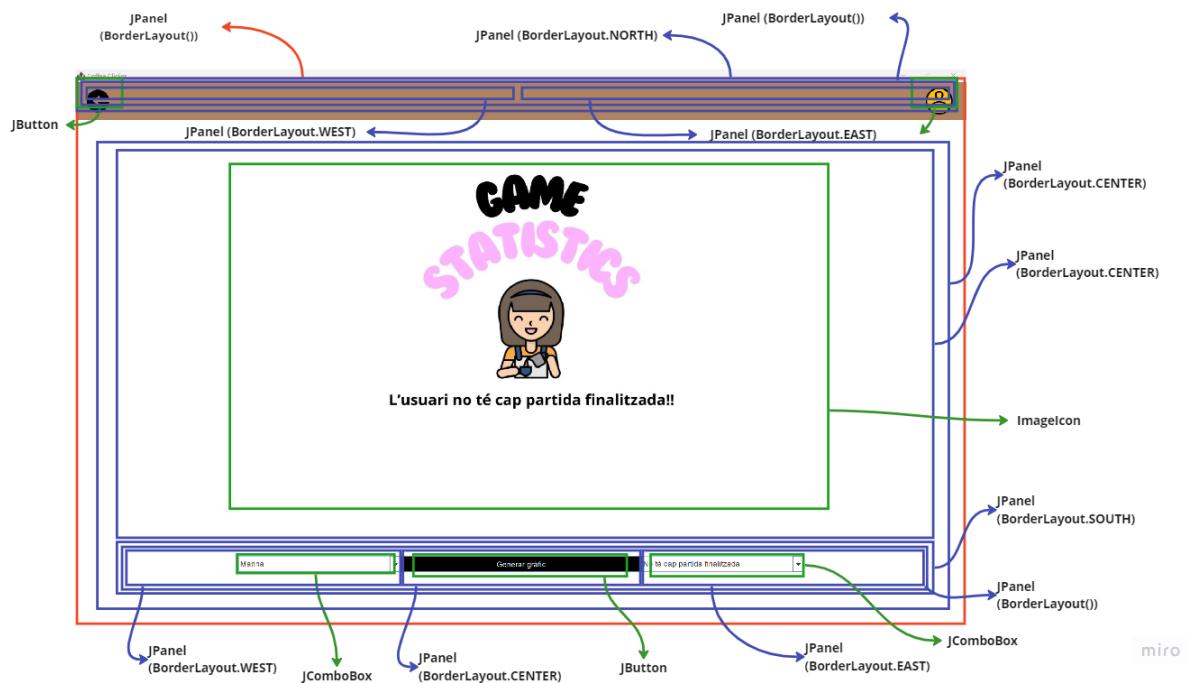


Un cop seleccionats els dos components per part de l'usuari, es pot prémer el botó per generar el gràfic de les estadístiques. Amb la intenció d'evitar errors de selecció, en el cas que es seleccioni alguna configuració incorrecta o no es seleccioni cap partida, el programa mostrarà un *pop-up* d'error indicant a l'usuari quin és l'error.

En el cas que l'usuari seleccioni de manera correcta la informació dels menús desplegables, la imatge superior on es mostren les instruccions es modificarà per mostrar el gràfic amb les estadístiques. Aquest gràfic és adaptatiu i es modifica tant en nombre de recursos com amb la dimensió de l'eix temporal, adaptant-se a la informació que es vol mostrar.



Finalment, destaca que en el cas que l'usuari no tingui cap partida finalitzada, en el menú desplegable on se seleccionen les partides es mostrarà un missatge indicant que no hi ha partides finalitzades. D'altra banda, la imatge on es mostren les instruccions es modificarà per mostrar un missatge que indiqui que és necessari que l'usuari seleccionat finalitzi alguna partida per poder mostrar les estadístiques.



## 2.9 VISTA PARTIDA

La vista de la partida és l'última que trobem en tota l'execució del programa, i és totalment diferent de totes les vistes explicades anteriorment.

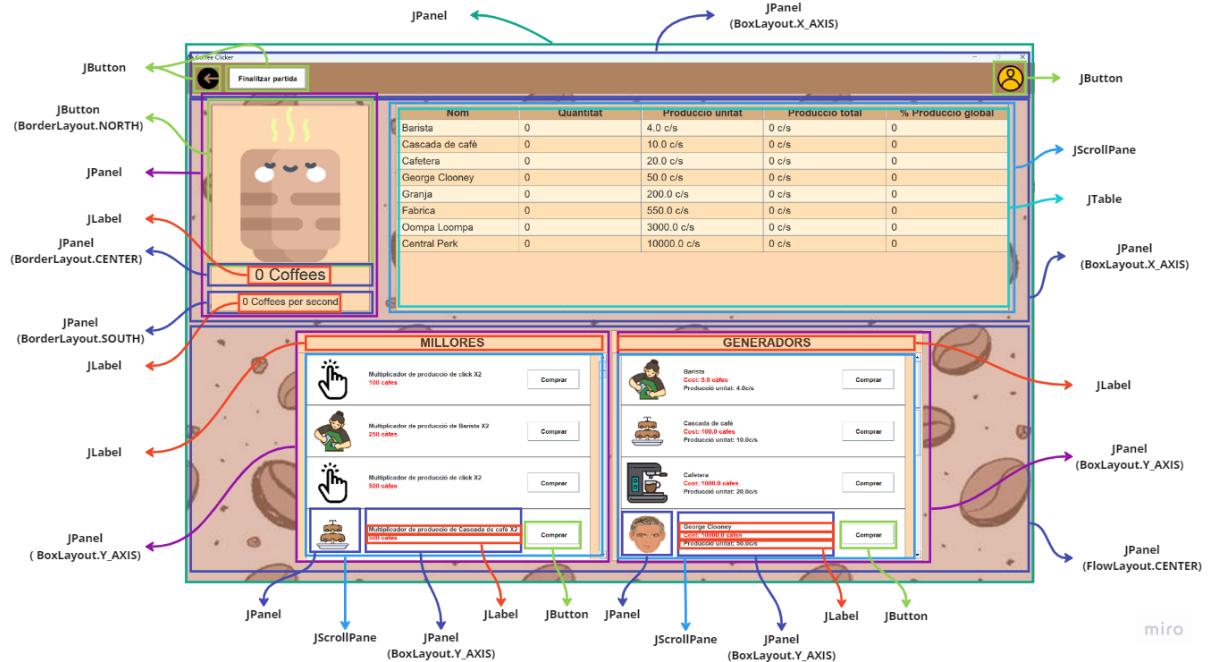
La part superior de la partida destaca per tenir la barra superior, igual que les anteriors. No obstant això, en aquest cas, incorpora el botó amb el qual es pot finalitzar la partida. Aquesta opció modifica la base de dades per marcar la partida com a finalitzada, permetent així iniciar una nova partida i que es puguin veure les estadístiques corresponents a l'apartat de mostra de les estadístiques.

Aquest botó de finalitzar la partida és clau per al funcionament del joc, ja que permet als jugadors tancar la partida quan ho desitgin i registrar-ne els resultats per a futures referències i anàlisis. A més, la inclusió d'aquest botó a la barra superior proporciona una navegació coherent i intuïtiva per als jugadors, degut a que es troba en un lloc familiar i accessible.

El panell central de la vista està format per tres parts. En primer lloc, trobem el propi generador de cafès, que es pot accionar mitjançant el clic del ratolí. Aquest és un botó amb una imatge que acumula recursos manualment. També es mostra la quantitat total de recursos acumulats, juntament amb el total de cafès per segon que es generen per mitjà dels generadors automàtics que es troben a la tenda.

A continuació, es mostra la taula dels generadors, on es pot veure de manera clara i resumida quins són els generadors actius amb tota la informació relacionada amb ells. Aquesta taula permet millorar la gestió dels recursos i modificar l'estrategia que vol seguir l'usuari.

Finalment, l'última part que trobem en aquesta vista és la tenda de millores i de generadors. Totes dues tenen el mateix format, que consisteix en una vista amb desplaçament vertical on es troben tots els articles disponibles. En el cas de les millores, aquests s'esborren de la vista cada cop que se'n compra una, ja que no es poden tornar a comprar. Pel que fa als generadors, no desapareixen, però sí que es modifiquen els valors i els costos que tenen cada un d'ells en funció del nivell que es compra.



En aquest punt, el programa pot tenir un problema de rigorositat en la presentació de les dades, ja que ens trobem amb dos generadors de recursos: el primer depèn de l'usuari i el segon depèn de la generació automàtica per part dels generadors.

Per solucionar aquest problema, s'ha optat per generar un rellotge "thread" que unifica tota la informació cada 250 ms. Aquest rellotge recull els recursos generats per l'usuari, així com els recursos generats pels generadors automàtics. Aquesta informació s'unifica i s'actualitza la vista de la taula de la vista, així com el total de recursos.

Durant el transcurs de la pràctica, es va observar un problema de latència en la base de dades a l'hora de generar les consultes SQL. Aquest problema era causat pel servidor en línia on està guardada la base de dades. Aquesta latència en les consultes impedia que el programa pogués jugar en temps real amb la informació de la base de dades. Per solucionar aquest problema, es va decidir generar un "thread" que guarda la partida a la base de dades cada 60 segons, mentre que tota la informació de la partida es gestiona en RAM. A més, cada vegada que se surt de la partida, es guarda l'estat actual a la base de dades.

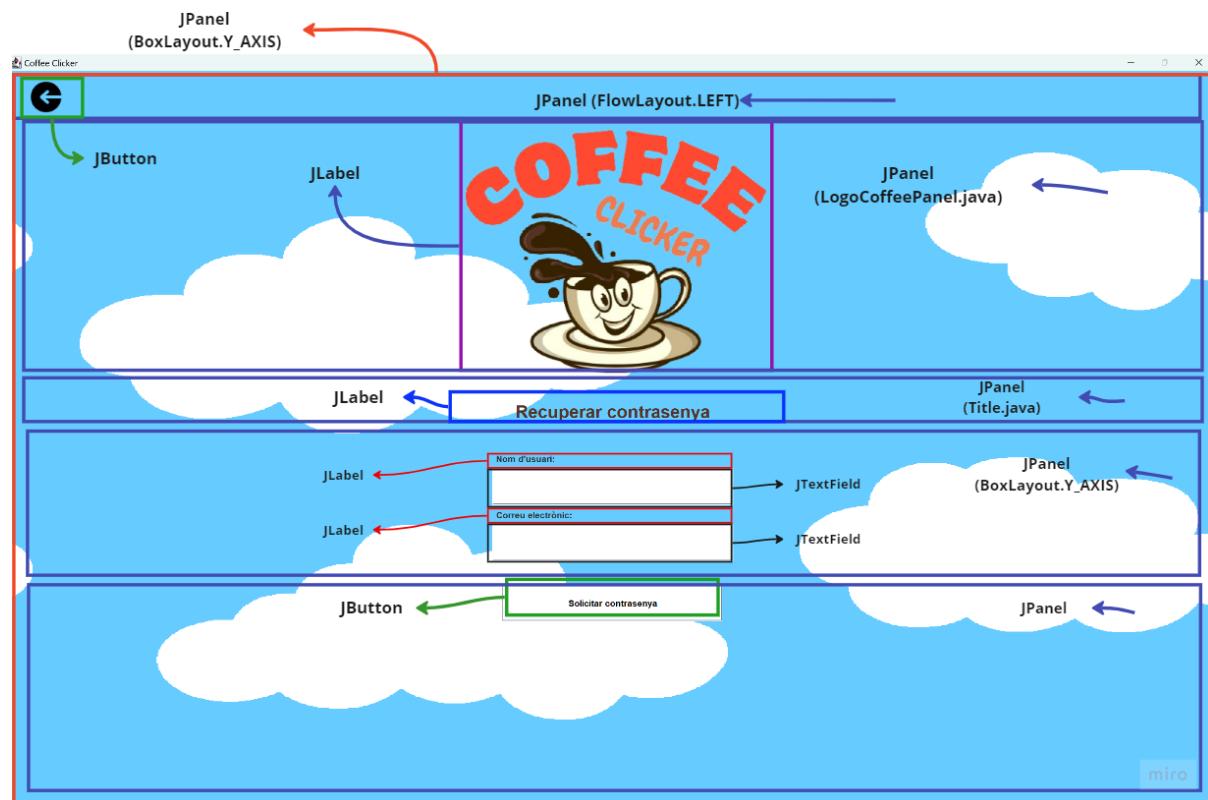
## 2.10 VISTA RECUPERAR CONTRASENYA (OPTATIU)

Aquesta vista permet que l'usuari pugui veure la contrasenya en cas d'haver-se oblidat. Segueix una mica l'estructura de les altres vistes de gestions per l'usuari. A dalt, el logotip i tot seguit a sota, les caselles per omplir la informació necessària.

Un cop l'usuari introduceixi les dades en els camps i premi el botó de sol·licitar, apareixerà un *pop-up* amb la informació, mostrant quina és la seva contrasenya.

S'ha considerat fer que aparegui la contrasenya en comptes de canviar-la per evitar més modificacions a la base de dades. Entenem que tant la base de dades com el sistema de contrasenyes no és massa segur. Així doncs, per facilitar la gestió a l'usuari, apareixerà la contrasenya per pantalla en cas d'introduir les dades correctament.

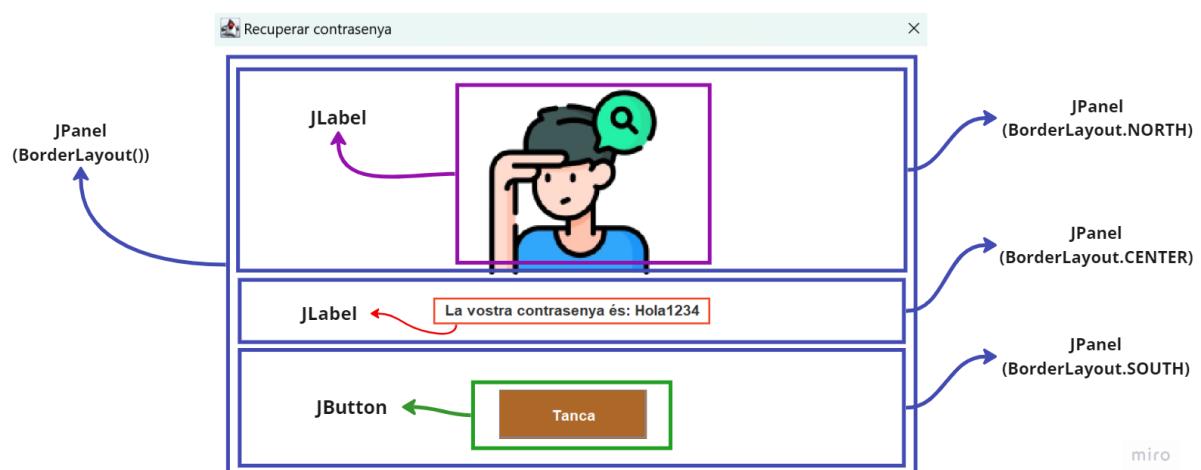
En cas que l'usuari introduceixi una contrasenya incorrecta, es mostra un missatge d'error especificant quin és el camp a corregir i quin ha estat l'error. Aquest error es mostra mitjançant el pop-up utilitzat en altres vistes, el pop-up d'error.



En cas que l'usuari introduceixi correctament tant el nom d'usuari com el correu electrònic, apareix un pop-up similar al que es mostra amb el missatge d'error. No obstant això, en aquest

cas, la dinàmica és una mica diferent, ja que es mostra un assistent que ha buscat la contrasenya a la base de dades per mostrar-la en el missatge.

Una particularitat d'aquest panell és que, en cas que l'usuari no hagi clicat al botó de tancar, per millorar la seguretat, aquest *pop-up* es tanca de forma automàtica després d'uns segons. Un cop tancat el panell, la vista torna a la pantalla d'inici de sessió de manera que l'usuari pugui introduir la informació per accedir al seu compte.



### 3. DIAGRAMA DE CLASSES

En la fase inicial del plantejament i disseny del projecte, un dels punts més importants que vam haver de decidir va ser quin i com hauria de ser el disseny d'arquitectura de software que voldríem aplicar a l'hora de desenvolupar l'aplicació.

#### 3.1 ARQUITECTURA MVC

El model d'arquitectura Model Vista Controlador junt amb l'estructura de capes, ens ha permès estructurar el projecte en blocs on cada un d'ells tenia una responsabilitat dins del projecte per facilitar la gestió i el manteniment del codi. Més endavant explicarem de manera més concreta cadascun d'aquests blocs.

Com hem mencionat abans en l'apartat de disseny de les vistes, cadascuna de les vistes del programa treballa de manera independent a la resta i tant les seves funcionalitats com les interaccions no podien estar relacionades entre elles, ja que podrien dificultar el flux de treball i la possibilitat d'errors no desitjats. Per això, vam decidir que cadascuna de les vistes comptes amb un controlador i *manager* propi per evitar aquest tipus de problemes, respectant al màxim el model d'arquitectura.

Tot i així, durant l'execució ha d'haver-hi un mínim d'interacció entre les diferents vistes per poder implementar funcionalitats com el salt entre pàgines, carregar les dades d'un nou usuari, etc. Per això vam implementar la classe *MainController* que és l'única classe encarregada de gestionar les interaccions entre vistes i és la que té el control sobre el rumb que ha de seguir l'execució del programa i de les vistes en cada moment. Tot i que el *MainController* és qui decideix la vista que s'ha de mostrar a cada moment, aquesta també delega aquesta tasca a la classe *MainFrame* que és l'encarregada de carregar les vistes al JFrame principal del programa quan rep la petició del *MainController*.

Explicades les característiques bàsiques del programa per tal de satisfer les necessitats de l'aplicació respectant al màxim els models d'arquitectura, podem veure que és essencial la delegació de tasques a la seva classe encarregada per tal de facilitar la tasca de desenvolupament. A continuació, explicarem cadascun dels blocs d'aquest patró al detall i com els hem aplicat en aquest projecte.

### 3.2 BLOC PRESENTATION

Aquest és el bloc on situem, principalment, dos dels elements principals del model d'arquitectura, la Vista i el Controlador. En el bloc *Presentation* té com a finalitat emmagatzemar totes aquelles classes que tinguin a veure amb la interacció del usuari amb l'aplicació i l'apartat visual d'aquesta. Dins d'aquest bloc podem ubicar els dos paquets principals del bloc *View* i *Controller*.

#### Bloc View:

En aquest paquet i podem trobar el subpaquet de *Frames* i la classe *MainFrame*. La classe *MainFrame* és la classe que genera la finestra de l'aplicació i l'encarregada de mostrar la *View* corresponent al moment de l'execució. El paquet de *Frame* conté totes les vistes del programa i els següents subpaquets dins:

- **Enums:** Són tots aquells valors que necessitem de manera recurrent per mantenir una mateixa estructura de disseny dins de les vistes. En aquests i podem destacar *ViewStyles* on enumerem els diferents tamanys i colors que pugui tenir la vista, i *TextStyle*, per les diferents mides dels textos.
- **Helpers:** Són totes aquelles classes les quals poden ser reutilitzades per altres vistes, ja que tenen unes funcionalitats específiques i d'ús general, com per exemple classes que et generen el logotip de l'aplicació, títols de l'estil de l'aplicació, etc.

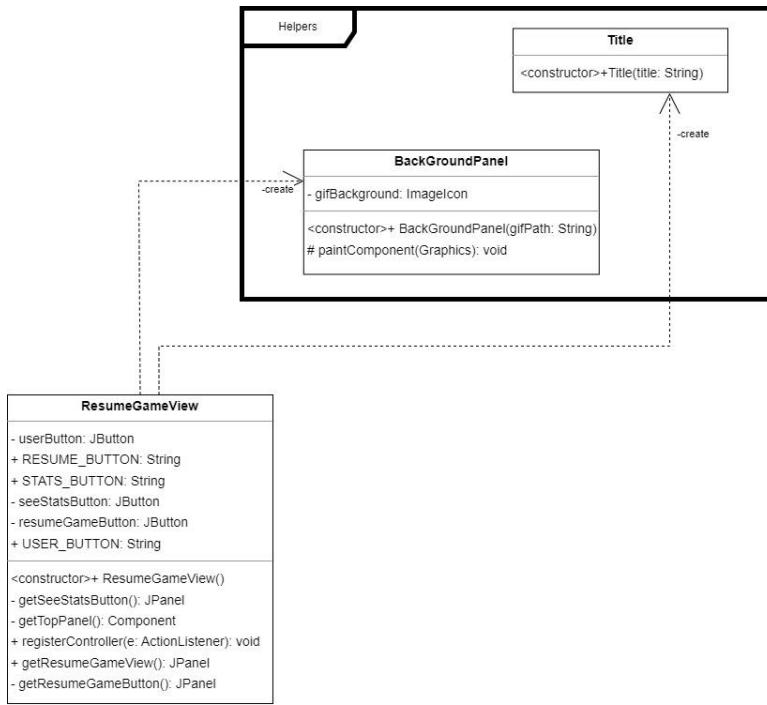
També, podem veure que en aquest paquet, hi podem trobar totes les Vistes del nostre programa, totes les vistes segueixen un mateix patró:

	HomeView
+SIGN_UP: String	
+LOG_IN: String	
-loginButton: JButton	
-signUpButton: JButton	
<Constructor>+HomeView()	
+ getHomeView(): JPanel	
- getLoginButton(): JPanel	
+ getSignUpButton(): JPanel	
- getTopPanel(): Component	
+ registerController(listener: ActionListener): void	
+ getSignUpButtonActionCommand(): Action	

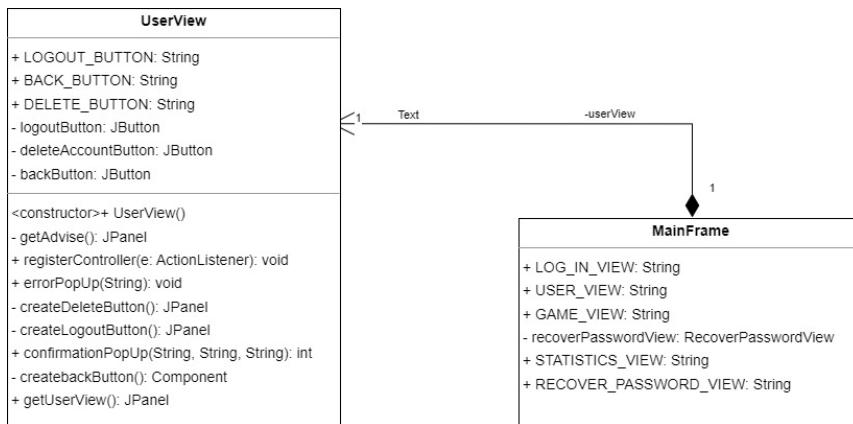
Un exemple de vista del nostre programa seria aquest, aquesta és la vista corresponent a la primera pàgina que se't genera al iniciar el programa, la *HomeView*. Totes les classes de les *Views* fan herència de la classe JFrame de la llibreria JavaFX, ja que més endavant puguin ser afegides pel MainFrame a la finestra principal, a partir de la funció corresponent de *getPanel* de la *View* que pertoqui.

En totes les vistes comptem amb dos tipus d'atributs d'acord amb la seva funcionalitat: En primer lloc, disposem de les constants de tipus *String*, com podem veure en l'exemple, aquestes serien *SING\_UP* i *LOGIN*. Aquestes dues constants ens serveixen per notificar al controlador per via la intereficie *d>ActionListener* quina és la comanda que ha d'executar. I després, tenim els atributs de tots els elements gràfics de la vista els quals necessitem gestionar quan l'usuari faci alguna interacció amb la *View*.

Passem ara a les funcions de la vista, com podem observar a l'exemple, disposem d'un constructor, que ens servirà per instanciar les vistes al *Main*, les diferents funcions *get<elementGràfic>* que ens serveixen per anar generant de manera separada el contingut de la vista, i el mètode *registerController* que ens serveix per guardar l'instància del controlador als diferents elements gràfics de la vista els quals hagin de ser controlats pel *Controller* corresponent de la vista.



Com hem mencionat amb anterioritat, les vistes utilitzen els recursos proporcionats per el paquet **Helpers**. Com podem observar la relació d'aquestes classes és de dependència ja que només són cridades únicament en el moment de la creació, però són estrictament necessàries per la mostrada dins la vista.



Totes les classes de les vistes tenen una relació de composició amb el **MainFrame** com podem veure en el següent bloc inserit. Ja que és el **MainFrame** qui té el control sobre quina és la vista que s'ha de mostrar a cada moment.

\*Mencionar que el **MainFrame** mostrat a l'exemple no reflecteix el **MainFrame** implementat, la finalitat de la mostra serveix per mostrar el tipus de relació.

## Bloc Controller

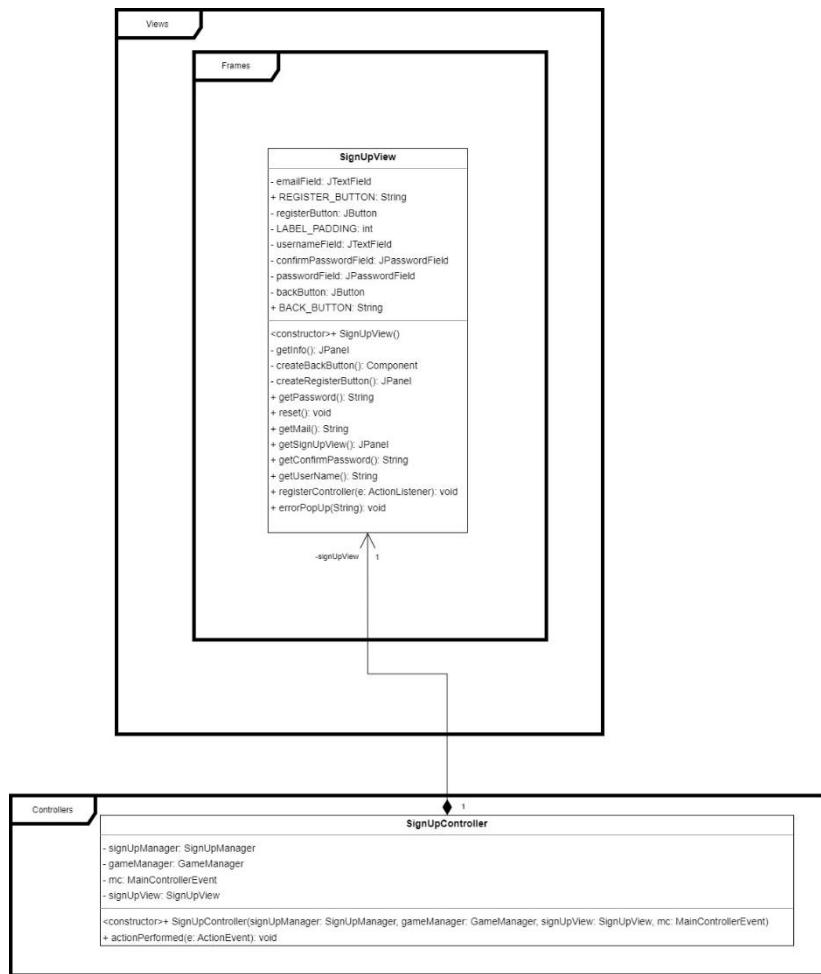
En aquest paquet és on ubicarem les classes encarregades de controlar les interaccions que fa l'usuari i reflectir en la vista tots els canvis en la lògica de l'execució del *manager* corresponent. Com s'ha mencionat amb anterioritat cada vista conté un controlador específic per aquesta i totes aquestes classes se situen dins d'aquest paquet.

<b>SignUpController</b>
<pre>- signUpManager: SignUpManager - gameManager: GameManager - mc: MainControllerEvent - signUpView: SignUpView</pre>
<pre>&lt;constructor&gt;+ SignUpController(signUpManager: SignUpManager, gameManager: GameManager, signUpView: SignUpView, mc: MainControllerEvent) + actionPerformed(e: ActionEvent): void</pre>

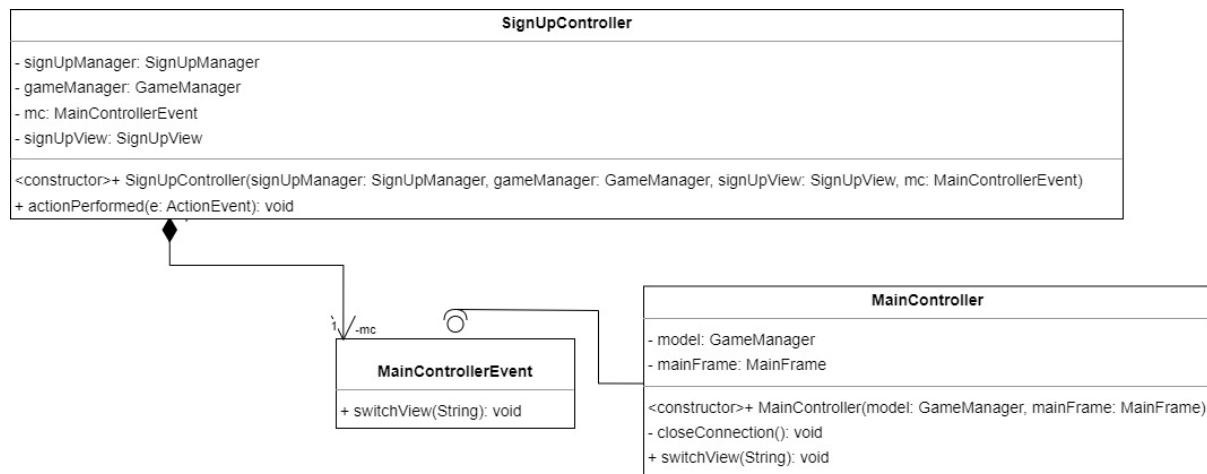
Aquest és el controlador que està relacionat amb la vista de Registre (*SignUp*). Podem observar que té els següents atributs:

- *<Vista>Manager*: Aquest atribut fa referència a la classe Manager de la vista que és l'encarregada d'ubicar tota la lògica de la vista corresponent.
- *GameManager*: Fem servir aquest atribut per guardar tota la informació de l'usuari que és utilitzat per fer les consultes a la font de la persistència de la ubicació. En aquest atribut és on ubiquem també totes les funcions lògiques de la vista Game, que només són utilitzades dins de la vista Game.
- *MainControllerEvent*: En aquest atribut ubiquem el controlador que implementi les funcionalitats de la interfície *MainControllerEvent*, que en aquest cas es tracta del *MainController*. Únicament serà cridat a la classe quan s'hagi de fer un canvi entre les vistes.
- *<Vista>View*: En aquest atribut és on instanciem la vista que el controlador ha de gestionar.

També podem observar que la classe conté dues funcionalitats, una d'aquestes és el constructor, que és cridat únicament a la classe *main*. L'altra funcionalitat es tracta d'un mètode heretat de la interfície *ActionListener* que aquesta es cridarà quan la vista que té associada generi un *ActionEvent*.



Aquesta és la relació de composició entre el controlador i la vista que té associada. Cadascun dels controladors disposa d'aquesta relació entre la seva vista.



Aquí podem veure la relació que tenen els *controllers* amb la interfície i en aquest cas amb la classe *MainController*. Com hem esmentat amb anterioritat, la comunicació que hi ha entre les diferents vistes ha de ser única i exclusivament controlada per via d'aquesta interfície.

### 3.3 BLOC BUSINESS

El bloc de Business és on trobem l'altre element essencial de l'arquitectura MVC, el Model. Aquest bloc té com a objectiu principal emmagatzemar totes aquelles classes relacionades amb la lògica de l'aplicació i la gestió de dades. Dins d'aquest bloc, podem trobar els següents blocs:

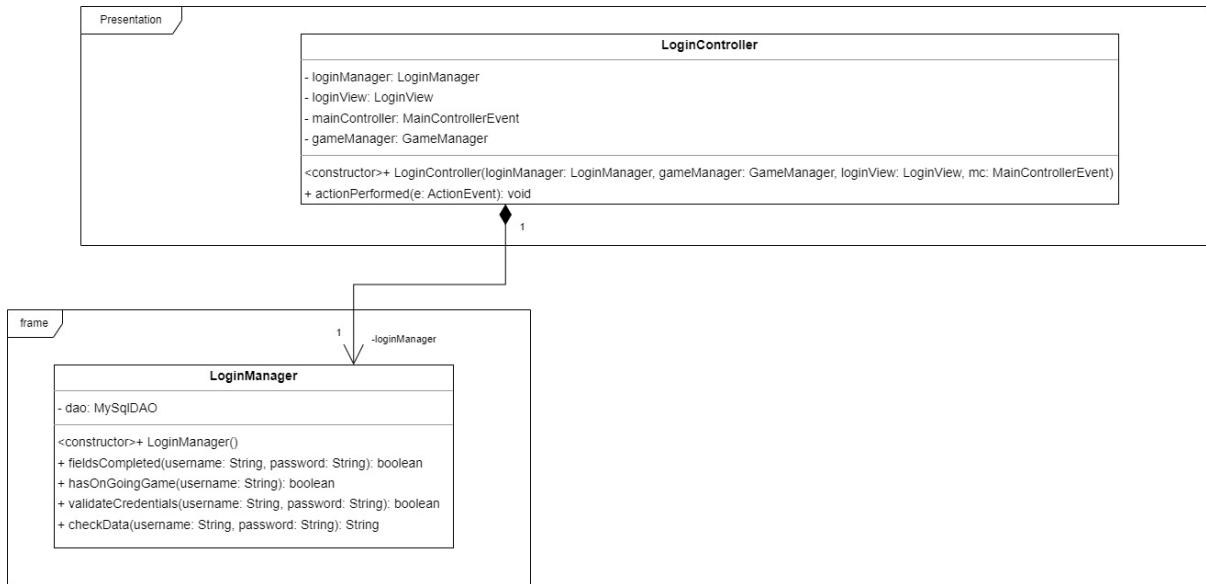
#### Bloc *Entity*

Aquest subpaquet conté totes les classes que defineixen les entitats principals del domini de l'aplicació, en aquest cas *PowerUp*, *Generator* i la classe *Game*. Cada classe representa una entitat i inclou els atributs i mètodes necessaris per gestionar i treballar amb la seva informació.

Les classes situades en aquest paquet són el Model de dades que instanciaran i gestionaran els respectius *Managers* de les diferents Vistes del joc.

LoginManager
- dao: MySqlDAO
<constructor>+ LoginManager()
+ fieldsCompleted(username: String, password: String): boolean
+ hasOnGoingGame(username: String): boolean
+ validateCredentials(username: String, password: String): boolean
+ checkData(username: String, password: String): String

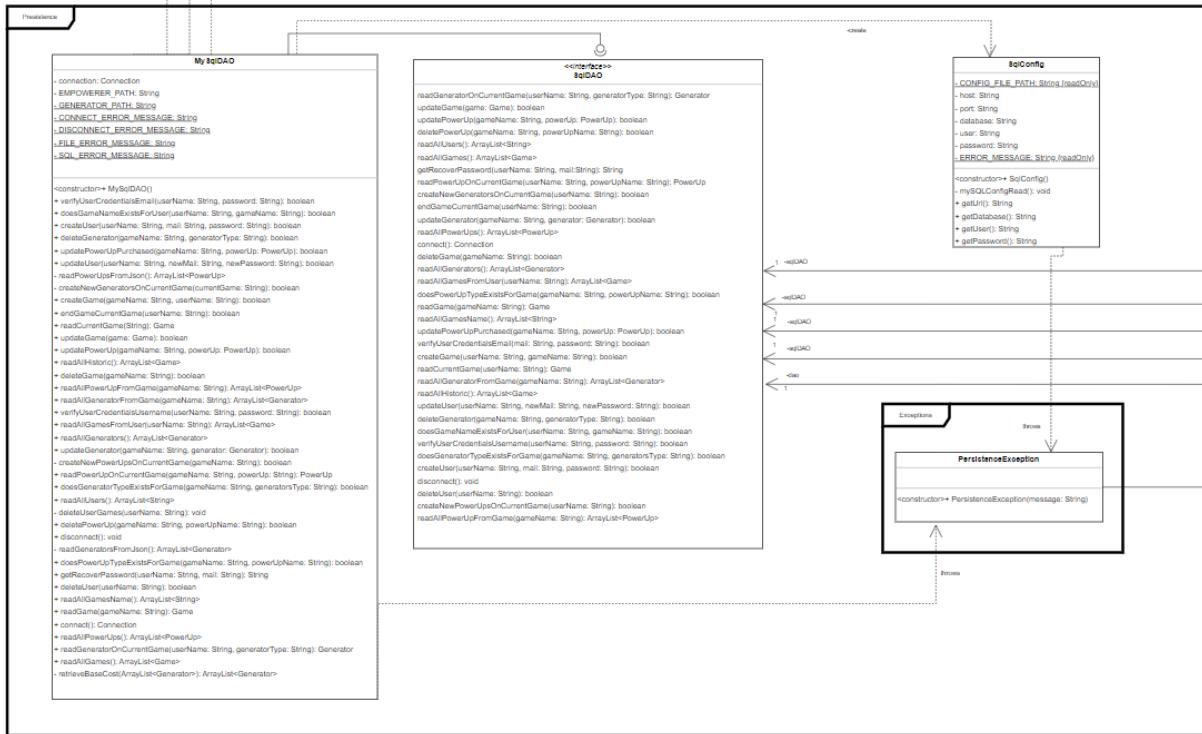
Passem ara a parlar sobre les classes que gestionen tota la lògica de les vistes, els *Managers*. Cadascuna de les vistes disposa d'un *Manager* propi per gestionar tota la informació i operacions que ens demana l'execució del programa. Com podem observar a l'apartat de funcionalitats, disposem de diferents funcions que ens permeten controlar i executar les ordres de cada una de les vistes. També, tots els *managers* tenen un atribut que ens permet enllaçar i consultar les dades que requerim a la font de la persistència, limitant l'accés i l'ús a les classes *Manager*.



El *Controller* gestiona les interaccions de l'usuari i delega la lògica de negoci i la gestió de dades al *Manager*. Aquesta relació de composició assegura que les accions de l'usuari es gestionen de manera efectiva en base a les arquitectures, seguint la lògica definida en els diferents *Managers*.

### 3.4 BLOC PERSISTENCE

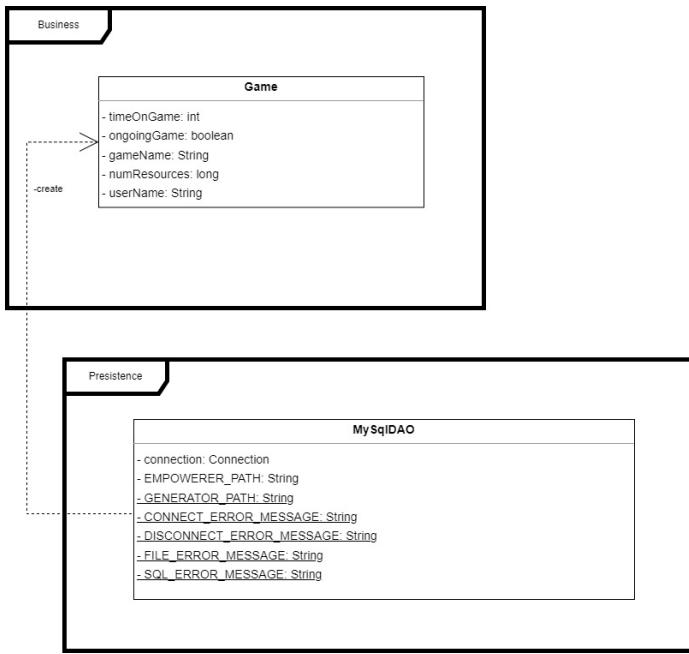
Dins del bloc de *Persistence* hi trobem tot el que està relacionat amb l'emmagatzematge i la recuperació de dades de la nostra aplicació. Aquest bloc és responsable de gestionar com i on es guarden les dades perquè puguin ser accedides i utilitzades més tard pels nostres *Managers* en el bloc de Business. En aquest paquet és on ubicarem les classes d'accés a la base de dades, que s'encarreguen de comunicar-se directament amb la base de dades o amb el nostre fitxer JSON, proporcionant mètodes per crear, llegir, actualitzar i eliminar dades (operacions CRUD). També ubicarem en aquest la classe encarregada de la configuració d'accés de la nostra base de dades.



Aquesta és la totalitat de l'estructura del Paquet de persistència. Com podem observar disposem de 3 classes i una interfície:

La classe *MySqlDao* és l'encarregada de la generació de consultes per a llençar-les a la base de dades, aquesta classe implementa la interfície *SQLDAO* que declara totes les consultes que el *MySqlDao* implementa. Si ens hi fixem, la interfície té diverses relacions d'agregació relacionades amb ella, són les relacions que li arriben des del paquet de Business, més concretament de cadascun dels *Managers* de les Vistes.

Podem observar que, el *MySQLDao* conté una altra relació amb la classe *SqConfig*. Aquesta classe és l'encarregada de la configuració entre el nostre programa i la font de la persistència, que en aquest cas, havíem de fer una connexió amb un servidor extern.



Podem observar també que *MySqlDao* també té una relació de dependència cap al paquet de *Business*, més concretament cap a tots els Models de dades. Això és donat que al cridar les consultes de la base de dades, *MySqlDao* s'encarrega de convertir les dades en un format que, en aquest cas els *Managers* del paquet de *Business*, poguessin treballar de manera més còmoda. En l'exemple de dalt es vol mostrar un exemple del tipus de relació.

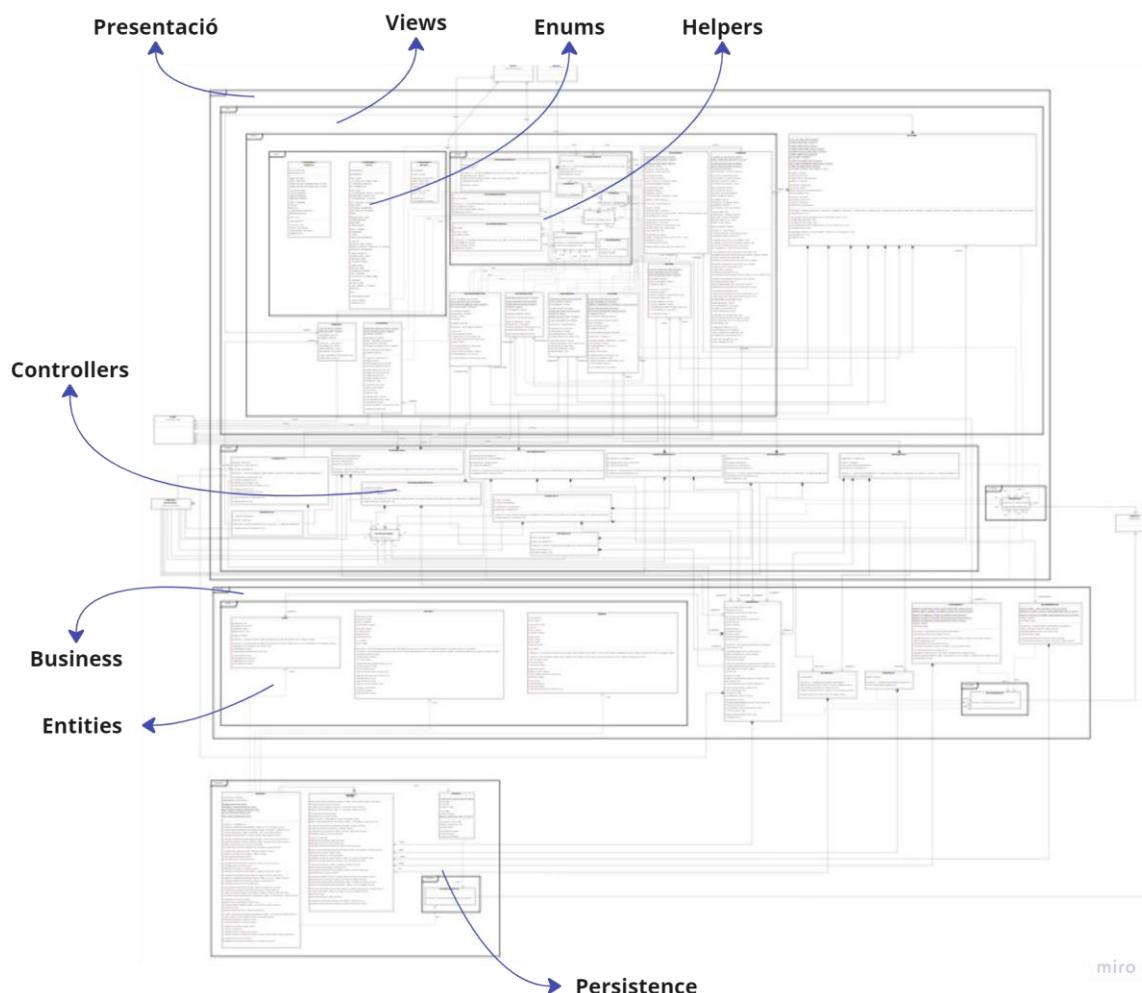
### 3.5 DIAGRAMA FINAL

Finalment, mostrarem el diagrama de classes final del nostre projecte, juntament amb l'enllaç per poder visualitzar-lo de manera més detallada:

[https://app.diagrams.net/#G1PviEYdma\\_HvsAxmu6X-](https://app.diagrams.net/#G1PviEYdma_HvsAxmu6X-)

J50Xy0DqbropZ#%7B%22pageId%22%3A%224DY9U1ki0FE12jfNw\_qm%22%7D

Als annexos es pot veure el PDF amb una resolució adequada per poder veure tots els parts.



## 4. METODOLOGIA DE DESENVOLUPAMENT

El desenvolupament del projecte es va configurar en tres fases de creació interna. La primera fase es va centrar en la creació de l'espai de treball i la configuració de l'estructura inicial del programa, juntament amb la creació de la lògica prèvia a la partida i totes les vistes corresponents a aquesta lògica.

La segona fase es va enfocar en la creació de la lògica de la partida, la vista de la partida i les accions relacionades amb aquesta. Aquesta fase va ser crucial per desenvolupar les funcionalitats principals del joc i garantir que els mecanismes de generació de recursos, tant manual com automàtica, funcionessin correctament.

La tercera fase es va centrar en el tancament de la pràctica, incloent-hi la redacció de la memòria, la depuració del codi i la finalització del diagrama de classes. Durant aquesta fase es va assegurar que tot el codi fos robust i eficient, es van documentar totes les parts del projecte i es van corregir els possibles errors per garantir el bon funcionament del programa en la seva totalitat.

### 4.1 PRIMERA FASE

La primera fase tenia una sèrie de tasques a completar, relacionades amb les històries que s'esperava completar en cada un dels sprints.

- La primera tasca d'aquesta fase va ser la creació i el disseny de la base de dades amb la qual es guardaria tota la informació. Cal destacar que aquesta tasca es va completar a l'inici d'aquesta fase, però es va anar modificant durant el transcurs de la mateixa a causa dels diferents errors que es van trobar en el moment de la implementació del programa.
- La segona tasca inicial va ser la realització del disseny creatiu de l'aplicació, creant el mockup amb la idea inicial de les diferents vistes i definint quina seria la dinàmica a seguir per part de l'usuari.
- La tercera tasca inicial tractava de definir els diferents elements que ens trobaríem durant el transcurs de la partida. Això incloïa definir els generadors, els intensificadors i la temàtica del disseny, així com recopilar imatges i recursos per poder desenvolupar a continuació el codi necessari.

Un cop es van completar aquestes tres tasques, es van realitzar una sèrie de reunions internes del grup per poder definir les següents passes a seguir. En aquestes reunions es va decidir formar dos equips: un de *front-end* i un de *back-end*.

- L'equip de *front-end* es va encarregar de definir i programar les vistes, així com els *pop-ups* que serien necessaris per completar aquesta primera fase. Es va aterrjar les idees del *mockup* en un format de vistes reals i funcionals, assegurant que l'experiència de l'usuari fos intuïtiva i atractiva.
- Per altra banda, l'equip de *back-end* es va centrar en la creació dels controladors corresponents a aquestes vistes, així com en la lògica necessària per poder iniciar una partida i el desplaçament entre les diferents vistes. Aquest equip es va assegurar que les funcionalitats internes del joc funcionessin correctament, permetent una integració fluida amb el *front-end* i garantint una experiència d'usuari coherent i sense interrupcions.

#### 4.2 SEGONA FASE

La segona fase va ser més complexa que la primera, tot i que no tan extensa. Amb les bases establertes en la primera fase, es va dividir la tasca en tres equips, però la separació no va ser tan estricta com en la primera fase, donada la major complexitat de les tasques a realitzar per part de cada equip, cosa que va dificultar la subdivisió de les tasques.

- El primer equip es va centrar en la creació de les vistes, tant de la partida com dels generadors i els potenciadors. Aquestes tasques estaven més enfocades en la generació de les vistes i no tant en la lògica interna del joc.
- El segon equip es va encarregar de configurar la lògica relacionada amb la creació d'una partida nova, la continuació d'una partida ja iniciada, així com la lògica relacionada amb la vista dels recursos actuals i la taula amb la informació dels generadors.
- El tercer equip es va enfocar en la lògica tant dels generadors com dels potenciadors, la qual cosa implicava una dificultat extra. Per aquest motiu, es va decidir destinar un equip específic per crear i implementar aquesta lògica.

#### 4.3 TERCERA FASE

Durant la tercera fase, el focus va estar més en la redacció de la memòria i la depuració del codi, així com en l'eliminació de codi innecessari i la correcció d'errors en l'execució final. A més, es va completar el disseny del diagrama de classes, relacionat amb les correccions fetes durant la depuració del codi.

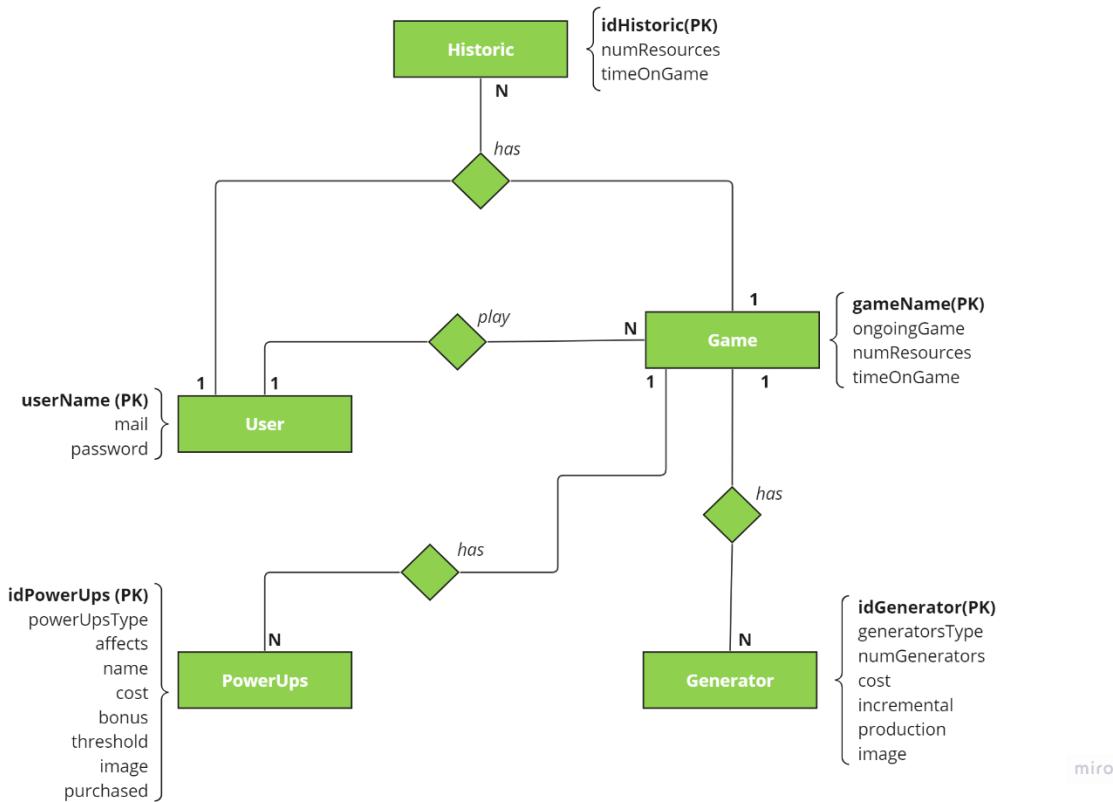
La distribució de la feina va ser equitativa, amb cada membre de l'equip centrat a realitzar les parts corresponents a les seves responsabilitats. Un dels avantatges d'aquesta fase és que es va dur a terme durant el transcurs de tota la pràctica, tot i que en aquest moment final es va dedicar més temps per assegurar que tot estigués complet i depurat adequadament.

#### 4.3 EINES DESENVOLUPAMENT

En el desenvolupament del projecte, s'han utilitzat diverses eines per gestionar tasques, controlar versions i organitzar el treball d'equip. En primer lloc, *Bitbucket* ha estat utilitzat com a plataforma de control de versions, permetent als membres de l'equip col·laborar en el codi font del projecte, gestionar branques i realitzar seguiment dels canvis amb facilitat. *JIRA* ha estat l'eina principal de gestió de projectes, permetent la creació i assignació de tasques, el seguiment del progrés i la prioritització de les tasques pendents. A més, s'ha utilitzat una eina addicional per gestionar subtasques i tasques més detallades, com *Asana*, que ofereix funcionalitats avançades per organitzar i seguir les activitats del projecte de manera més granular.

#### 4.4. DISSENY DE LA BASE DE DADES

Un cop explicat tot el disseny i l'estructura del programa, és important destacar com està estructurada la base de dades i les decisions que s'han pres al respecte. En l'elecció del disseny, s'ha optat per tenir cinc entitats que gestionaran tota la dinàmica del programa.

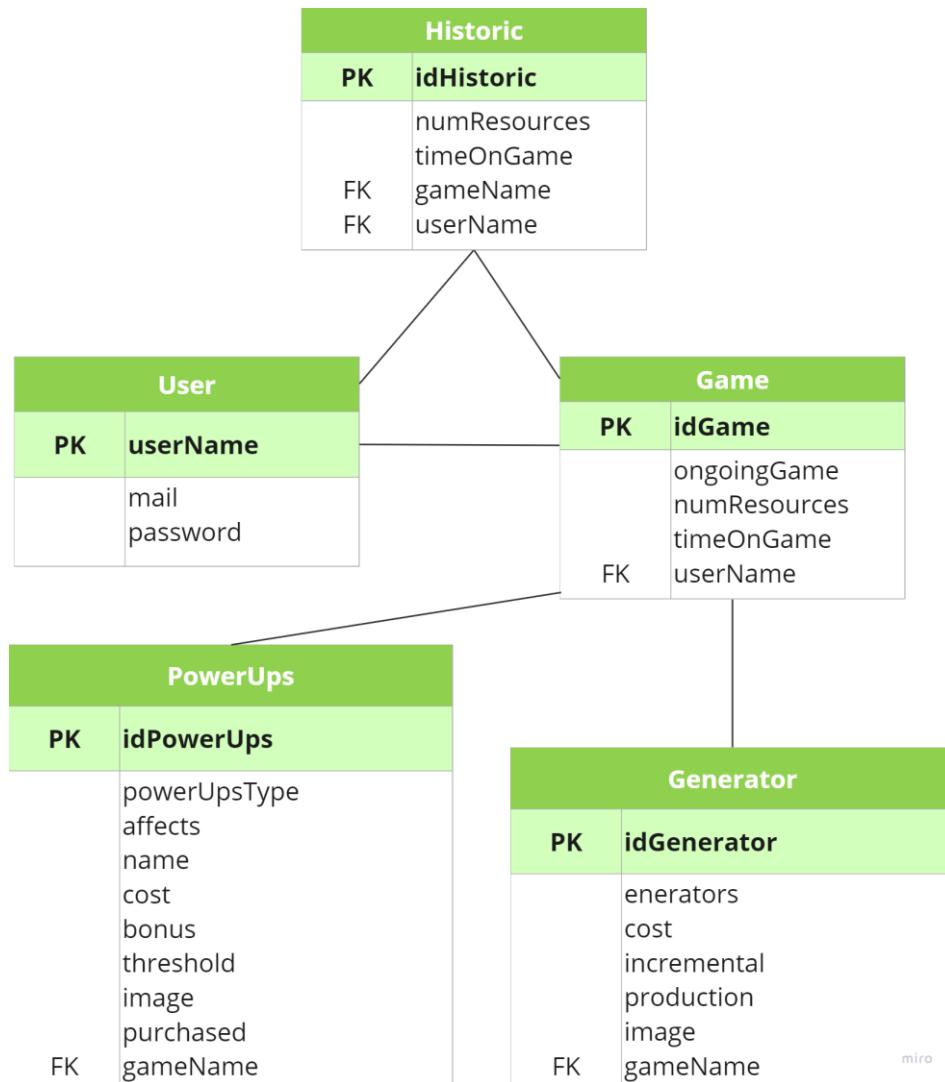


La primera està centrada en els usuaris, on es guardarà tota la informació necessària i servirà com a taula mare per gestionar tota la informació de les partides.

En segon lloc, es pot observar la taula relacionada amb les partides. Aquesta taula no només guardarà la informació relacionada amb l'estat actual de la partida, sinó que també serà la taula mare de les tres següents taules.

A continuació, trobem les taules de generadors i potenciadors, que guarden la informació relacionada amb aquests elements dins la partida.

Per últim, es pot observar la taula d'historic, la qual s'actualitza cada cop que es guarda la partida. Aquesta taula guarda una fotografia de l'estat de la partida en aquell moment, permetent així tenir un històric de l'estat de la partida i poder mostrar les estadístiques corresponents als gràfics.

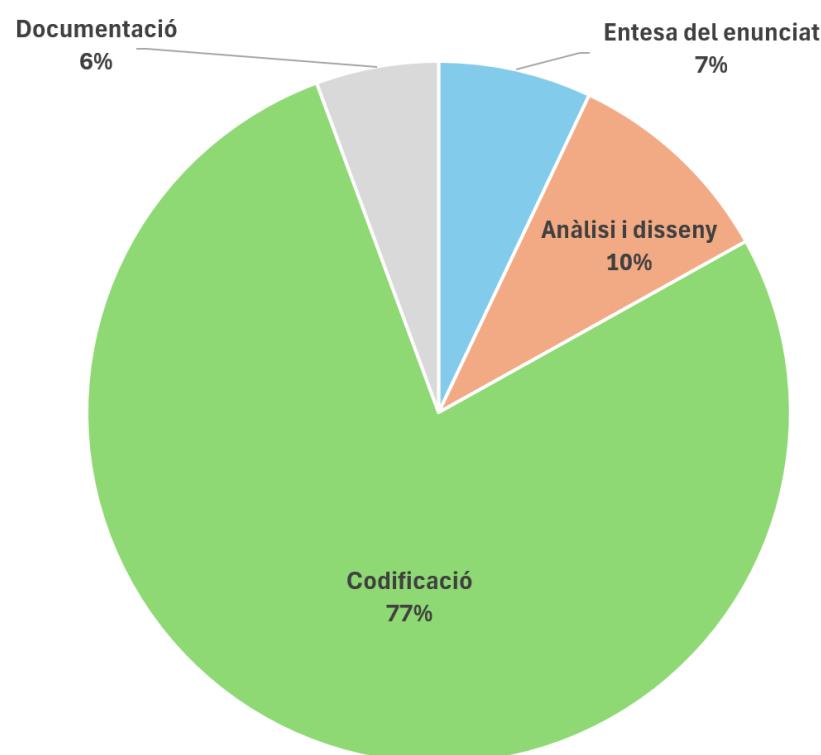




## 5. DEDICACIÓ

En aquesta secció es presenta una taula que mostra la distribució del temps dedicat a cada una de les etapes del projecte. Aquestes etapes inclouen la comprensió de l'enunciat, l'anàlisi i el disseny, la codificació i, finalment, la documentació. Cadascuna d'aquestes etapes ha estat considerada en els quatre apartats per calcular el total d'hores dedicades a la pràctica.

Entesa del enunciat	Anàlisi i disseny	Codificació	Documentació
50 hores	70 hores	550 hores	40 hores



- Entesa de l'enunciat: Aquesta etapa implica la lectura i la comprensió dels requisits i objectius del projecte. En aquest cas, s'han dedicat 50 hores per aquesta tasca, el que reflecteix el temps necessari per entendre completament el que es demana i establir una base sólida per al desenvolupament.
- Anàlisi i disseny: És una fase crucial on s'identifiquen les necessitats dels usuaris i es dissenyen les solucions adequades. Aquí, s'han invertit 70 hores, indicant un temps suficient per planificar i conceptualitzar el projecte abans de començar a programar.

- Codificació: Aquesta és la fase on es converteix el disseny en codi real. Amb 550 hores dedicades a aquesta tasca, es mostra un esforç significatiu per implementar les funcionalitats i les vistes del joc.
- Documentació: La documentació és essencial per comprendre el funcionament del projecte i facilitar la seva futura manutenció i extensió. S'han dedicat 40 hores a aquesta tasca, garantint que s'hagi documentat adequadament el codi i s'hagin elaborat instruccions clares per a futurs desenvolupadors o usuaris.

## 6. CONCLUSIONS

Aquest projecte ha estat una eina d'aprenentatge molt forta per als alumnes del grup, tothom considera que s'han treballat molts aspectes, i que s'ha obtingut un gran coneixement. Podríem destacar molta de l'experiència obtinguda, però essencialment és la següent.

En primer lloc, s'ha destacat la importància de la planificació i el disseny adequats en el desenvolupament d'un projecte de programació. La distribució equilibrada del temps entre les diferents etapes, com l'enteniment de l'enunciat, l'anàlisi i disseny, la codificació i la documentació, ha contribuït a una execució més eficient i un producte final de qualitat.

Durant la implementació del projecte, s'ha posat de manifest la necessitat de flexibilitat i adaptabilitat per abordar els desafiaments que van sorgir al llarg del procés. Això es va reflectir en la redistribució de tasques entre els equips i l'ajust de la planificació segons les necessitats emergents.

La col·laboració i la comunicació efectiva entre els membres de l'equip han estat claus per a l'èxit del projecte. L'assignació equitativa de responsabilitats i la coordinació entre els diferents equips han contribuït a un desenvolupament harmoniós i una integració efectiva de les diferents parts del sistema.

La fase final del projecte, centrada en la depuració del codi i la finalització de la documentació, ha posat de manifest la importància de la revisió i el refinament continu per assegurar la qualitat i la fiabilitat del producte final.

En resum, el treball realitzat ha demostrat l'èxit d'un enfocament metòdic. A través de la planificació adequada, la flexibilitat en l'execució i la comunicació efectiva, s'ha aconseguit un producte final que compleix amb les expectatives i requisits establerts.

## 7. BIBLIOGRAFIA

ypszjnv2013. (2022, 11 de desembre). “Diferencia entre AWT y Swing en Java”. *GeeksforGeeks*. Disponible a: <https://www.geeksforgeeks.org/difference-between.awt-and.swing-in-java/>

JAISWAL, Sonoo. (2015, 13 de novembre). “Java CardLayout”. *Javatpoint*. Disponible a: <https://www.javatpoint.com/CardLayout>

-, Diego. (2018, 2 de maig). “Cambiar tamaño o estilo de botones Java”. *StackOverflow*. Disponible a: <https://es.stackoverflow.com/questions/130285/cambiar-tama%C3%B1o-o-estilo-de-botones-jav>

CRUSOVEANU, Loredana. (2024, 11 de maig). “Create a Custom Exception in Java”. *Baeldung*. Disponible a: <https://www.baeldung.com/java-new-custom-exception>

BEARER, Ring. (2010, 19 de abril). “Java MouseListener”. *StackOverflow*. Disponible a: <https://stackoverflow.com/questions/2668718/java-mouselistener>

OTHMAN, Hasan. (2018, 11 de octubre). “What is the difference between ActionListener and ActionEvent for a button?”. *StackOverflow*. Disponible a: <https://stackoverflow.com/questions/23033439/what-is-the-difference-between-actionlistener-and-actionevent-for-a-button>

## 8. ANNEXOS

