Disseny i programació orientats a objectes

Projecte del segon semestre

Age Royale

Departament d'Enginyeria

La Salle – Universitat Ramon Llull

Grup B10:
Marçal Ausió - marcal.ausio
Jonathan Barragán - jonathan.barragan
Laia Benito - laia.benito
Abel Selfa - abel.selfa
Daniel Valls - daniel.valls

Índex

1. Resum de les especificacions

En aquest projecte es planteja la creació de l'Age Royale, un joc on intervenen dos jugadors, i cada jugador té diferents cartes de tropes per utilitzar-les dins d'un taulell per tal d'eliminar la base enemiga. A més, els usuaris també poden enviar sol·licituds d'amistat, veure partides com espectadors, veure el llistat de partides disponibles. Per tal de fer aquestes funcionalitats i jugar les partides, el client es comunica mitjançant sockets amb el servidor.

El servidor s'encarrega de registrar i autentificar els jugadors, gestionar els amics, tenir la lògica de les partides, és a dir, cap a on es mouen les tropes, la suma de les coins, quan ataquen les tropes, i quan es finalitza la partida. També, el servidor s'encarrega de mostrar l'evolució de les partides respecte el temps i mostrar el top 10 de jugadors amb més partides guanyades.

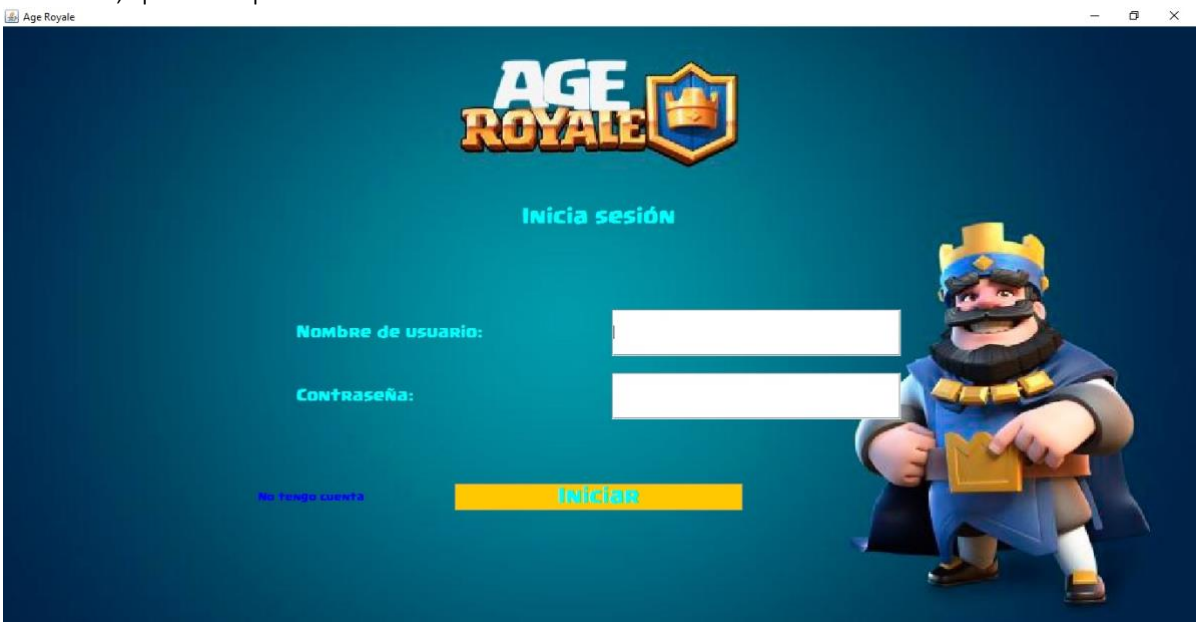
El servidor també s'encarrega d'emmagatzemar les dades dels jugadors i informació del seu historial, com poden ser les partides guanyades i perdudes, per això, el servidor té accés a la base de dades.

El client, a l'hora d'iniciar ha de llegir un fitxer anomenat "config.json" on es troba la direcció IP i el port de connexió amb el servidor.

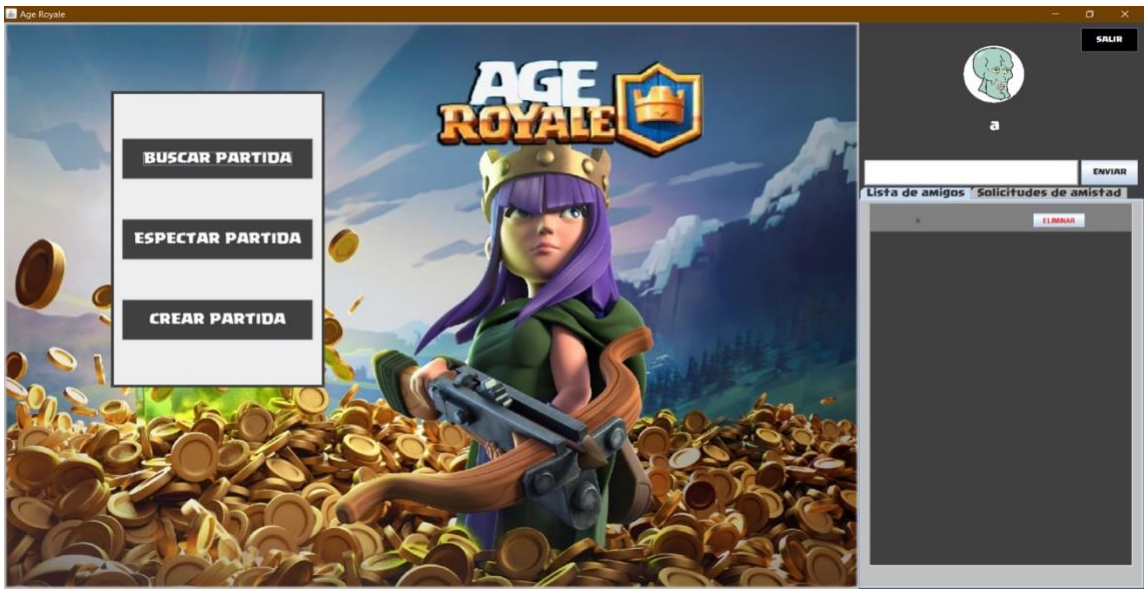
2. Disseny de la interfície gràfica

2.1 Programa Client

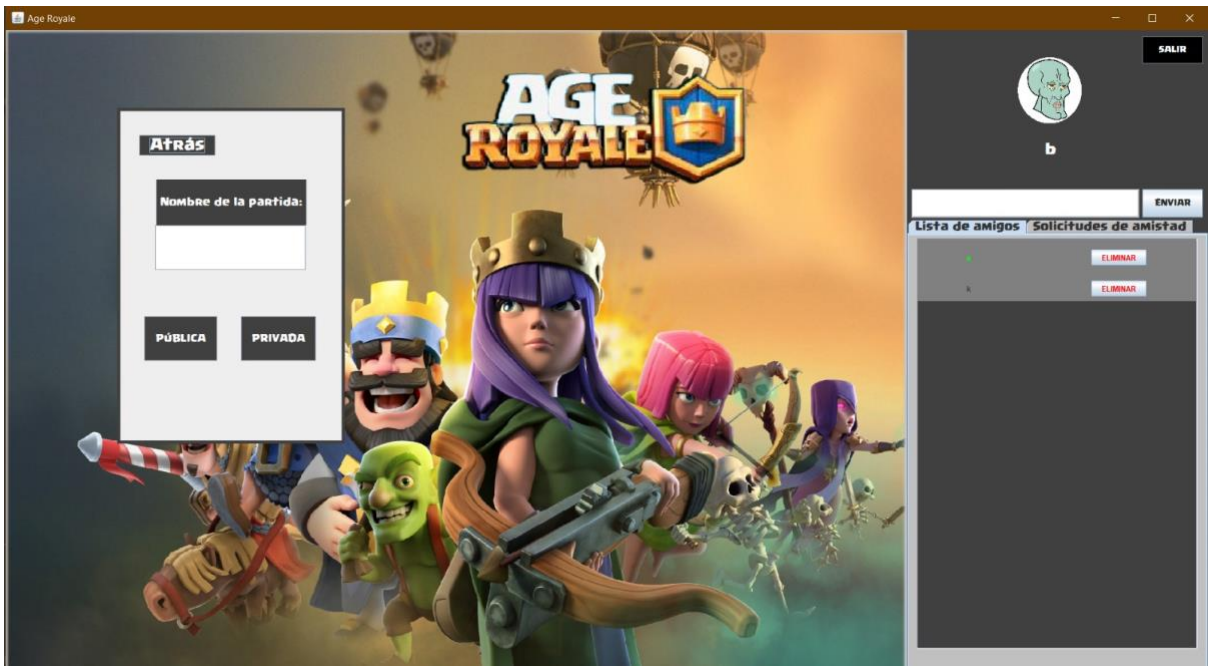
A l'executar el programa client, se'ns obrirà una finestra que ens permetrà registrar-nos en cas que no tinguem un compte creat, o iniciar sessió, en cas que ja en tinguem un. En aquesta finestra hi trobem 2 JButtons que són ("No tengo cuenta", "Iniciar"), també hi ha un JPanel on hi ha tota la informació de la finestra, tenim un JPasswordField, per introduir el password, un JTextField pel nom d'usuari, i 3 JLabel per mostrar el nom d'usuari, contrasenya, i el títol de la finestra, que en aquest cas és "Inicia sesión".



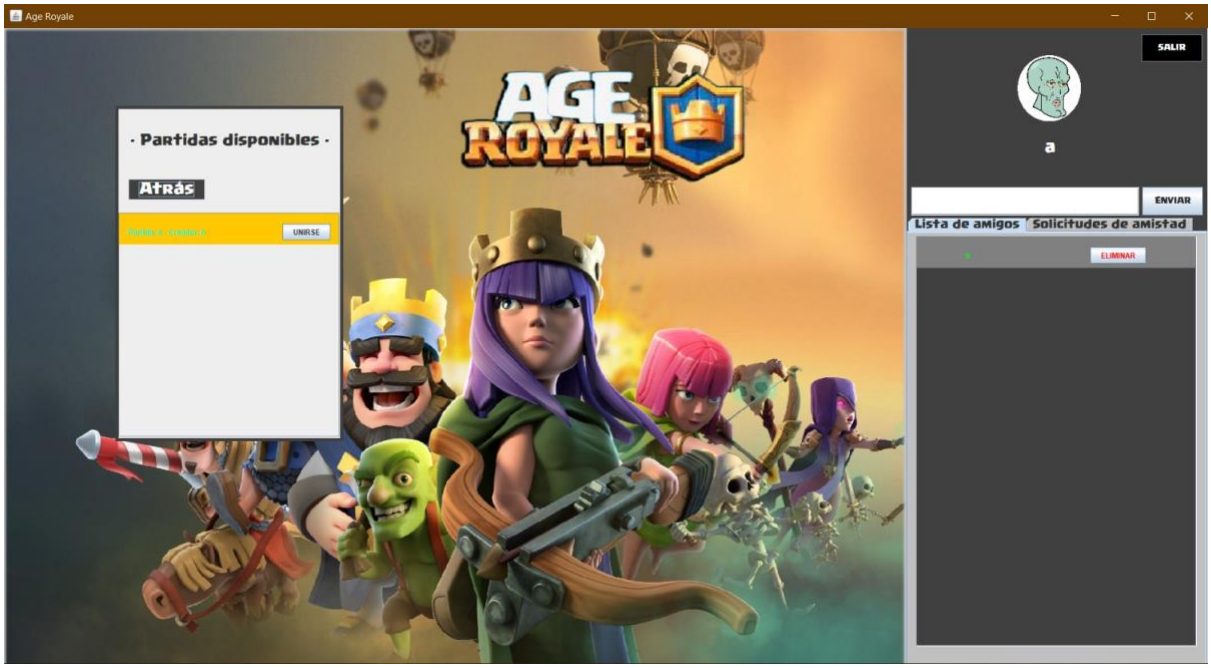
En el menú principal, trobem un JLabel per mostrar el nom d'usuari, també trobem que a la part esquerra hi ha 3 JButton diferents ("Buscar Partida", "Espectar Partida", "Crear Partida"). A la part dreta trobem 2 JButton ("Sortir", "Enviar"), també hi trobem un JPanel on dins hi ha un JScrollPane per mostrar la llista d'amics i un altre JPanel on hi ha un JScrollPane per mostrar les sol·licituds d'amistat.



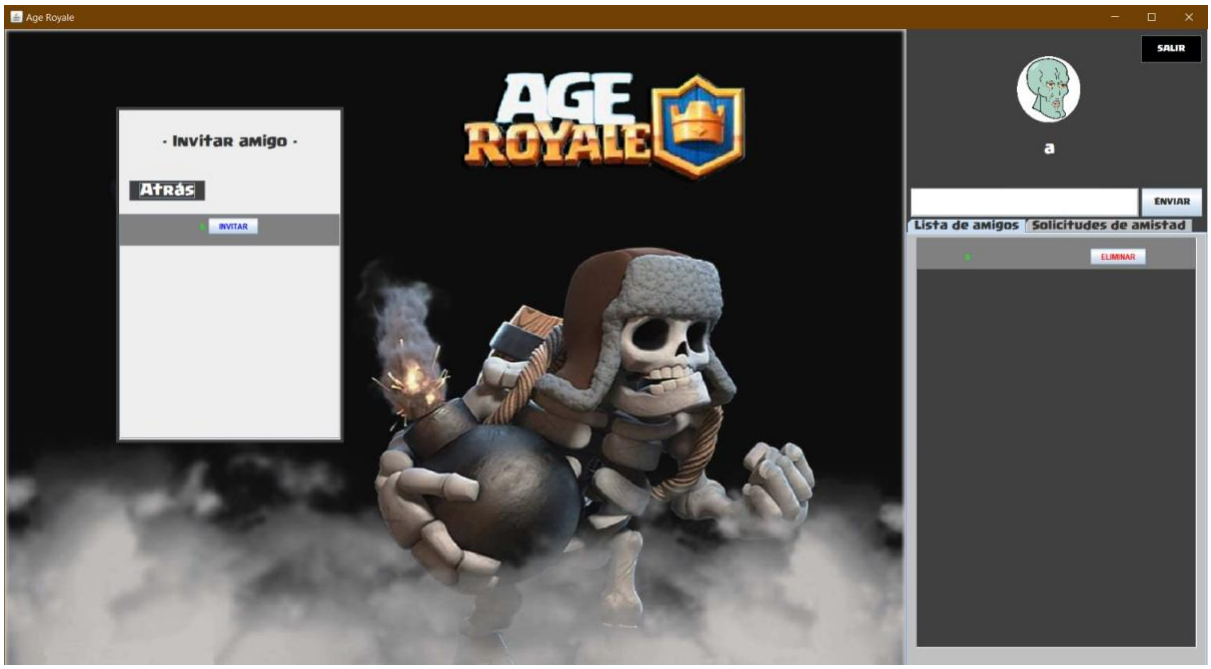
En aquesta pantalla és on es crea la partida. Utilitzem un JPanel per mostrar les diferents opcions, un JTextField per escriure el nom de la partida, dos JButtons, un per escollir si la partida és Privada i un altre per si la partida és Pública. També hi trobem, com a la pantalla anterior, el JPanel per mostrar la llista d'amics i sol·licituds d'amistat.



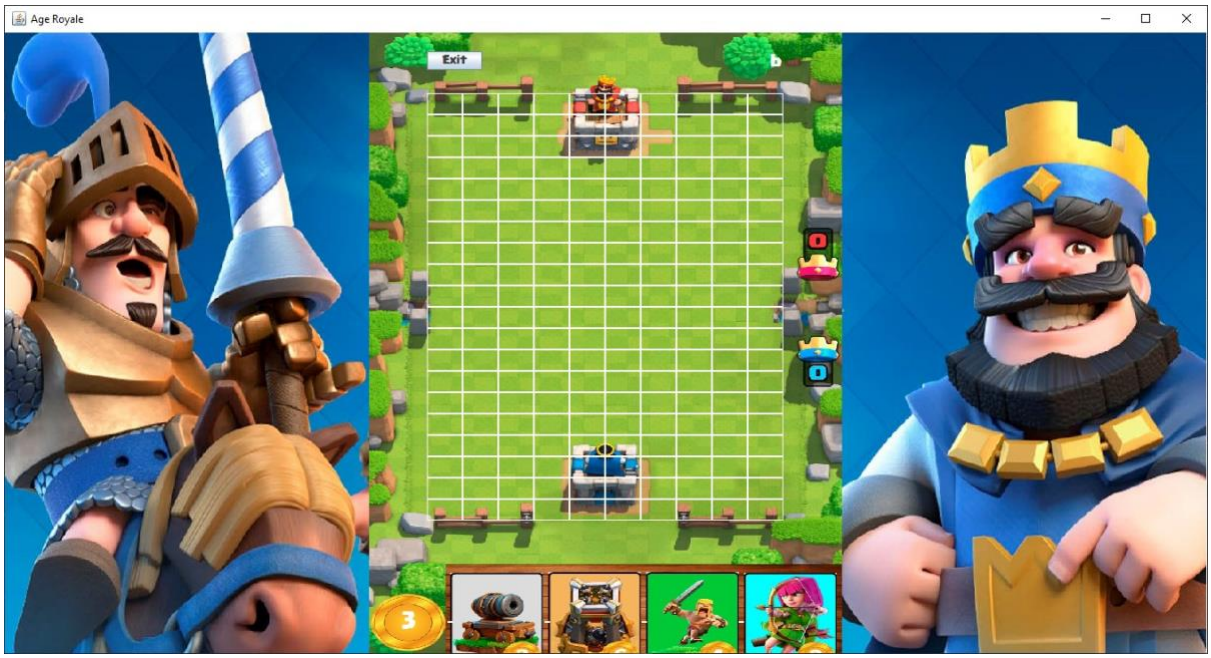
Un cop l'usuari ha escollit l'opció de buscar o espectar partida en el menú principal es mostra aquesta pantalla, on es veu el llistat de partides disponibles. Per fer-ho utilitzem un JPanel on dins hi ha un JButton d'anar enrere, i finalment hi ha un botó per tal d'unir-se a la partida o espectar-la, segons l'opció que s'hagi escollit.



En aquesta pantalla s'hi accedeix quan l'usuari ha escollit l'opció de crear partida privada. Hi trobem un JPanel on dins hi ha un JButton per anar enrere. També tindrem un llistat de tots els amics on cadascú tindrà un JButton que permetrà invitar altres usuaris.



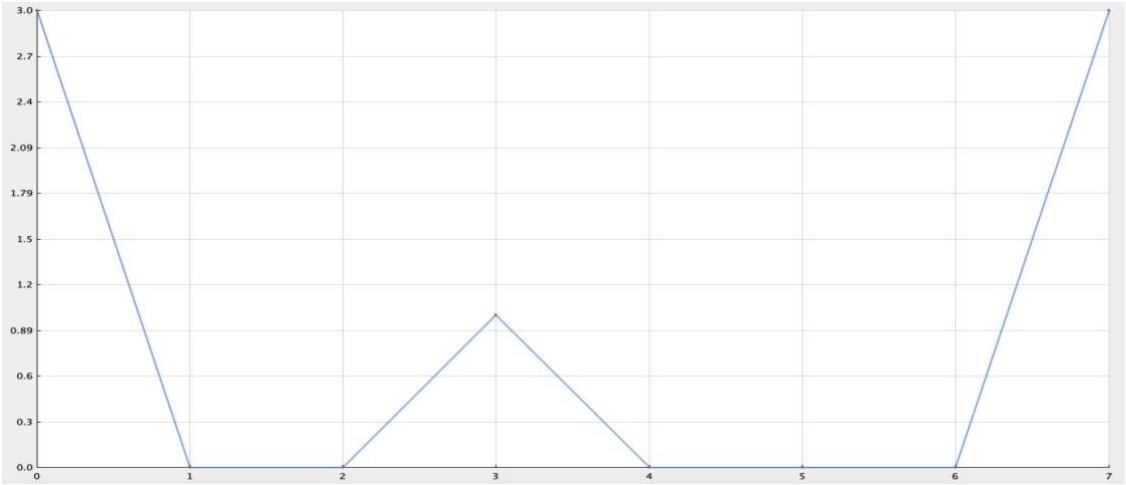
Un cop s'inicia una partida tenim un JPanel on tenim el taulell de joc, també tenim un Jbutton ("Exit"), que serveix per sortir de la partida, i finalment tenim una classe Background per pintar les diferents cartes.



2.2 Programa Servidor

A l'executar el programa del servidor se'ns obrirà un menú (JFrame) on es poden escollir quatre opcions diferents (JButtons): mostrar l'evolució de partides jugades respecte al temps de la darrera setmana, del darrer més o del darrer any o mostrar el ranking del top 10 de jugadors amb més partides guanyades. Cada opció s'obre en un nou JFrame i, per tant, si la tanques, tornes al menú on hi ha les quatre opcions.

En les tres opcions de l'evolució de partides jugades respecte al temps es veu un gràfic de línies amb el nombre de partides que ha jugat. L'usuari pot triar si vol veure el valor de partides jugades de cada dia de la darrera setmana, del mes o de l'any.



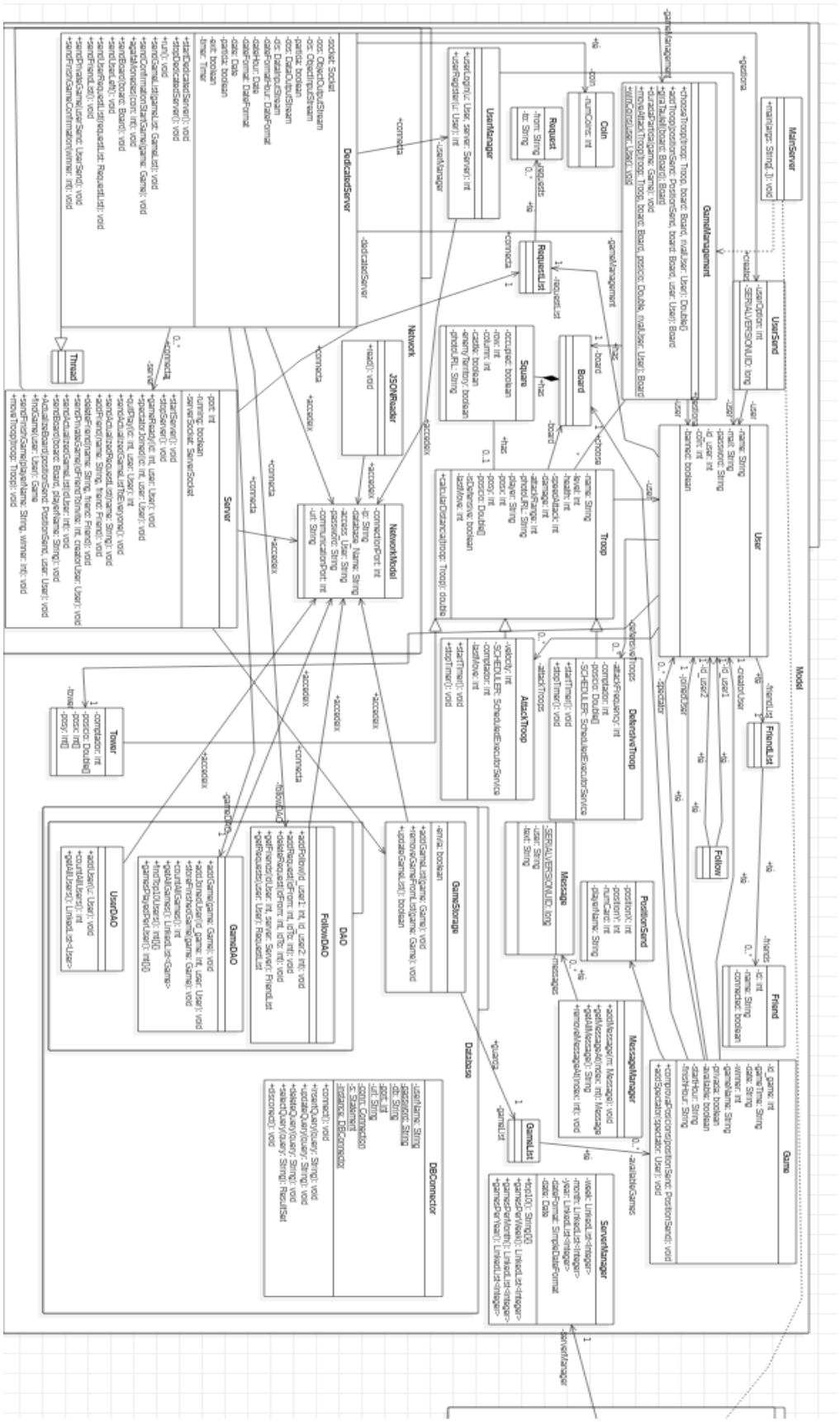
Finalment, tenim la finestra del top 10 de jugadors que més partides han guanyat, mostrant també el percentatge de victòries respecte al total de partides i la mitjana de temps amb la que aconsegueixen guanyar aquestes partides.

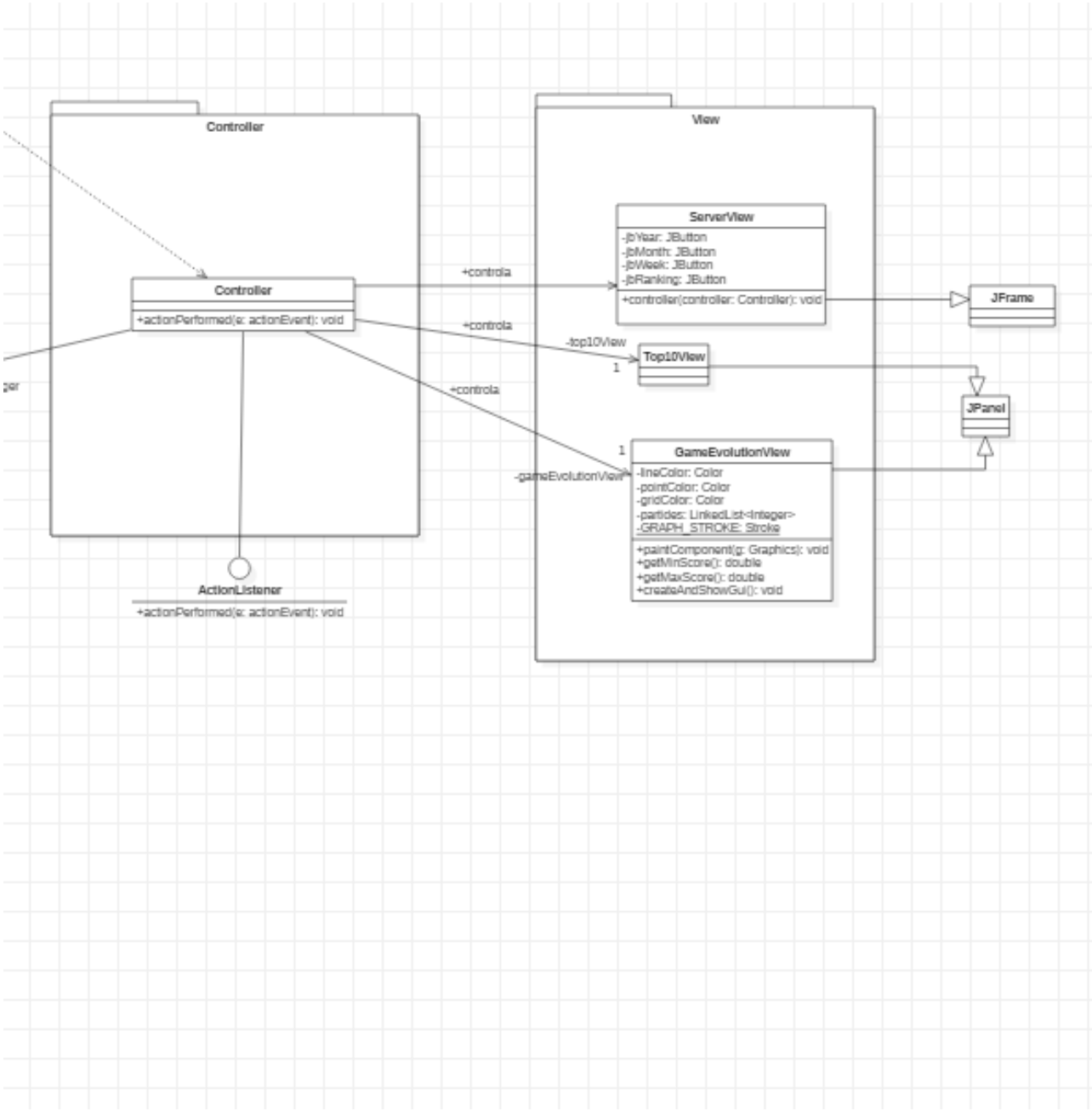
Position	User Name	Number of Games Won	% of Victories	Avg. Time per Game Won
1	laia	3	75.0	22
2	lbb	2	66.66667175292969	32
3	lbb	2	100.0	13
4	lbb	1	100.0	33
5	lbb	1	50.0	23
6	lbb	1	100.0	34
7	l	1	8.333333969116211	20
8	lbb	1	100.0	10
9	lbb	1	100.0	10
10	lbb	1	100.0	30

3. Model de disseny

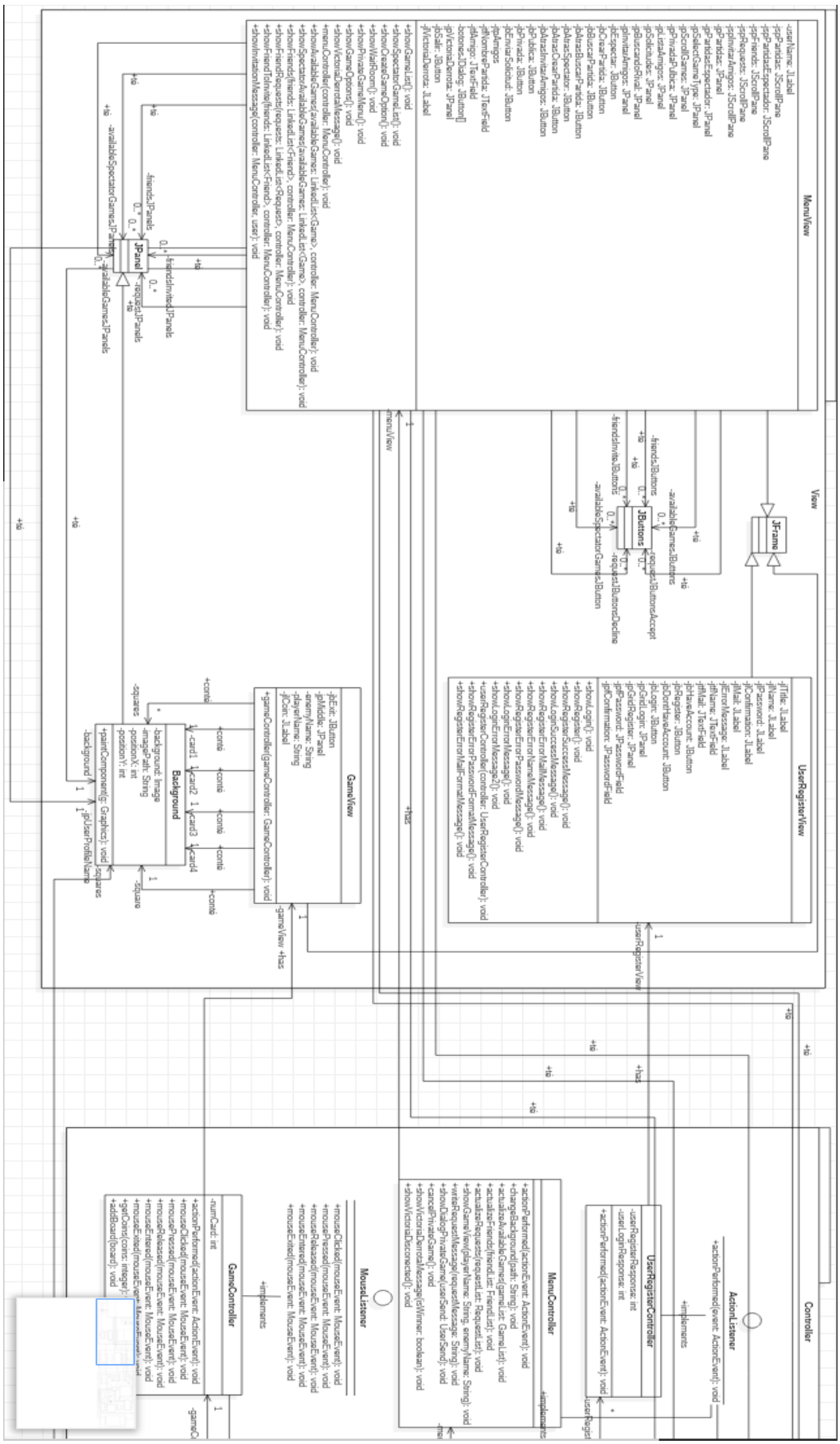
3.1 Diagrama de classes

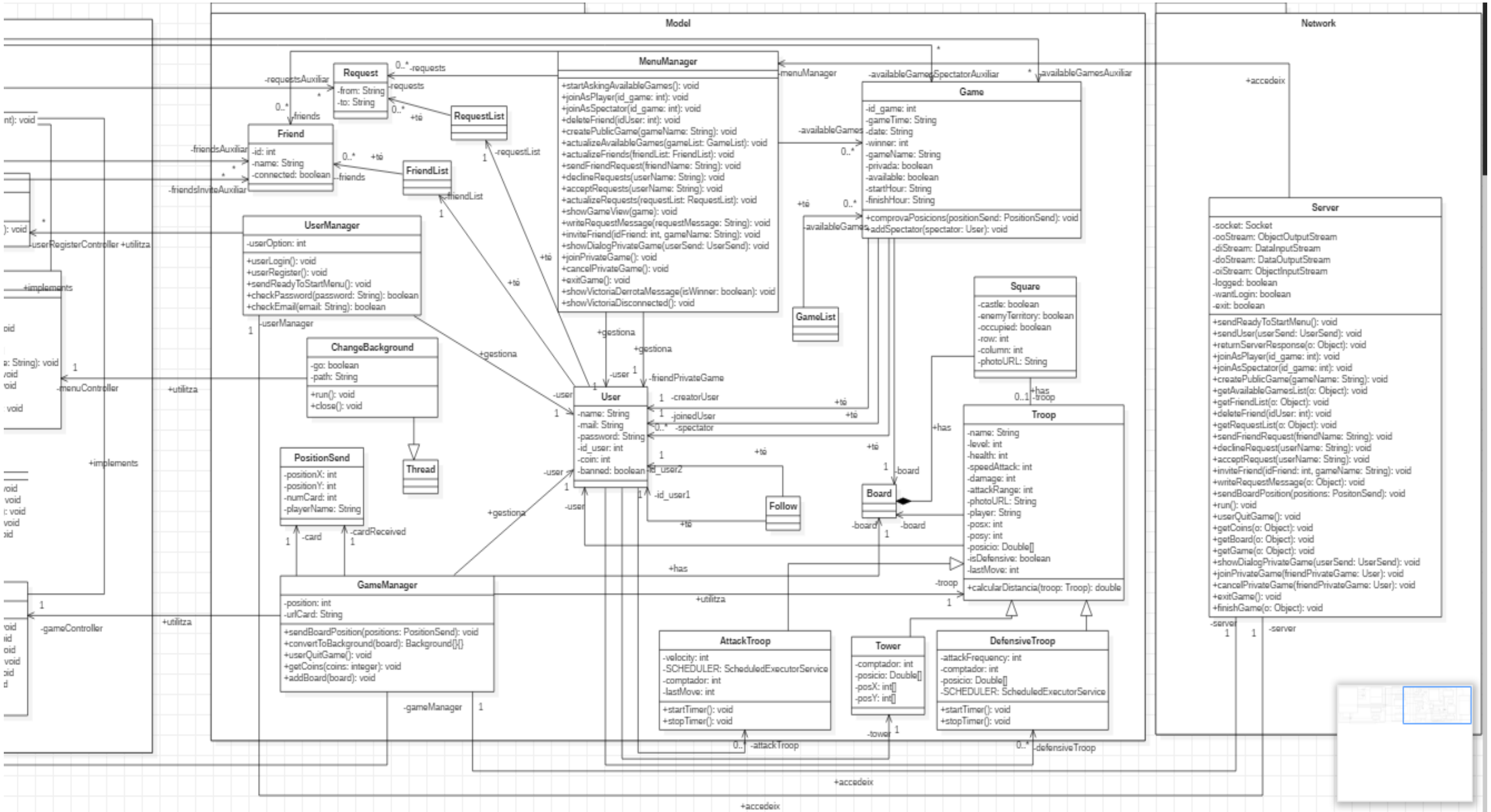
- Servidor




Controller
Controller
+actionPerformed(e: actionEvent): void
ActionListener
+actionPerformed(e: actionEvent): void
+controla
+controla
+controla
View
ServerView
-jbYear: JButton
-jbMonth: JButton
-jbWeek: JButton
-jbRanking: JButton
+controller(controller: Controller): void
-top10View
1
Top10View
1
-gameEvolutionView
GameEvolutionView
-lineColor: Color
-pointColor: Color
-gridColor: Color
-partides: LinkedList<Integer>
-GRAPH_STROKE: Stroke
+paintComponent(g: Graphics): void
+getMinScore(): double
+getMaxScore(): double
+createAndShowGui(): void
JFrame
JPanel

- Client


View
MenuView
UserRegisterView
GameView
Background
JFrame
JPanel
JButtons
Controller
ActionListener
UserRegisterController
MenuController
GameController
MouseListener


Model
Network
Request
-from: String
-to: String
RequestList
Friend
-id: int
-name: String
-connected: boolean
FriendList
UserManager
-userOption: int
+userLogin(): void
+userRegister(): void
+sendReadyToStartMenu(): void
+checkPassword(password: String): boolean
+checkEmail(email: String): boolean
MenuManager
+startAskingAvailableGames(): void
+joinAsPlayer(id_game: int): void
+joinAsSpectator(id_game: int): void
+deleteFriend(idUser: int): void
+createPublicGame(gameName: String): void
+actualizeAvailableGames(gameList: GameList): void
+actualizeFriends(friendList: FriendList): void
+sendFriendRequest(friendName: String): void
+declineRequests(userName: String): void
+acceptRequests(userName: String): void
+actualizeRequests(requestList: RequestList): void
+showGameView(game): void
+writeRequestMessage(requestMessage: String): void
+inviteFriend(idFriend: int, gameName: String): void
+showDialogPrivateGame(userSend: UserSend): void
+joinPrivateGame(): void
+cancelPrivateGame(): void
+exitGame(): void
+showVictoriaDerrotaMessage(isWinner: boolean): void
+showVictoriaDisconnected(): void
Game
-id_game: int
-gameTime: String
-date: String
-winner: int
-gameName: String
-privada: boolean
-available: boolean
-startHour: String
-finishHour: String
+comprovaPosicions(positionSend: PositionSend): void
+addSpectator(spectator: User): void
GameList
ChangeBackground
-go: boolean
-path: String
+run(): void
+close(): void
Thread
User
-name: String
-mail: String
-password: String
-id_user: int
-coin: int
-banned: boolean
Follow
PositionSend
-positionX: int
-positionY: int
-numCard: int
-playerName: String
GameManager
-position: int
-urlCard: String
+sendBoardPosition(positions: PositionSend): void
+convertToBackground(board): Background[][]
+userQuitGame(): void
+getCoins(coins: integer): void
+addBoard(board): void
Square
-castle: boolean
-enemyTerritory: boolean
-occupied: boolean
-row: int
-column: int
-photoURL: String
Board
Troop
-name: String
-level: int
-health: int
-speedAttack: int
-damage: int
-attackRange: int
-photoURL: String
-player: String
-posx: int
-posy: int
-posicio: Double[]
-isDefensive: boolean
-lastMove: int
+calcularDistancia(troop: Troop): double
AttackTroop
-velocity: int
-SCHEDULER: ScheduledExecutorService
-comptador: int
-lastMove: int
+startTimer(): void
+stopTimer(): void
Tower
-comptador: int
-posicio: Double[]
-posX: int[]
-posY: int[]
DefensiveTroop
-attackFrequency: int
-comptador: int
-posicio: Double[]
-SCHEDULER: ScheduledExecutorService
+startTimer(): void
+stopTimer(): void
Server
-socket: Socket
-ooStream: ObjectOutputStream
-diStream: DataInputStream
-doStream: DataOutputStream
-oiStream: ObjectInputStream
-logged: boolean
-wantLogin: boolean
-exit: boolean
+sendReadyToStartMenu(): void
+sendUser(userSend: UserSend): void
+returnServerResponse(o: Object): void
+joinAsPlayer(id_game: int): void
+joinAsSpectator(id_game: int): void
+createPublicGame(gameName: String): void
+getAvailableGamesList(o: Object): void
+getFriendList(o: Object): void
+deleteFriend(idUser: int): void
+getRequestList(o: Object): void
+sendFriendRequest(friendName: String): void
+declineRequest(userName: String): void
+acceptRequest(userName: String): void
+inviteFriend(idFriend: int, gameName: String): void
+writeRequestMessage(o: Object): void
+sendBoardPosition(positions: PositonSend): void
+run(): void
+userQuitGame(): void
+getCoins(o: Object): void
+getBoard(o: Object): void
+getGame(o: Object): void
+showDialogPrivateGame(userSend: UserSend): void
+joinPrivateGame(friendPrivateGame: User): void
+cancelPrivateGame(friendPrivateGame: User): void
+exitGame(): void
+finishGame(o: Object): void
+accedeix
+gestiona
+utilitza
+implements
+té
+has

3.2. Descripció

- Classes del servidor

La part del servidor s'ha dividit segons el patró de disseny Model-View-Controller. Tenim diferents packages que serveixen per organitzar el codi per tal que quedi molt més clar.

Package Controller

Controller: *Classe que ens serveix com a Controller del servidor, es a dir, per relacionar el model amb la vista*

Package Model

Package Database

Package DAO

FollowDAO: *Classe que ens serveix per accedir a la base de dades i extreure informació sobre les Amistats*

GameDAO: *Classe que ens serveix per accedir a la base de dades i extreure informació sobre les partides*

UserDAO: *Classe que ens serveix per accedir a la base de dades i extreure informació sobre els usuaris*

DBConnector: *Classe que ens serveix per accedir a la base de dades*

GameStorage: *Classe que ens serveix per manipular la llista de partides*

PackageNetwork

DedicatedServer: *Classe DedicatedServer que serveix per connectar-se amb el servidor i enviar informació al client*

JSONReader: *Classe que utilitzem per llegir el JSON i omplir el NetworkModel amb la informació extreta*

NetworkModel: *Classe on guardem la informació del JSON sobre la connexió*

Server: *Classe Servidor amb la que ens comuniquem amb els clients*

GameManagement: *Classe que utilitzem per les funcionalitats de la partida*

ServerManager: *Classe que ens serveix per les funcionalitats de la vista del servidor*

UserManager: *Classe que serveix per actualitzar la informació dels usuaris*

Package View

GameEvolutionView: *Classe que serveix per a mostrar els gràfics de la evolució de partides respecte el temps*

ServerView: *Classe que forma la vista del servidor amb els 4 botons que ens serviran per saver que vol veure l'usuari*

Top10View: *Classe que crea la taula amb el top10 d'usuaris que mes partides han guanyat i la mostra*

- Classes del client

L'estructura del programa ha estat formada utilitzant el patró de disseny Model-View-Controller. Hem dividit el programa en diferents packages independents que s'encarreguen cadascun de les seves respectives tasques.

Package Controller

Game Controller: *Controlador del joc que implementa Listeners per poder utilitzar a la vista del joc*

MenuController: *Controlador del menu que implementa Listeners per poder utilitzar a la vista del menu*

UserRegisterController: *Controlador de la pantalla de registre que implementa Listeners per poder utilitzar a la vista del menu*

Package Model

ChangeBackground: *Classe que ens serveix per canviar el fons del menu*

GameManager: *Manager del joc que ens serveix per incialitzar totes les variables necessaries per la partida*

MenuManager: *Manager del menu que ens serveix per incialitzar totes les variables necessaries pel menu*

UserManager: *Manager del registre d'usuari que ens serveix per incialitzar totes les variables necessaries pel registre/login*

PackageNetwork

Server: *Classe Servidor del Client, la qual es permet comunicar-nos mab el servidor i intercanviar informació*

Package View

Background: *Classe Backgrouond de la vista, que ens serveix per pintar la imatge de fons que va canviant*

GameView: *Aquesta classe ens serveix per la vista del joc*

MenuView: *Classe que ens serveix per crear la vista del menu desde el qual podem unir-nos a una partida, crear una partida, afegir amics, borrar amics, etc.*

UserRegisterView: *Classe que ens serveix per crear la vista inicial del programa, on l'usuari pot registrar-se o iniciar Sessió*

- Classes Shared

Package Entity

AttackTroop: *Classe que hereda de Troop ja que tenim dos tipus de tropes: ofensives i defensives. En les ofensives afegim velocitat i un comptador per tal de que es mogui*

Board: *Classe que guarda la matriu de Squares que compren el Taulell*

Coin: *Classe que guarda el nombre de monedes actuals*

DefensiveTroop: *Classe que hereda de Troop ja que tenim dos tipus de tropes: ofensives i defensives. En les defensives afegim la freqüència d'atac, la posició d'aquesta i un comptador per saber cada quan ataca*

Follow: *Classe que guarda els dos usuaris que es segueixen*

Friend: *Classe que guarda l'id, el nom i si està connectat un amic*

FriendList: *Classe que guarda la llista d'amics d'un usuari*

Game: *Classe que guarda tota la informació referent a la partida*

GameList: *Classe que guarda la llista de partides disponibles*

PositionSend: *Classe que guarda la posició i la carta a mes del nom de l'usuari*

Request: *Classe que guarda de qui a qui va una sol·licitut d'amistat*

RequestList: *Classe que guarda la llista de sol·licituts d'amistat*

Square: *Classe que guarda el que te cada casella del taulell*

Tower: *Classe que guarda la informació sobre la torre*

Troop: *Classe que guarda tota la informació de cada tropa*

User: *Classe que guarda tota la informació de l'usuari, desde la del compte del joc fins a la informació dintre de la partida*

UserSend: *Classe que guarda l'usuari a enviar i la opcio que serà el que l'usuari vol fer*

4. Metodologia de desenvolupament

Per a realitzar aquest projecte el que vam fer va ser seguir el planning que ens havien establert a classe de repartir el projecte en 4 sprints diferents, de manera que poguéssim mantenir un ritme de treball constant i ens anés millor dividir-nos la feina.

Al principi de tot vam decidir organtizar-nos en dos grups per tal que uns fessin el diagrama del servidor i els altres el del client. Un cop acabats els diagrames els vam ficar en comú i vam acabar de retocar petits errors de disseny.

Un cop acabada aquesta primera fase del treball van començar les tasques de programació. Aquí sí que durant els dos primers sprints cadascú va fer una tasca de les quals hi havia penjada al jira, ja que tenien una quantitat de càrrega de treball assequible i era fàcil d'ajuntar i repartir un cop acabada cada tasca. Però un cop va arribar la part del game management es va complicar i vam decidir ficar-nos tres membres del grup a intentar desenvolupar-lo, ja que creiem que era una part molt important del projecte i probablement la més complexa. Evidentment també vam tenir problemes amb altres parts del projecte que van fer que de mica en mica se'ns anés endarrerint una mica.

Un cop acabada la fase d'implementació vam passar a fer la part de documentació, que consistia en fer la memòria i fer els retocs necessaris a la part dels diagrames, que van ser bastants, ja que les idees que teníem en un principi no eren del tot dolentes però ens faltaven molts detalls que no ens havíem plantejat.

5. Temps dedicat

El projecte s'ha vist molt influenciat pel tema del coronavirus. El fet d'estar tots confinats a casa ha fet que coses com resolucions de dubtes o trobar-nos tot el grup per posar les diferents tasques en comú es fessin més complicades i, per tant, hi perdéssim més temps.
Pel que fa a la comprensió de les especificacions ens vam ficar tot el grup sencer a mirar l'enunciat durant les dues primeres classes així que més o menys van ser unes 4 hores.
La part de codificació és, evidentment, la que més hores ens ha portat. Com que la major part del projecte ha estat dividida en diferents tasques per cada un és una mica difícil de saber quantes hores hi hem dedicat grupalment, però creiem que poden ser unes 60 aproximadament.
La part d'anàlisi i disseny vam decidir fer-la en dos grups, uns feien el diagrama del servidor i els altres els del client. Després vam posar en comú els diagrames i vam acabar de polir alguns detalls i fer el diagrama de la BBDD. Un cop acabat el projecte hem hagut de modificar algunes parts dels diagrames, ja que algunes idees que teníem en un principi no eren del tot efectives.
Finalment, la part de la documentació hem trigat unes 8 hores, ja que s'han hagut d'explicar moltes coses i tenir-ho tot ben presentat.

6. Conclusions

Després d'haver realitzat el projecte AgeRoyale ens hem adonat de la complexitat que té elaborar un projecte complet des de zero i amb moltes diferents funcionalitats. La divisió del projecte en 4 sprints és un fet que ajuda bastant a mantenir un ritme de treball constant i va molt bé per portar un bon seguiment. Creiem que el repartiment de les tasques per a cada sprint va ser correcte. Tot i això, la quantitat de pràctiques d'altres assignatures i el fet que amb el confinament s'augmentés la càrrega de treball d'avaluació continua en determinades matèries, ha fet que algunes setmanes no poguéssim dedicar-hi tot el temps necessari.
Per poder controlar bé les diferents versions del nostre codi hem utilitzat Bitbucket, fet que ens ha permès poder pujar el nostre codi i poder-lo actualitzar fàcilment per després baixar-lo i així tenir actualitzat el codi que anàvem fent els diferents membres del grup.
El projecte ha anat molt bé per consolidar temes com el model Model-View-Controller i les connexions amb sockets.
També hem pogut aplicar aspectes d'una altra assignatura com és Bases de Dades, ja que hem hagut de crear taules, amb els seus PK i FK corresponents. A més, hem hagut d'establir la connexió entre la base de dades i Java mitjançant la classe "DBConector", amb l'ús de JDBC, de manera que hem pogut relacionar dues matèries diferents.
Un altre aspecte que hem après a utilitzar són els timers, ja que fins ara no n'havíem fet servir i ens feia falta per incrementar les monedes de cada jugador i per tal de tenir en compte el moviment de les tropes.
Finalment, creiem que s'aprèn molt més d'aquesta manera, més que no pas fent un examen com el del primer semestre. Això és perquè de cara al món laboral és absurd pensar en fer un projecte sense apunts/buscar informació per internet. Evidentment s'han de tenir uns coneixements, però realitzant projectes així s'adquireixen molt millor que no pas estudiant tota la teoria i havent de fer un examen en només una hora o hora i mitja i sense poder buscar informació.

7. Bibliografia

1. Manual BBDD en Java. La Salle URL. Disponible a la plataforma eStudy: Disseny i Programació Orientats a Objectes.

2. StackOverflow[en línea] [fecha de consulta: 20 de junio de 2020] Disponible en: https://stackoverflow.com/questions/12908412/print-hello-world-every-x-seconds

3. GitHub[en línea] [fecha de consulta: 22 de junio de 2020] Disponible en: https://gist.github.com/roooodcastro/6325153

4. StackOverflow[en línea] [fecha de consulta: 16 de junio de 2020] Disponible en: https://stackoverflow.com/questions/5921175/how-to-set-jpanels-width-and-height

5. Baeldung[en línea] [fecha de consulta: 24 de junio de 2020] Disponible en: https://www.baeldung.com/java-date-difference

6.Java [en línea] [fecha de consulta: 24 de junio de 2020] Disponible en: https://docs.oracle.com/javase/8/docs/api/java/time/Period.html#getDays--

7.Javatpoint [en línea] [fecha de consulta: 25 de junio de 2020] Disponible en: https://www.javatpoint.com/java-jtable

8.Oracle [en linea] [fecha de consulta: 26 de junio de 2020] Disponible en: https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html