Projecte

Minder

Grup B4

Aleix Ollé
Daniel Benet
Ferran Mateu
Víctor Valls
Ona Mañer

Disseny i programació Orientat a Objectes

CURS 2019-2020

TABLE OF CONTENTS

1. RESUM DE LES ESPECIFICACIONS DEL PROJECTE

Aquesta pràctica està dissenyada seguint el model client-servidor, per tant podem separar les especificacions en dues parts.

En la part del client el programa ofereix la possibilitat als usuaris de registrar-se introduint un nom d'usuari i correu electrònic per tal de poder-se identificar, a més de una contrasenya, la data de naixement i el tipus de compte (Premium o normal) que volen tenir. Un cop l'usuari ha introduït tota la informació el programa comprovarà que aquesta sigui correcta i, en aquest cas, enviarà la informació al servidor per tal de comparar-la amb la dels usuaris ja existents. Si tot és correcte l'usuari serà registrat i se li donarà accés al programa.

La primera vegada que l'usuari accedeix al programa ha d'omplir la informació del seu perfil que inclou una petita descripció, el seu llenguatge de programació preferit i una imatge que es mostrarà a la resta d'usuaris. Després d'omplir tota la informació el servidor la guardarà a la base de dades i el client haurà entrat al programa.

En la vista principal se li mostrarà al client altres usuaris que tenen gustos similars als seus i podrà decidir si aquests usuaris li agraden o no fent click al cor o a la creu respectivament. L'avantatge de ser Premium és que primer de tot apareixen els usuaris que ja t'han acceptat prèviament i un cop ja els hagi acceptat o descartat començaran a aparèixer la resta de manera aleatòria. En qualsevol moment un usuari pot editar el seu perfil en cas que vulgui canviar algun dels camps que el formen, la nova informació serà guardada a la base de dades i actualitzada a la resta d'usuaris.

Quan dos usuaris s'han fet like mútuament els hi apareixerà un missatge anunciant-los que han fet match i a partir d'aquell moment podran començar a xatejar.

Si un usuari es cansa de parlar amb un altre, pot trencar el match en qualsevol moment i així ja no rebrà més missatges d'aquell usuari.

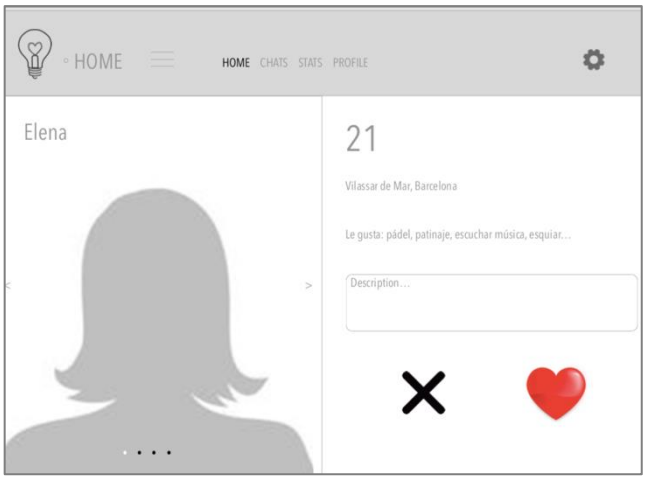
En cas que l'usuari vulgui sortir del programa té un botó de LogOut per tal de tancar la sessió i si vol tancar el programa ho pot fer tancant la finestra directament.

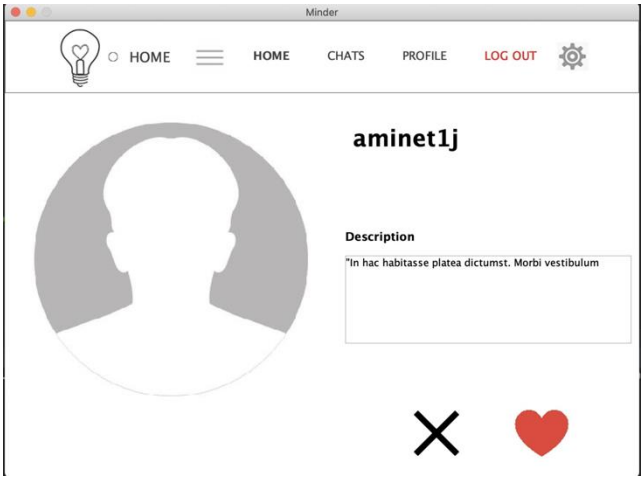
El servidor és l'encarregat de fer possible la comunicació entre els clients i la base de dades i entre els diferents clients. Quan un client actualitza alguna informació o bé dona like a un altre usuari el servidor és l'encarregat d'actualitzar la base de dades i notifica a l'altre client sobre el nou like. Cada usuari que es connecta obre un nou thread al servidor per tal de gestionar-lo de forma independent a la resta de clients que estiguin connectats en aquell moment.

Per acabar el servidor té dues vistes que permeten fer un seguiment del nombre de matches que s'han fet durant un període de temps concret i del top 5 d'usuaris amb més matches.

2. DISSENY INTERFÍCIE GRÀFICA

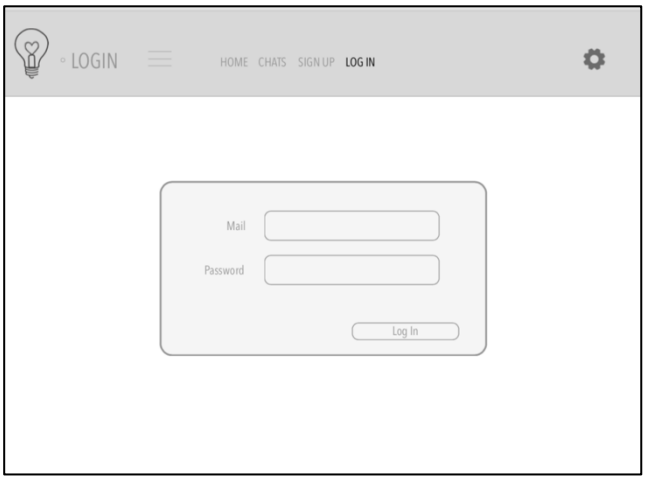
A continuació, us mosatrem un seguit d'imatges del que varen ser el snostres primers sketchos del que serien les interfícies gràfiques i al seu costat el resultat final de cada vista.

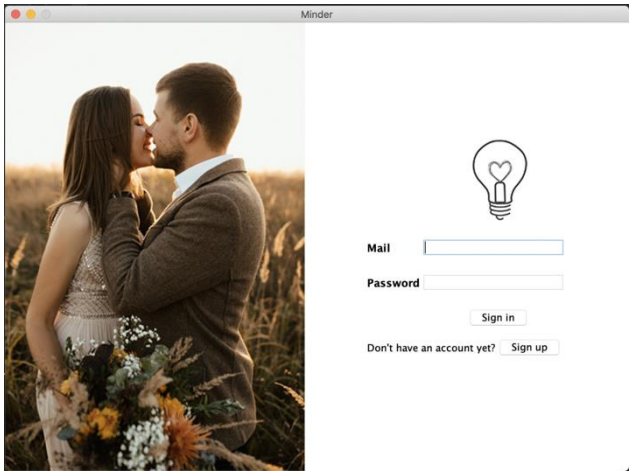




"HomeView": Un cop l'usuari estigui registrat apareixarà a la pantalla d'inici, on li apareixerà un usuari aleatori dels registrats a l'aplicació, i aniran passant a mesura que li doni like o dislike, dos JButtons. Altres components utilitzats han sigut JPanels, per estructurar la informació, JLabels, per posar la imatge, el username i el títol de la descripció de cada usuari ben col·locada i alineada, també hem fet servir JArea, per mostrar la descripció de cada usuari .

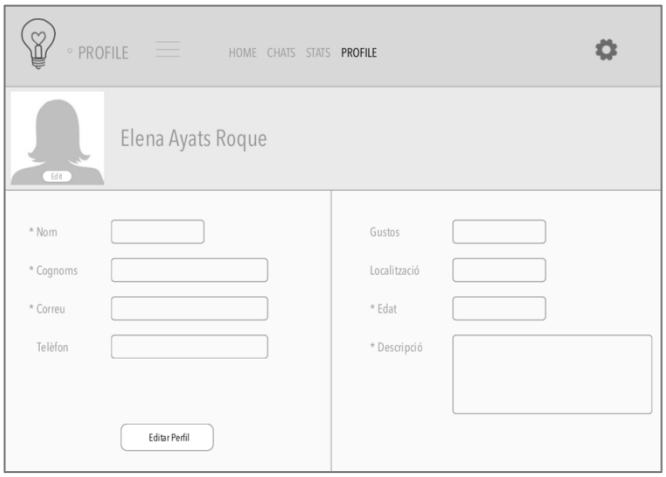
Per últim, tenim el "Header", la capçalera, que és una classe a part, ja que moltes vistes la compartiran i faran ús d'ella. Aquesta capçalera conté cada apartat de l'apicació, JButtons, on depenent de on es vulgui dirigir l'usuari, el portaran a una vista o a una altra.

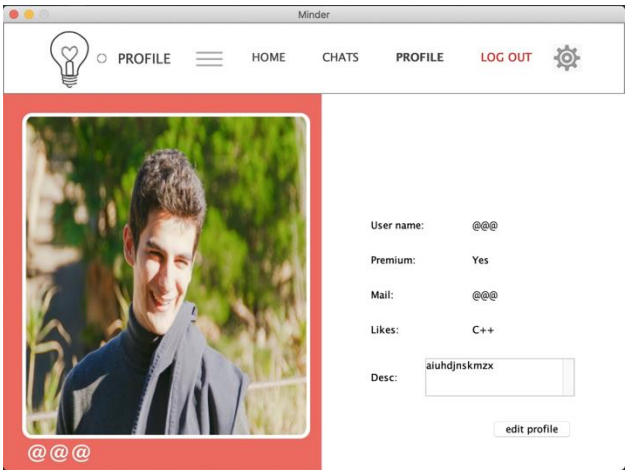




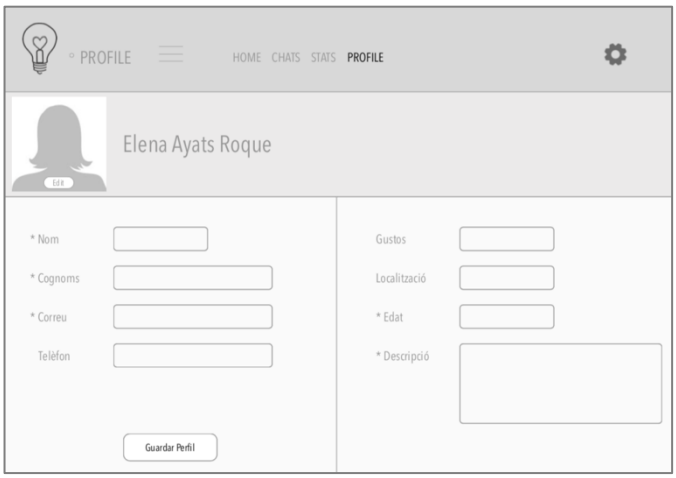
"LoginView": La pantalla del Login és la primera en sortir quan obres el programa, ja que si no estas registrat o no has entrat no pots accedir a lla "Home", on es veuen els perfils del usuaris registrats.

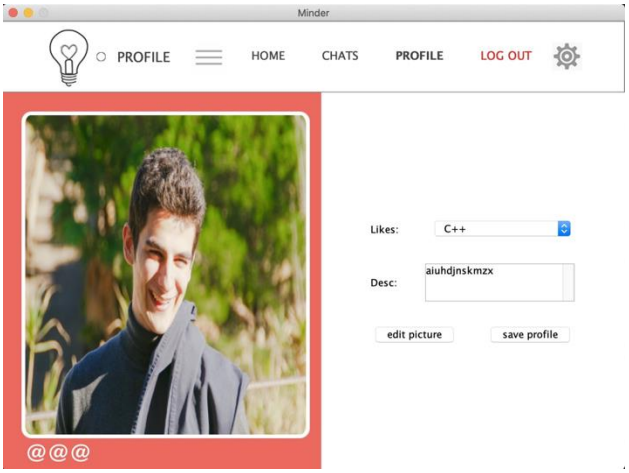
Com podem veure hem fet ús de JLabel, BufferedImages, JPanel, JTextField i JPasswordField, on introduirà les credencials l'usuari, i JButtons per poder accedir al LogIn o al Sign Up si encara no s'ha registrat.



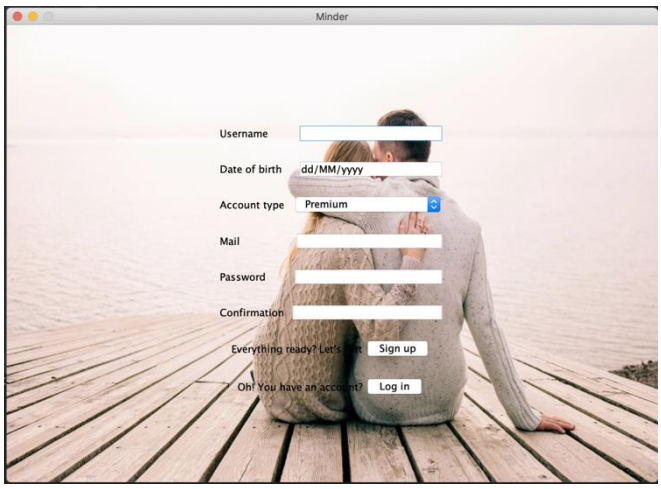


"ProfileView": El més destacable d'aquesta vista és que fem servir la classe Drawing, juntament amb la "ChatView", per mostrar el perfil i que sigui més fàcil dibuixar imatges i alinear els apartats desitjats. En quant als components , els més bàsics que fem servir són els JPanel, els JButtons, per editar el perfil i per guardar el perfil, el TextField per guardar la descripció, i un JcomboBox per escollir el tipus de llenguatje.





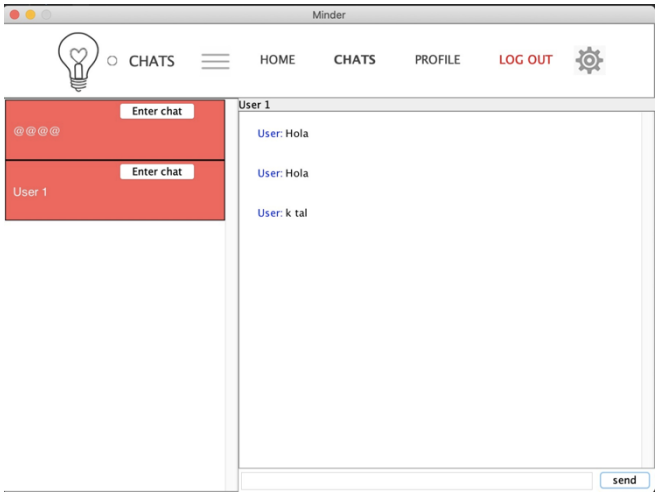
"Profile View": Aquesta vista és una modificació del ProfileView, ja que quan l'usuari decideix editar el perfil, quan fa ús del JButton, aquest podrà modificar les seves dades i guardar-les amb el JButton de saveProfile.



"Register View": Un cop l'usuari decideix registar-se arriba a aquesta pantalla, on haurà de posar les seves dades, el nom d'usari, fet amb JTextField, la data de naixement, que l'hem fet amb tipus Date, el tipo de compte, fet amb JComboBox, el mail, i el Password amb la seva corresponent comfirmació fet amb JTextField i JPassowordField. També hi haurà dos JButtons, el de sign up i el de log in per si ja s'havia registrat.

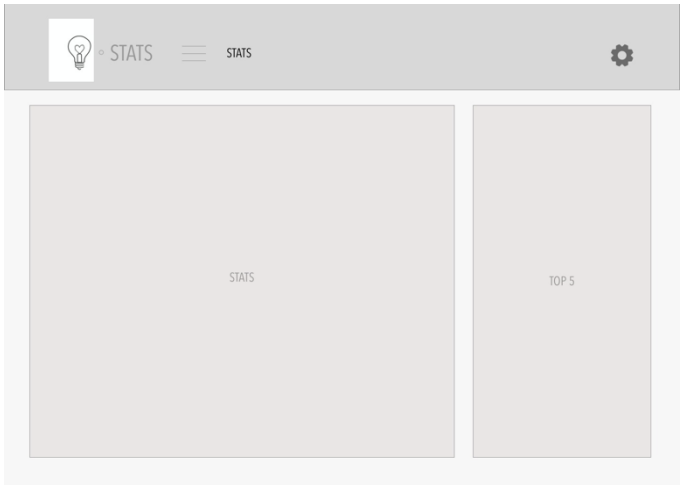
Una dada interessant d'aquesta vista és que els missatges són tots del propi JPanel, que és una classe a a part, "messagePanel", aon aquí es fa servir una funció, paintComponent perquè sigui més senzill alinear els missatges.

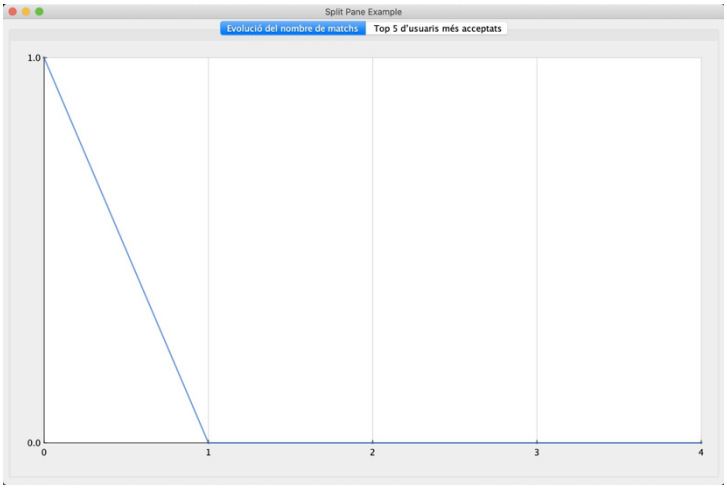
En quant els components més bàsics que fem servir són els JPanel, el JButton, per enviar els missatges, el JLabel i el JTextField.



Una dada interessant d'aquesta vista és que els missatges són tots del propi JPanel, que és una classe a part, "messagePanel", es aquí es fa servir una funció, paintComponent perquè sigui més senzill alinear els missatges.

En quant els components més bàsics que fem servir són els JPanel, el JButton, per enviar els missatges, el JLabel i el JTextField.



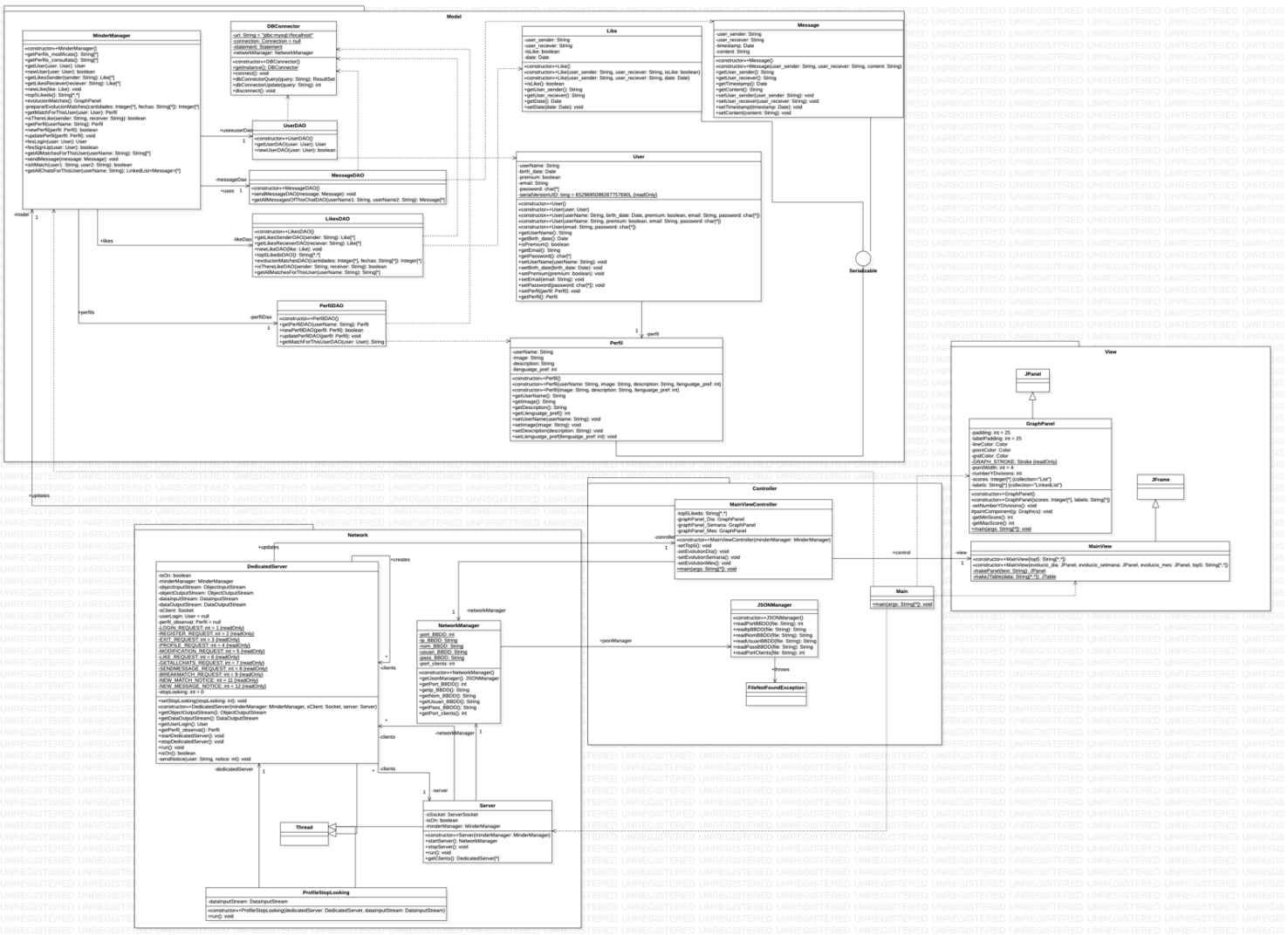


"MainView": Per últim tenim la vista, situada al Server a diferència de les altres, ja que l'usuari no podrà visualitzar-la. Aquesta mostrarà un gràfic amb l'evolució dels matches i també mostrarà el top 5 dels usuaris més acceptats.
Els components més utilitzats han sigut el JPanel, el JLabel, el JTable, per fer la taula, i el GraphPanel que és una classe a part per dibuixar els components dins d'un mateix JPanel.

3. MODEL DE DISSENY

3. 1. DIAGRAMA DE CLASSES

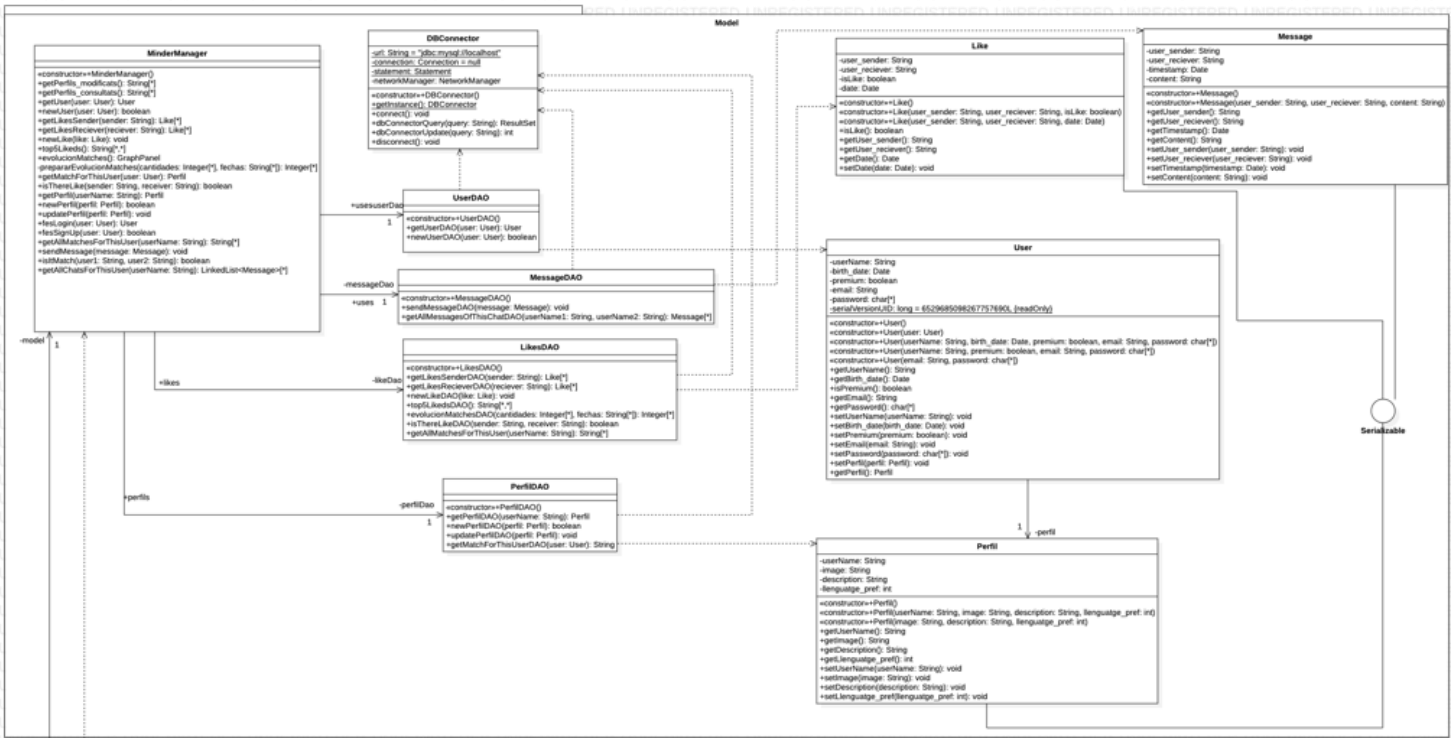
Server UML:



Aquest és un esquema general del Server, dividt per ordre en: Model, Network, Controller i View.

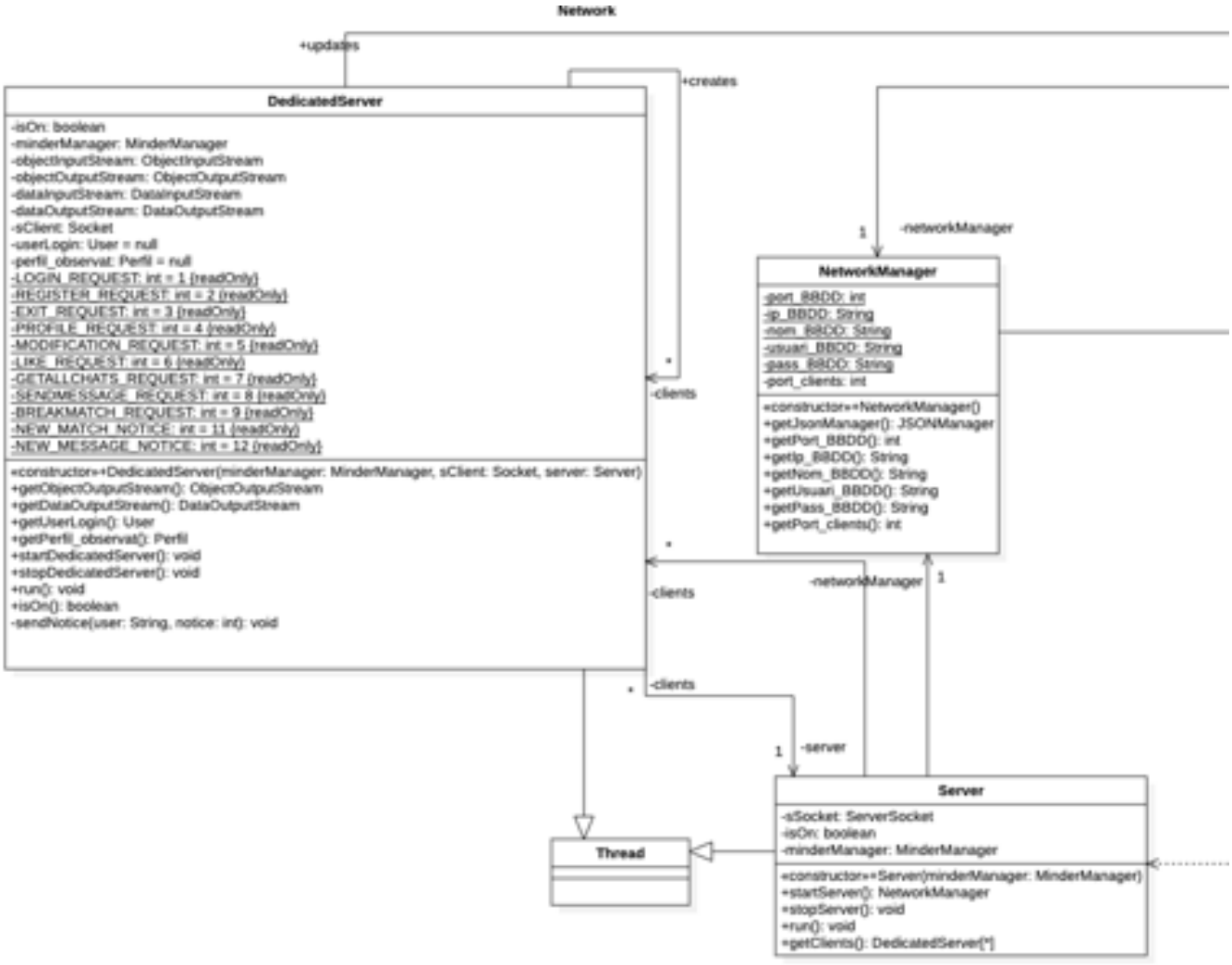
Nota: En el MinderManager hi ha dependències amb User i Perfil.

Model:



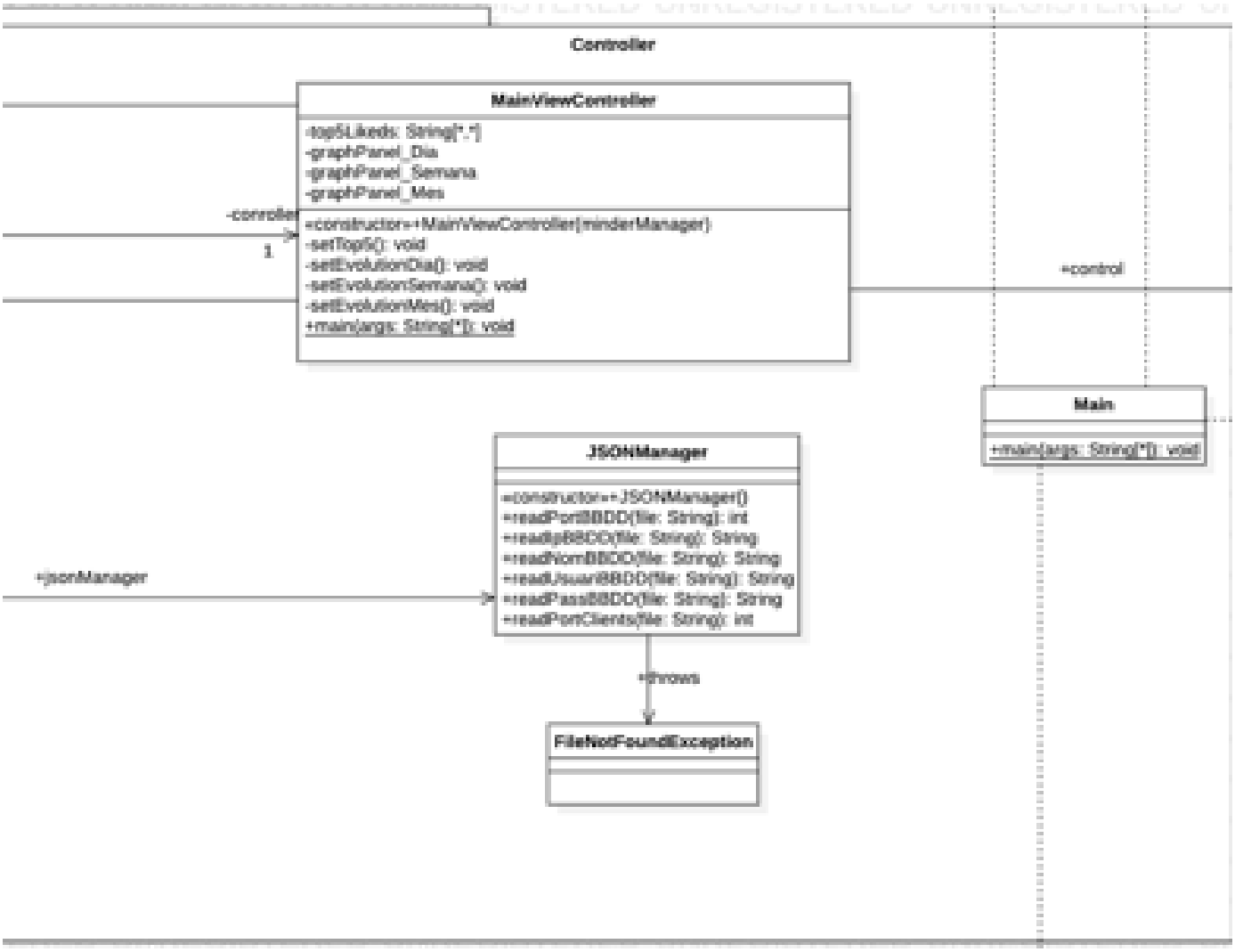
En el Model del Server podem observar les classes MinderManager, DBConector, LikesDAO, MessagesDAO, PerfilDAO i UserDao. A més hi trobem les respectives classes del Shared que són: User ,Like, Message i Perfil.

Network:



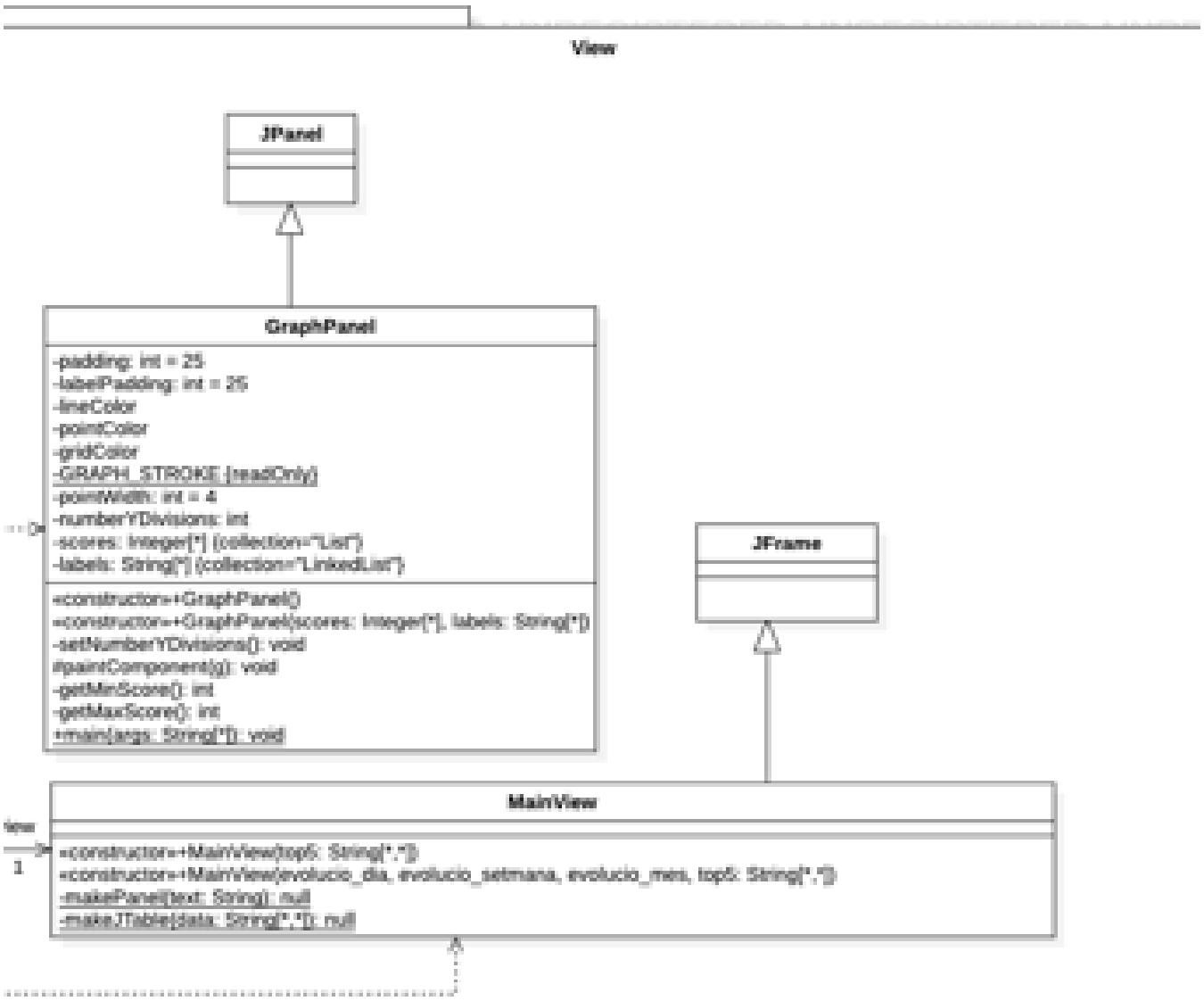
En el Network hi podem trobar les classes DedicatedServer, NetworkManager i Server.

Controller:



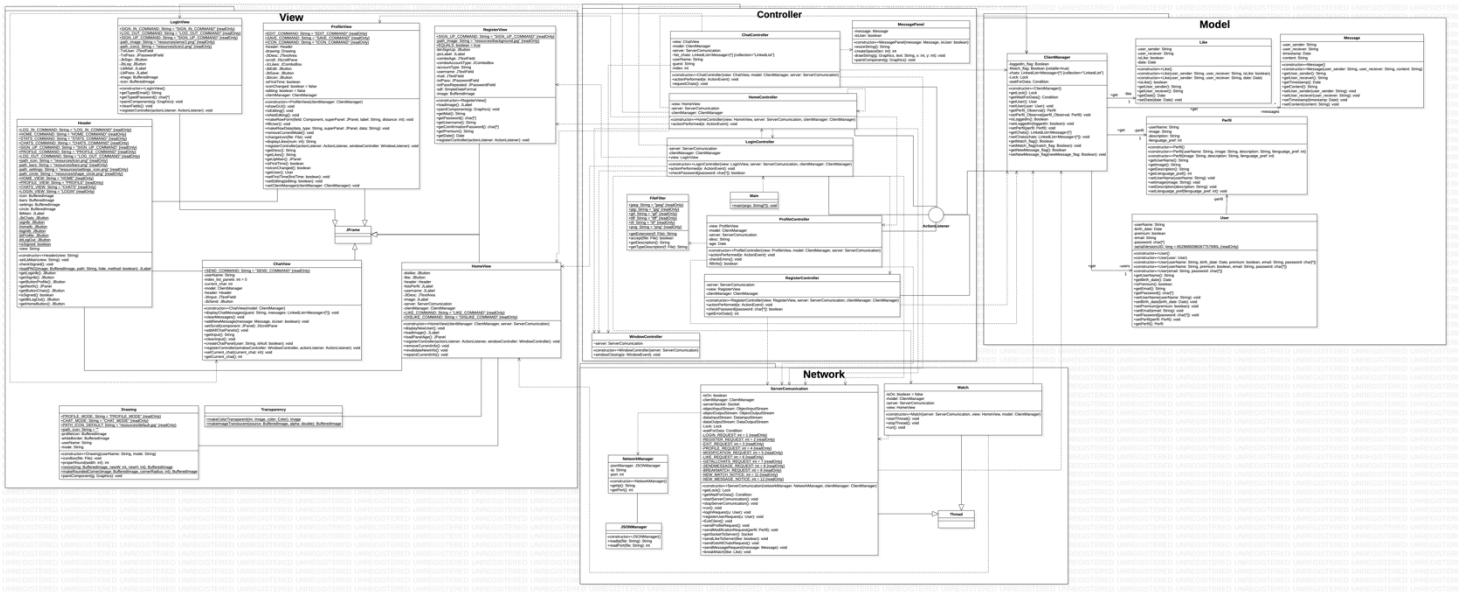
En el Controller hi podem trobar aquelles classes que es relacionaran amb View i Model i Network, que són: MainViewController, el JSONManager i el Main que l'hem col·locat dins del controller.

View:



Per últim, dins del Server hi podem trobar la View, on apareixen les classes de MainView (que ve del JFrame), on podrem visualitzar l'interfície gràfica del Server i el GraphPanel, un JPanel associat a la MainView.

Client UML:

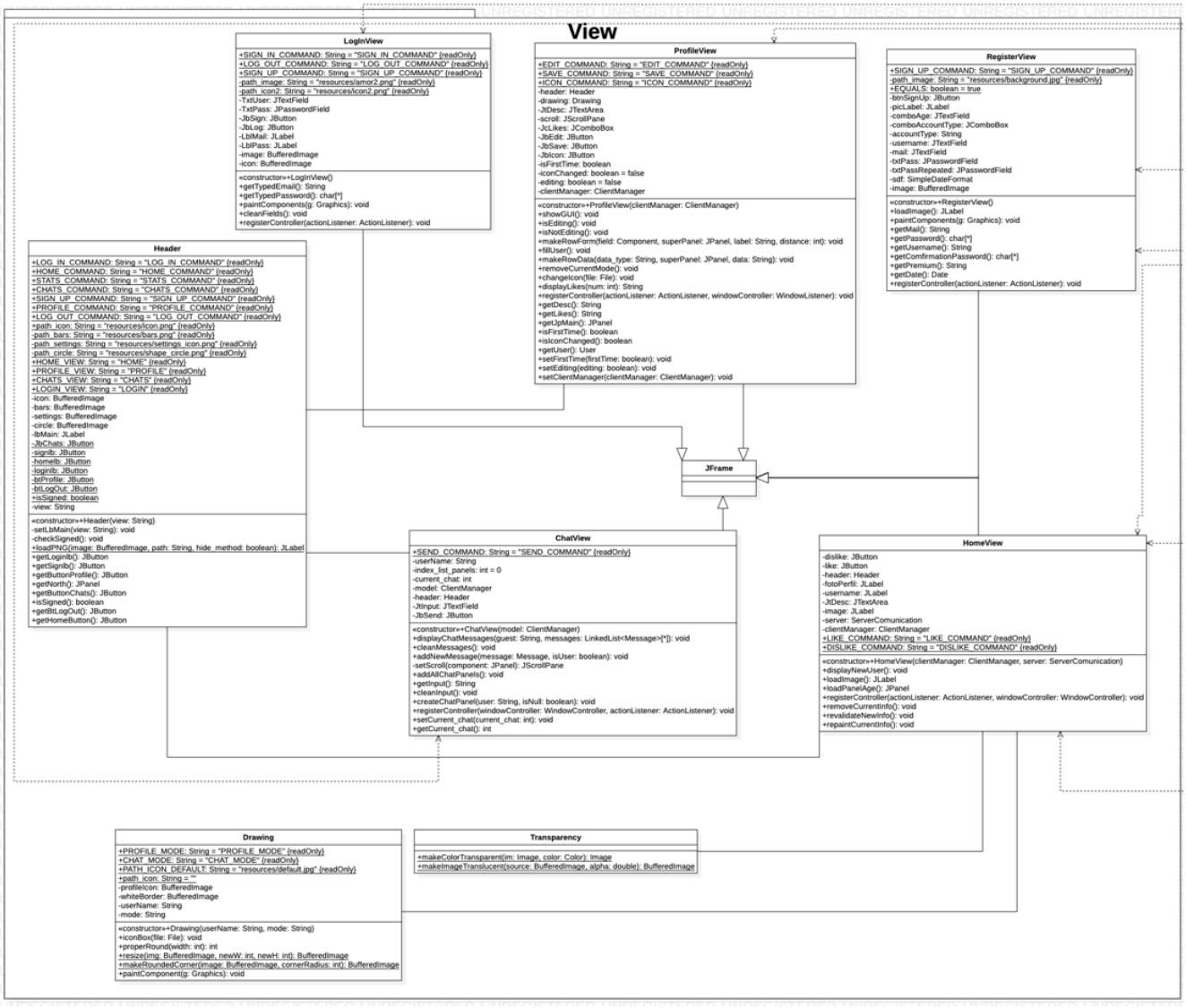


Aquest és l'esquema general, dividit per ordre en: View, Controller, Network i Model.

A continuació us mostrarem els packages per separat:

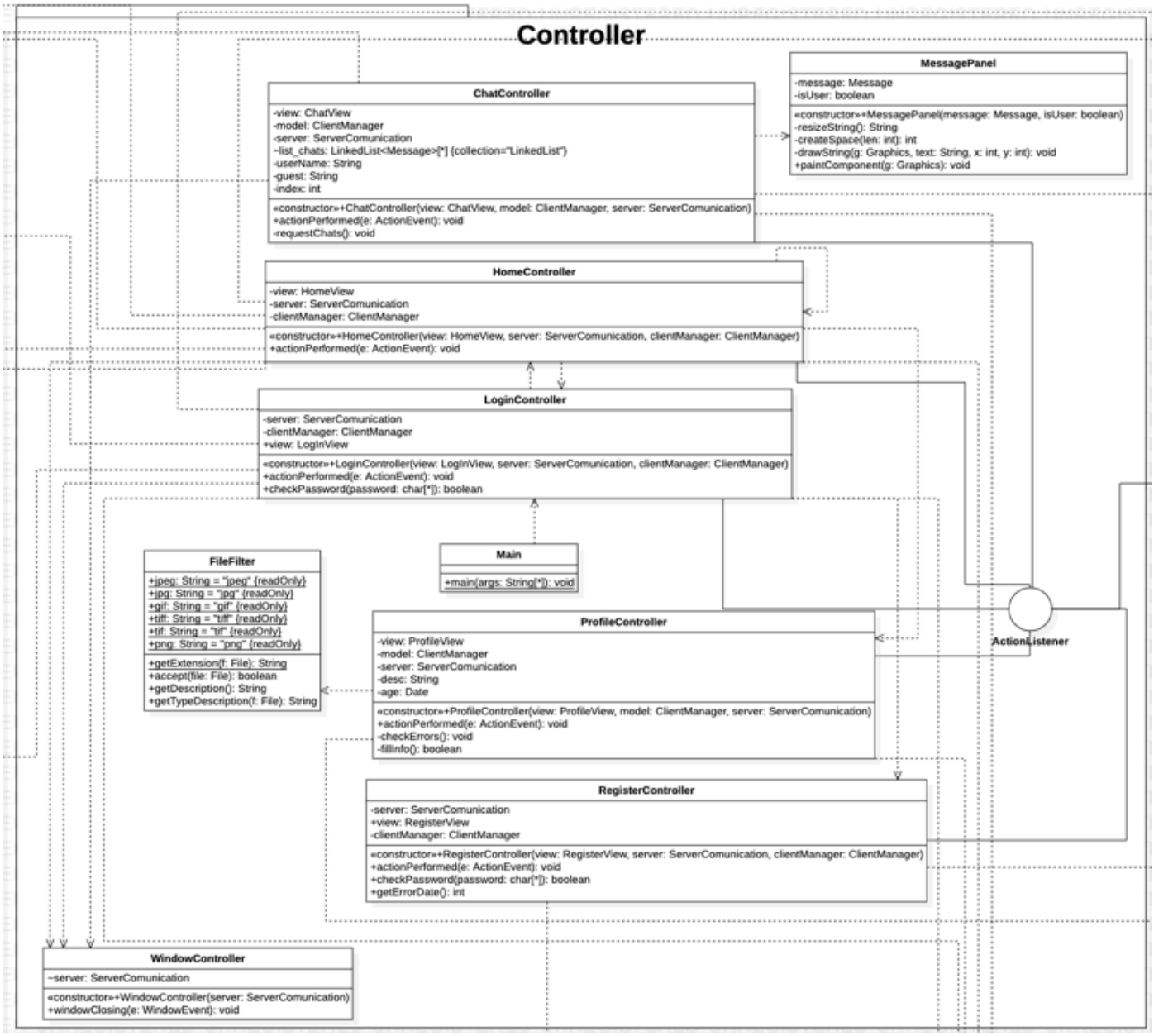
Notes: De cada controller surt una assossiació a la seva respectiva vista, a més, cada vista està relacionada amb les altres.

View:



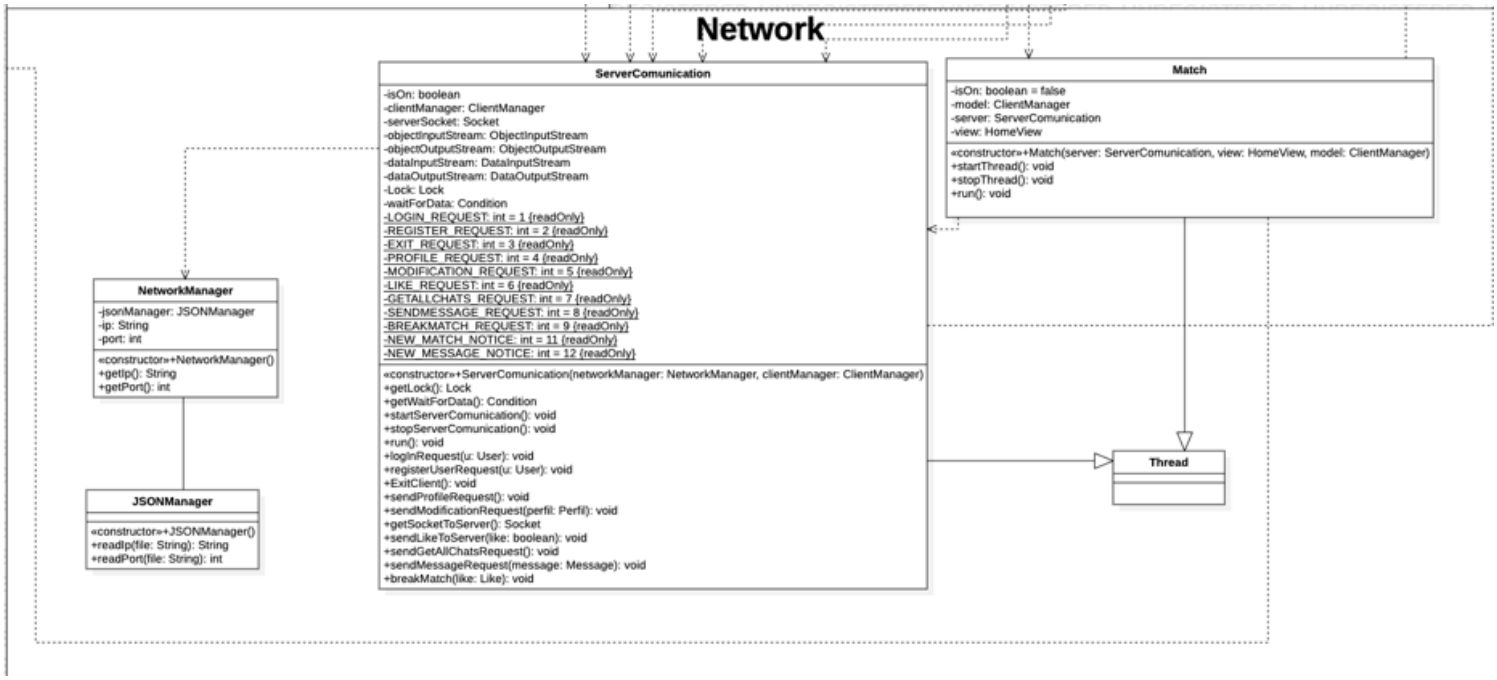
La View del Client és on es mostraran totes les interfícies gràfiques corresponents a l'aplicació que l'usuari podrà visualitzar i navegar en aquesta mitajançant diferents pestanyes. Hi podem trobar les classes de LoginView, ProfileView, HomeView, RegisterView, ChatView, que aquestes ens mostren els diferents apartats, i després tindrem les classes de Drawing, Header i Transparency, que formen part de les vistes per facilitar l'estructuració d'aquestes.

Controller:



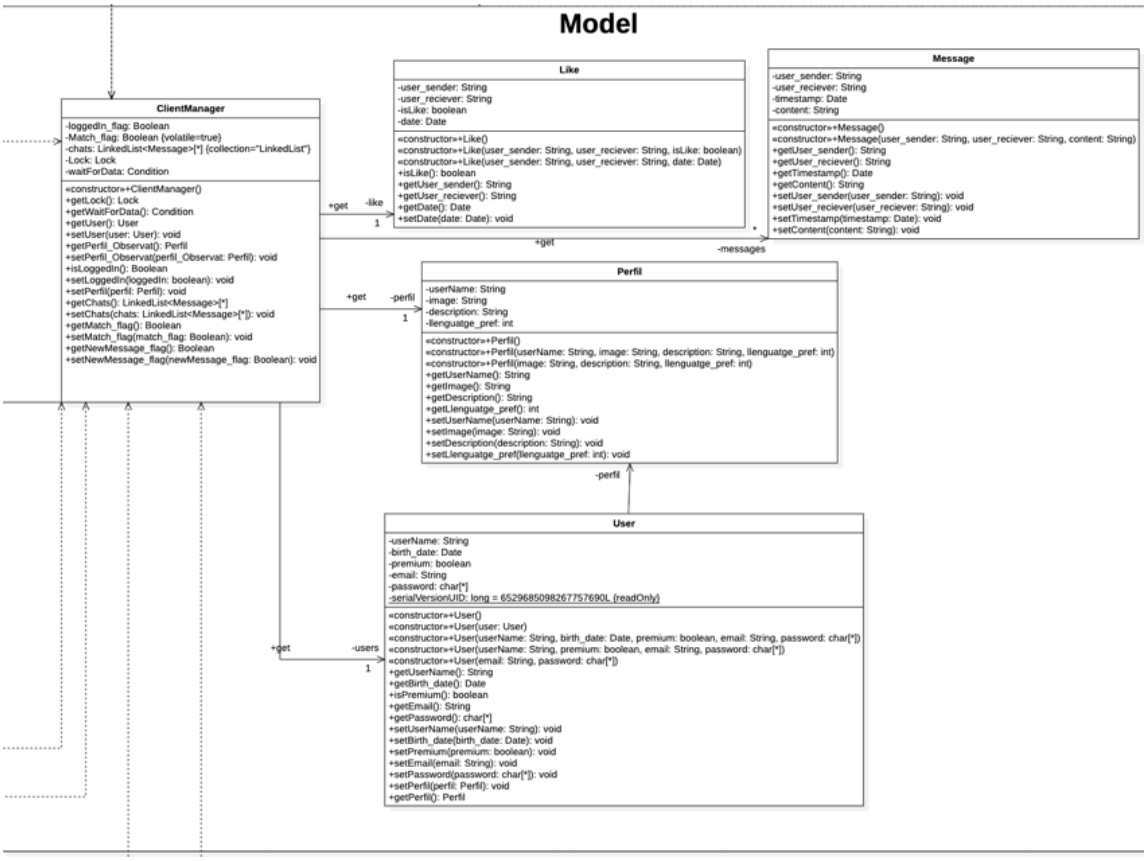
Aquí tenim el Controller, el que serà l'encarregat de comunicar-se amb la View, la Model i la Network, on podem trobar-hi les classes CharController, HomeController, LoginController, ProfileController, RegisterController, les encarregades de controlar les respectives vistes, i MessagePanel, WindowController , FileFilter, i el Main general.

Network:



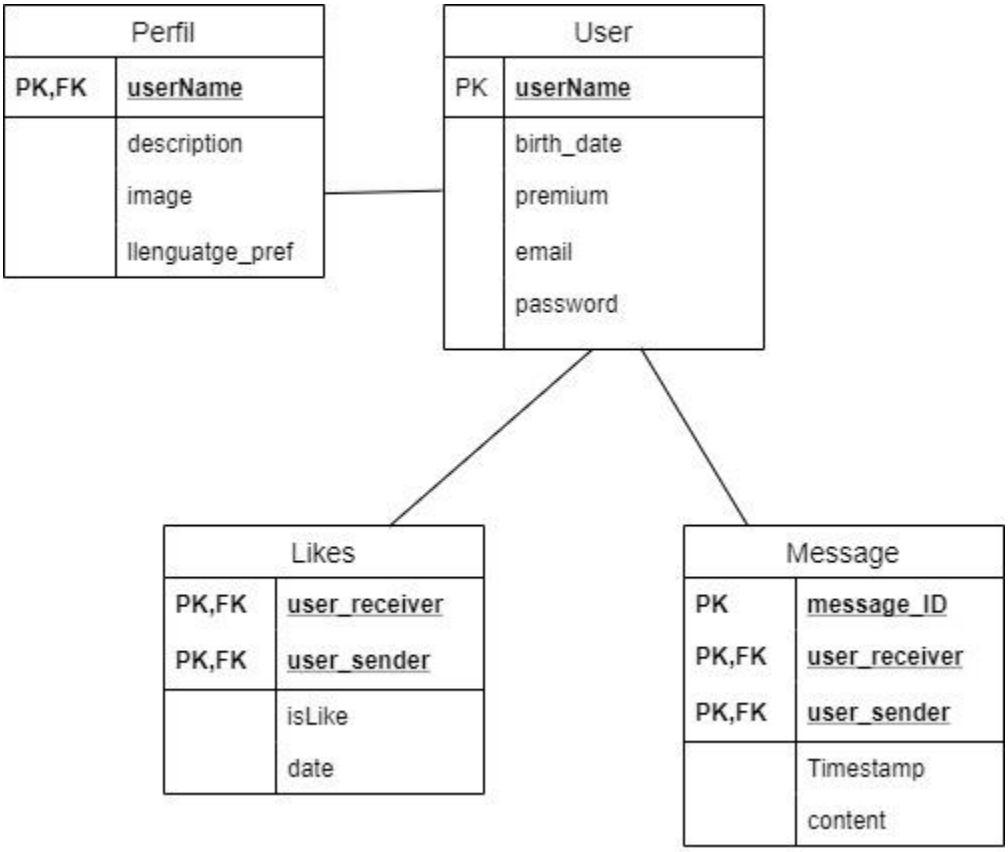
A continuació tenim la Network, on aquesta està formada per les classes ServerComunication, que s'encarrega de comunicar-se amb el Server mitjançant el Controller, la NetworkManager, el JSONManager i el Match, que ens dirà si dos usuaris han fet Match.

Model:



Per acabar, ens situem a la Model, que està formada per les classes ClientManager, Like, Message, Perfil i User.

BBDD MODEL RELACIONAL:



No hem inclòs les taules d'importació ja que aquestes només son per carregar dades de testeig.

3.2. PATRÓ DE DISSENY MODEL VISTA CONTROLADOR.

Com es demanava en l'enunciat hem seguit el patró de disseny Model-Vista-Controlador, en aquest model tenim el projecte dividit en tres parts. El model és la part del projecte que guarda les dades i la informació necessària pel funcionament del projecte.
La vista presenta la informació del model en un format fàcil d'entendre i d'interactuar per l'usuari final del nostre projecte així que necessita la informació proveïda pel model per tal de duu a terme la seva tasca correctament.
Per acabar, el controlador és el "cervell" del projecte, és la part que escolta a les vistes i fa peticions al model quan és necessari. És la part que uneix model i vista en la demanda i lliurament de la informació necessària per al funcionament del projecte.
En el nostre cas, com també seguim el model client-servidor totes dues parts consten d'un mòdul Network que s'encarrega de la comunicació entre els clients i el servidor i que, pel seu funcionament, podríem incloure com a part del controlador del projecte.

3. 3. DESCRIPCIÓ DE LES CLASSES MÉS IMPORTANTS.

3.3.1. MODEL SHARED

En primer lloc comentarem les classes a les quals tenen accés ambdós programes. Això és degut a que seran objectes “serialitzables” és a dir, que es podran enviar del client a servidor i viceversa. Al poder ser enviades i rebudes les dues han de ser iguals en ambdós programes.
User: User és la classe on s’emmagatzemen les dades privades dels usuaris. Mai s’enviarà un objecte tipus usuari a un client que no en sigui propietari ja que aquest conté informació com per exemple la contrasenya. A demés ofereix diversos constructors per adaptar-se a tots els casos en que aquest pugui ser utilitzat.
Perfil: Perfil és la classe que conté la informació pública de cada usuari. Aquesta classe inclou la imatge de l’usuari, el seu userName, el llenguatge preferit i una petita descripció sobre l’usuari. És la classe que fem servir per les relacions entre dos usuaris, com el nom d’usuari és únic no poden haver confusions amb altres usuaris i ja que la informació és pública no es posa en perill la privacitat dels usuaris en cas que aquesta informació es mostri per error.
Message: Message és la classe on s’emmagatzemen els missatges que els usuaris s’envien entre sí. Aquesta classe conté els userName de l’usuari remitent i del receptor del missatge i la data de l’enviament del mateix, per descomptat també guarda el text del missatge.

Like: Like és la classe encarregada de guardar la informació sobre els likes entre dos usuaris. Aquesta classe guarda el userName de l'usuari que està fent ús del programa i el userName de l'usuari que està observant, també conté un booleà anomenat isLike, aquest booleà valdrà true si l'usuari que fa ús del programa dona like a l'usuari observat o false si l'usuari dona dislike. Per acabar conté la data quan s'ha fet el like per tal de poder-la utilitzar en les estadístiques del servidor. Per tal de generar els matchs fem ús d'aquesta classe ja que si a la base de dades trobem dues files on el els usuaris són els mateixos i les dues files tenen el booleà isLike a true voldrà dir que aquells dos usuaris han fet match i se'ls ha de permetre xatejar entre ells. Quan un dels dos usuaris trenca el match actualitzem la fila on ell és l'usuari que ha observat a l'altre usuari i posem isLike a false, d'aquesta forma trenquem el match.

3.3.2. SERVIDOR

Server: Tot i ser una classe senzilla i amb relativament poc codi és el pal de paller del servidor. En aquesta classe s'obre un thread que espera la comunicació amb algun client, i és precisament perquè l'espera d'aquesta comunicació és bloquejant que cal fer un thread separat. Tan bon punt aparegui un client, s'iniciarà un servidor dedicat a aquest, se l'afegirà a la llista de clients connectats i el programa tornarà a mantenir-se a l'espera que aparegui un nou usuari.

DedicatedServer: Cada client tindrà accés a un servidor dedicat a ell, cada servidor serà un thread del programa principal. Un cop activat un servidor dedicat, aquest esperarà un Integer del client, en funció d'aquest integer sabrem quina és l'operació que demana el client. Precisament, ja que cada servidor dedicat està esperant un output del client, i que aquestes crides siguin bloquejants és la raó per la qual cada servidor dedicat consistirà d'un nou thread dins del programa del servidor. Destaca també que cada DedicatedServer te accés a tots els altres servidors per poder enviar notificacions a altres usuaris com per exemple quan s'ha rebut un match o un missatge nou.

MinderManager: Aquesta es la classe principal del model del servidor. Aquí tot i que no s'hi guardarà gaire informació, residirà la connexió entre el segment d'JDBC i la resta del programa. No es podrà cridar a cap de les operacions DAO des de cap punt del programa que no sigui aquest, per aquesta raó apareixerà una gran quantitat de funcions aparentment redundants que consistiran únicament en cridar la funció del mateix nom del connector amb la base de dades i retornar-la sense modificacions.

DBConnector: Pel que fa a la base de dades, encara que cada classe del model te la seva versió "DAO" per tal de separar adequadament model i networking la classe que des d'on realment es produeix la connexió es aquesta. Aquí es on estan els procediments que estableixen la connexió inicial amb la base de dades i cada cop que es vol executar una query a la base de dades, això es fa a través d'algun procediment d'aquesta classe.

3.3.3. CLIENT

ServerComunication: El client haurà d'estar escoltant constantment al servidor per tal de saber si ha aparegut un nou match o missatge en temps real. Una vegada més per esquivar la problemàtica de les escoltes bloquiejants, aquesta classe generara un nou fil d'execució. Aquest fil tindra una estructura similar a la del dedicated server, romandrà a l'escolta d'un valor Integer i en funció d'aquest s'executarà la opció corresponent

ClientManager: Aquesta és la clase principl del model del client. Es aquí on es guardarà información rellevant sobre, per exemple quin usuari está conectat al model, l'usuari que está observant i els chats. Adicionalment hi apareixen algunes variables interesants, el programa compta amb uns flags que hauran de ser llegits per diferents threads, per asegurarnos que aquests els llegeixin sense falta i no assumeixin el seu valor, aquestes variables incorporen l'etiqueta volatile. D'altre banda també es contenen dos objectes tipus Lock i Condition. Ja que el nostre programa treballa amb threads i aquests no es comporten sempre de la mateixa manera, utilitzem la clase lock per sincronitzarlos i Condition per fer que s'esperin entre ells. Tot i que sense aquestes clases els threads podrien funcionar, en alguna circunstancia especial podrien no ferho, donant lloc a punters buits ja que aquests han estat accedits abans de poderhi carregar l'informació

Entrant ara a la part visual del Client, Gran part de les classes son autoexplicatives i no desenvolupen cap tasca fora del normal, hi ha dues però que tenen unes característiques especials que si ens agradaria comentar.
Drawing: Aquesta es una classe que exten Jpanel i que s'utilitza per poder fer servir PaintComponents". D'aquesta manera podem fer que elements de les nostres vistes com per exemple les imatges es printin per pantalla de forma més estètica a la vegada que ens facilita la tasca de dibuixar.

Header: Com es pot observar a les vistes del programa, pràcticament totes en les que l'usuari esta conectat apareix el mateix menú a la part superior de la pantalla. Hem fet ús d'aquesta classe per tal de generalitzar aquest menú i no necessitar crear-lo de nou en cada vista, sinó simplement fer un get de Header.

4. METODOLOGIA DE DESENVOLUPAMENT

A continuació farem un anàlisi del procés que hem seguit per desenvolupar el projecte, esmentant els objectius inicials i si han estat realistes i satisfactoris. Addicionalment explicarem com s'ha repartit la feina entre eles membres del grup, i quines eines s'han utilitzat per fer-ho.

En la primera etapa del procés de desenvolupament, ja teníem clara una cosa: Era imprescindible poder comprovar que el codi funcionava en tot moment. Això implicava focalitzar els esforços inicials en tenir uns bons protocols de comunicació Client-Servidor i una base de dades plena de dades utilitzables. Donades aquestes consideracions fins i tot abans de realitzar el primer disseny tots els membres del grup vam fer recerca per tal d'esbrinar com solucionar aquests elements abans de començar tan sols a dibuixar les vistes. Aquesta investigació va incloure des de aprendre d'exercicis de classe com establir una correcta connexió, a dominar Swing per tenir clars quins contenidors caldria utilitzar a cada vista.

Un cop fets els primers dissenys, amb més certesa de que aquests serien útils gracies a la investigació prèvia, aconseguir un programa funcional amb connexió client servidor base de dades sobre el qual poder treballar individualment i poder comprovar cadascú el seu codi, tot i que no va ser fàcil no va ser la part més complicada del projecte. No cal dir, que tot i que el projecte final s'aproxima al disseny inicial ha calgut fer una gran quantitat de modificacions.

A partir d'aquesta etapa la resta del projecte va consistir en anar assolint petites fites, les quals no es donaven per vàlides fins a tenir feta una extensiva comprovació, possible gracies a haver prioritzat les estructures base del programa.

Val a dir, que com era d'esperar algun error s'escaparia per aquesta raó la fase final del projecte ha consistit en una profunda fase de "testeig ".

Entrant ara ja en el repartiment de la feina dins del grup, a grans trets vam dedicar dues persones a les vistes, una a la connexió client servidor una a la base de dades, i un comodí que ajudes en tots els aspectes. Òbviament ens vam trobar en moments en que, per exemple, un membre de la parella de les vistes no podia progressar o comprovar el seu codi sense que el servidor li fes arribar les dades necessàries. Per aquesta raó, i entrant en les eines que s'han utilitzat, la comunicació entre membres del equip va ser imprescindible.

Per estar adequadament comunicats l'eina principal que vam utilitzar va ser Discord. Aquesta eina ens ofereix un chat entre membres del grup, tant de veu com per text, juntament amb la capacitat de deixar missatges per que aquests siguin llegits en qualsevol moment per altres usuaris. Amb discord, compartir codi, solucionar problemes o ajudar a companys de grup es va fer més fàcil tot i la situació de confinament actual.

L'altre eina que cal destacar, es que vam ser un dels grups que es va decidir a fer us de Git. Tot i que en un primer moment va representar un gran mal de cap, Git ha acabat sent una eina d'una enorme utilitat amb la qual tots podíem treballar en el codi en qualsevol moment sense patir que aquest fos incompatible amb els dels companys. A demés ens ajudava a saber exactament que

és el que havia fet cada membre del grup, reduint així la necessitat de treballar en el mateix moment i agilitzant considerablement tot el procés de desenvolupament.
Per anar acabant, hem esmentat anteriorment que hem tingut una base de dades amb usuaris i dades vàlides per poder treballar i comprovar el correcte funcionament del nostre codi. Per obtenir aquesta base de dades hem fet us de Mockaroo. Plataforma web que ofereix crear fitxers CSV amb diferents dades, ideals per al “testeig” d’aplicacions com la nostra.

5. DEDICACIÓ

En aquest gràfic mostrem la quantitat d'hores aproximades que cada integrant del grup ha dedicat a la pràctica. Hem fet una divisió segons el tipus de feina que s'ha dut a terme en les diferents etapes del desenvolupament de la pràctica. Tot i que les proporcions poden variar segons cada integrant del grup, tots hem dedicat aproximadament el mateix nombre total d'hores a la pràctica. Afegim també hores que considerem d'aquesta pràctica com les hores d'autoaprenentatge, ja que molts conceptes que hem hagut d'utilitzar no han estat explicats a classe. En l'apartat de codificació també s'hi inclou les hores de correcció de possibles errors del codi, ho adjuntem d'aquesta manera perquè des d'un principi hem intentat anar corregint totes les noves funcionalitats que incloíem al codi i així no haver de dedicar temps extra a la correcció un cop la pràctica estigués finalitzada.

6. CONCLUSIONS

Per anar acabant és adient fer un petit repàs sobre què ens ha aportat tant en l'àmbit educatiu com a personal la realització d'aquest projecte.

En primer lloc ens agradaria fer una petita reflexió sobre la càrrega de treball. A les acaballes de curs, en retrospectiva, veiem com tot i que aquesta assignatura només representa aproximadament un 10% dels crèdits totals als quals estem matriculats, ens ha representat entre un terç i la meitat de les hores de feina dedicades a la Universitat. Estem parlant d'una quantitat d'hores equivalent a projectes de final de grau d'altres universitats. Això ens provoca sentiments creuats, per un costat haver sigut capaços de realitzar aquesta tasca representa una de les fites més importants de la nostra etapa d'estudiants. Contrasten però, aquestes últimes setmanes, en què ha prioritzat un sentiment d'alliberació sobre el d'autosatisfacció per haver fet progrés o après noves eines.

D'altre costat, al llarg del desenvolupament de la pràctica ha tornat a aparèixer el debat sobre fins a quin punt és beneficiós per l'alumnat el fet d'haver de descobrir de forma autodidàctica una quantitat tan gran d'eines necessàries per a la realització del projecte. Ens agradaria recordar que som una generació que no només ha crescut ja dins de l'era digital, sinó que la nostra educació ha estat adaptada a aquesta. Som la generació de l'autoaprenentatge. Per aquesta raó quan ha calgut dedicar una gran quantitat d'hores a aprendre noves formes, i eines per programar, sortint-nos ja del nombre d'hores que caldria dedicar basat en el nombre de crèdits de l'assignatura, no s'ha pogut evitar un creixent sentiment de frustració.

Per anar acabant, voldríem fer una reflexió sobre l'estructuració dels treballs al voltant de Jira i les altres funcionalitats d'altassian. L'estructuració en esprints de l'assignatura ha estat d'un benefici extraordinari per al nostre grup. Ens ha ajudat a tenir clar en tot moment que calia fer per progressar i a tenir una clara referència de la feina que han seguit els companys en tot moment. Gràcies a Jira, hem tingut objectius assequibles al llarg de tot el projecte que han ajudat a tenir resultats tangibles inclús en les primeres etapes. Això ha ajudat també a mantenir una gran motivació per part de tots els membres del grup en el projecte. Ha estat una llàstima que aquestes últimes setmanes, per culpa del tracte a distància amb els professors, no s'hagi pogut mantenir el format dels sprints i l'organització no hagi estat tan bona.

Finalment, els cinc membres del grup podem afirmar unànimement que la recepta de l'èxit, i la raó per la qual hem pogut entregar de forma ordinària, ha estat el vincle i bona relació que hi ha hagut entre tots els membres del grup. El bon ambient ha estat clau per assegurar que cap dels membres del grup és sentis apartat i disminuís el seu ritme de treball, a més tot i que des d'un primer moment hi havia membres amb diferents nivells i experiències a l'hora de programar, ningú ha tingut problemes en dedicar hores a ajudar o ensenyar des de zero qualsevol dels conceptes a un company.

7. BIBLIOGRAFIA

Mkyong [en línia]. [data de consulta: 24 maig 2020]. Disponible a: https://mkyong.com/java/how-to-parse-json-with-gson/

Oracle [en línia]. [data de consulta: 24 maig 2020]. Disponible a: https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/locks/Condition.html

Mockaroo, LLC. [en línia] [data de consulta: 24 maig 2020]. Disponible a: https://mockaroo.com/

Java2s [en línia] [data de consulta: 24 maig 2020]. Disponible a: http://www.java2s.com/Tutorials/Java/Swing_How_to/JOptionPane/Position_JOptionPane_showMessageDialog_at_the_center_of_screen_.htm

StackOverFlow [en línia] [data de consulta: 24 maig 2020]. Disponible a: https://stackoverflow.com/

Universitat d'Alacant. *La norma ISO 690:2010(E)* [en línia] [data de consulta: 24 maig 2020]. Disponible a: http://werken.ubiobio.cl/html/downloads/ISO_690/Guia_Breve_ISO690-2010.pdf

Jira [en línia] [data de consulta: 24 maig 2020]. Disponible a: https://atlassian.salle.url.edu/

BitBucket [en línia] [data de consulta: 24 maig 2020]. Disponible a: https://atlassian.salle.url.edu/