Disseny i programació orientats a objectes

Pràctica del segon semestre
Minder

Departament d'Enginyeria
La Salle – Universitat Ramon Llull
11 de Febrer de 2020

Índex

1. Descripció general

Actualment hi ha força aplicacions que serveixen per a conèixer, xatejar i quedar amb altres persones, però no en tenim cap que sigui exclusiva per a estudiants de La Salle. Així, des del departament de Desenvolupament i Recerca pels Projectes de DPOO, s'ha decidit implementar una alternativa, el Minder.

L'arquitectura del sistema Minder és client-servidor. Tal com es pot observar a la figura següent, en aquesta hi ha un programa servidor que s'executa en una màquina en concret (el servidor central) i un conjunt de màquines que executen el programa client (els usuaris de la pròpia aplicació).

El programa servidor, entre d'altres, emmagatzemarà les dades d'accés, les connexions entre els diferents usuaris i els missatges de xat. El programa client, serà el programari que utilitzaran els usuaris mitjançant el qual es podran connectar al sistema, accedir al seu compte i connectar i xatejar amb altres persones.

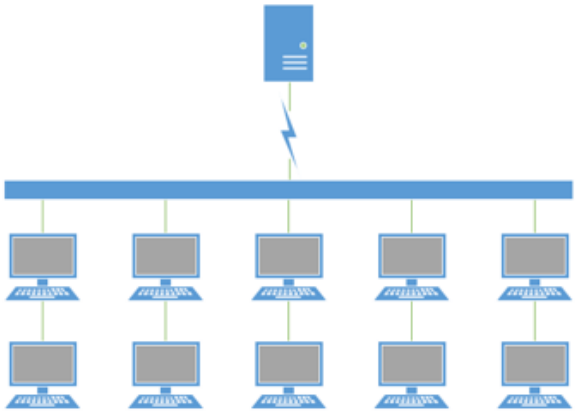

Figura 1. Representació de l'arquitectura client – servidor del Minder.

Els apartats següents descriuen els detalls i el comportament que s'espera de les funcionalitats concretes que ha d'implementar cadascuna de les parts identificades en aquest escenari: servidor i client.

2. Especificació de requeriments

Aquest apartat recull el funcionament i el comportament que s'espera de l'aplicació Minder per tal de satisfer les especificacions requerides. En primer lloc les del programa servidor i tot seguit les del client.

2.1. Servidor

El servidor ha de ser un programa que pugui ser executat en una màquina independent de la resta dels clients, que ofereixi una interfície gràfica amb diverses opcions i capaç de proveir unes funcionalitats mitjançant la recepció i enviament de dades per una xarxa. Concretament ha d'oferir funcionalitats per:

1. Registrar / autentificar usuaris (no accessible mitjançant la interfície, opció remota)

2. Gestionar els perfils dels usuaris i sincronitzar-los a temps real entre ells (no accessible mitjançant la interfície, opció remota)

3. Gestionar els *matchs* i els xats entre usuaris i sincronitzar-los a temps real (no accessible mitjançant la interfície, opció remota)

4. Mostrar l'evolució del nombre de matchs respecte el temps (des de la interfície gràfica)

5. Mostrar el top 5 d'usuaris més acceptats (des de la interfície gràfica)

Les tres primeres funcionalitats són d'accés remot o tenen a veure amb la comunicació remota amb els programes client en execució. Estan a la llista en el sentit que el programa servidor ha de tenir la capacitat d'atendre les peticions corresponents mitjançant una connexió TCP (amb *sockets*) des del programa client. En canvi, les funcionalitats 4 i 5, són opcions que ha d'oferir la interfície gràfica del propi programa servidor. Els detalls d'aquestes funcionalitats s'expliquen al llarg dels propers apartats.

Com es veurà a continuació, el programa servidor haurà d'emmagatzemar les dades dels usuaris (nom, edat, correu...) i informació referent al seu històric (matchs, persones acceptades...) en una base de dades MySQL. Referent a això, és important notar el fet de que el programa servidor és l'únic programa que té accés a la base de dades, i que la lògica de funcionament del sistema (com per exemple els xats en temps real) comprèn l'intercanvi d'informació entre els programes servidor i client mitjançant la comunicació TCP amb *sockets*. Aquestes característiques es poden apreciar en el següent diagrama de desplegament UML.

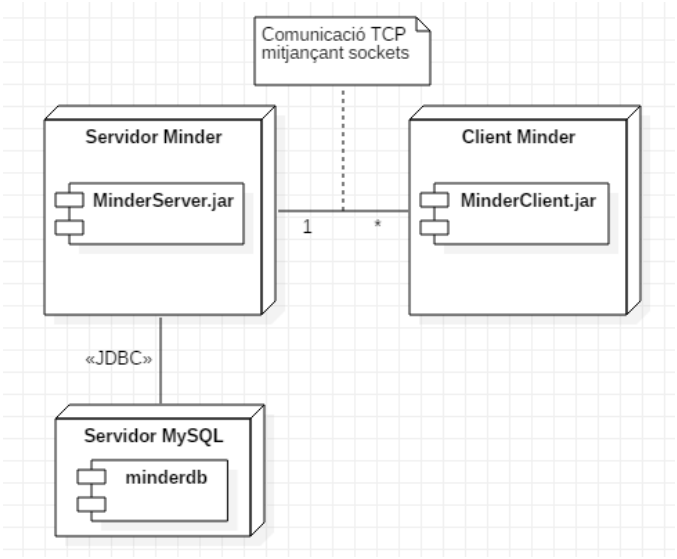

Figura 2. Diagrama de desplegament UML del Minder

2.1.1. Configuració

En el moment d'engegar el servidor aquest haurà de llegir un fitxer de configuració anomenat *config.json* ubicat en el directori arrel del projecte on hi haurà la següent informació:

- Port de connexió amb la Base de dades
- Direcció IP del servidor de la base de dades (normalment "localhost")
- Nom de la Base de dades
- Usuari d'accés a la Base de dades
- Contrasenya d'accés a la Base de dades
- Port de comunicació on escoltar les peticions dels programes Client

2.1.2. Registrar usuaris

La primera funcionalitat ha de permetre registrar nous usuaris al sistema. Els camps necessaris per a registrar un usuari seran explicats més endavant en el client. El registre es farà de forma remota, opció que ha d'estar activa en tot moment mentre el servidor és en execució. El servidor, s'encarregarà de rebre la informació enviada pels clients, mitjançant *sockets*, en la seva pròpia opció de registrar-se.

Opcionalment, es pot implementar la possibilitat de registrar usuaris també des de la interfície gràfica del servidor, complint els mateixos requeriments.

El servidor haurà de ser qui comprovi que les dades de registre són vàlides en aquelles situacions en què cal consultar la base de dades. És a dir, que l'usuari introdueixi un camp buit en registrar-se pot ser gestionat des del programa client, però que el nom d'usuari ja existeixi ha de ser verificat pel programa servidor, ja que és aquest qui té accés a la base de dades. Tanmateix, per motius de seguretat tot allò que comprovi el programa client també haurà de ser comprovat pel programa servidor.

A l'igual que pel registre, per a l'autentificació el servidor és l'encarregat de verificar que l'usuari/email existeix i que la contrasenya introduïda és l'associada a aquest usuari.

Quan el servidor rebi una petició de registre ha de procedir a efectuar-lo i donar resposta al programa client segons sigui el resultat, satisfactori o insatisfactori, de la petició. En cas d'error cal explicitar, sempre que sigui possible, què ha impedit el registre.

2.1.3. Gestionar els perfils

El programa ha de guardar les diferents dades dels perfils dels usuaris a la base de dades. En cas que un usuari estigui veient el perfil d'un altre i aquest sigui modificat, aquesta actualització s'haurà de dur a terme a temps real.

2.1.4. Gestionar les connexions i xats entre usuaris

El programa ha de gestionar i guardar quins usuaris han estat acceptats per altres, de manera que en cas que s'acceptin mútuament, es produirà l'anomenat match i tindran l'opció de xatejar entre ells.

A més, en cas que donat un *match*, dos usuaris comencin a xatejar, s'hauran de guardar els missatges de manera que, si un usuari es desconnecta i posteriorment torna a iniciar sessió, tingui accés a la conversa. Com es comenta més endavant, s'hauran de guardar, **com a mínim**, els últims 20 missatges de cada conversa entre dos usuaris.

2.1.5. Mostrar evolució del nombre de matchs

Aquesta funcionalitat ha de ser una opció accessible mitjançant el menú de la interfície gràfica del servidor. Aquesta ha de mostrar mitjançant un gràfic de línies, l'evolució del nombre de matchs respecte el temps. S'ha de permetre veure aquesta evolució durant l'últim dia, l'última setmana o l'últim mes, escalant la gràfica adequadament depenent del cas.

El gràfic de línies haurà de ser generat des del programa, utilitzant les eines de dibuix que ofereix la llibreria AWT/SWING de Java i escalat de forma conseqüent en funció dels valors mínim i màxim de l'eix de les y (l'eix d'ordenades) on es mostri el nombre de matchs que s'han donat. No es poden utilitzar llibreries externes.

El servidor ha de mantenir les dades actualitzades adequadament, tenint en compte que dos usuaris poden fer match però que posteriorment qualsevol dels dos usuaris pot desfer el match. Els històrics dels usuaris s'han d'emmagatzemar a la base de dades relacional MySQL.

2.1.6. Mostrar el top 5 d'usuaris més acceptats

De forma similar a la funcionalitat anterior, aquesta també ha de ser accessible mitjançant una opció del menú de la interfície del programa servidor.

En aquest cas, el programa ha de permetre veure una taula mostrant els 5 usuaris que més cops han estat acceptats, ordenats descendentment, mostrant també la quantitat de matchs de cada un.

Opcionalment, es poden mostrar més estadístiques (com el percentatge d'acceptats respecte el total d'usuaris visualitzats).

La taula haurà de ser generada amb les eines que ofereix la llibreria AWT/SWING de Java. No es poden utilitzar llibreries externes.

Un cop descrites les funcionalitats del servidor, l'apartat següent exposa les funcionalitats dels clients, aquells programes que seran utilitzats pels usuaris i que enviaran informació, mitjançant *sockets*, al programa servidor.

2.2. Client

En aquest apartat es descriuen les opcions que ha d'implementar el programa client. És a dir, del programa que mitjançant la comunicació amb el servidor implementa les funcionalitats que poden executar els usuaris de la plataforma.

2.2.1. Configuració

En engegar el programa, aquest haurà de carregar el fitxer de configuració *config.json* amb la direcció IP i el port de connexió amb el servidor que està obert pels clients. Aquest fitxer ha d'estar ubicat en el directori arrel del projecte del programa Client.

Un cop llegit el fitxer s'obrirà la finestra gràfica, la qual haurà de permetre:

1. Registrar-se i iniciar sessió
2. Editar el perfil
3. Acceptar o descartar usuaris
4. Xatejar amb els usuaris amb els quals s'ha fet match
5. Tancar sessió

Els propers apartats expliquen les característiques de cadascuna d'aquestes opcions.

2.2.2. Registrar-se i iniciar sessió

Quan un usuari executi el programa client, aquest es podrà registrar o bé iniciar sessió.

Per tal de registrar-se al sistema, l'usuari haurà d'introduir la següent informació en un formulari (el format del qual és totalment lliure):

- **Nom d'usuari** (ha de ser únic en el sistema)
- **Edat** (numèric major a 17, ja que no podran ser menors, el límit superior queda a elecció vostra)
- **Tipus de compte** (es podrà triar entre compte Normal i compte Premium)
- **Correu** (el format del correu ha de ser correcte i ser únic en el sistema)
- **Contrasenya** (la contrasenya pot contenir caràcters alfanumèrics incloent minúscules i majúscules així com tot tipus de caràcters especials. Cal que la contrasenya tingui com a mínim una longitud de 8 caràcters així com contingui com a mínim majúscules, minúscules i valors numèrics. No obstant, es recomana la lectura de la política de contrasenyes del MIT com exemple de quines restriccions s'apliquen en entorns reals.)
- **Confirmació de contrasenya** (el contingut d'aquest camp ha de coincidir amb el contingut del camp "contrasenya")

Quan l'usuari cliqui a enviar el formulari, caldrà comprovar que les dades del formulari compleixin els requisits (excepte que el nom d'usuari i el correu siguin únics en el sistema, que es comprovarà quan es faci la petició de registre al servidor) i, en cas que no sigui així, es mostrarà un missatge explicatiu a la interfície gràfica del client. En cas que en el client tot sigui correcte, s'enviarà la petició de registre al servidor el qual processarà la informació i en cas que es validi aquesta, es procedirà a registrar l'usuari en el sistema. Cal destacar que al

servidor caldrà comprovar de nou que la informació rebuda compleixi els requisits pertinents. En cas que les dades no siguin validades correctament en el servidor, caldrà notificar-ho al client i mostrar un missatge d'error explicatiu en la interfície del client. En cas de registre satisfactori, l'usuari podrà autenticar-se i emprar totes les funcionalitats del client a partir d'aquest moment i s'autenticarà automàticament l'usuari en el sistema. Així doncs, l'usuari formarà part del sistema Minder.

Per tal d'iniciar sessió al sistema, l'usuari haurà d'introduir el seu nom d'usuari o el correu amb el qual es va registrar en el seu moment i la contrasenya associada. És important remarcar que s'ha de poder accedir tant amb l'usuari com amb el correu.

Un cop s'enviï el formulari al servidor, aquest el validarà i garantirà accés si les credencials són correctes i, en cas contrari, mostrarà un error genèric (per exemple, "Les credencials introduïdes són incorrectes").

2.2.3. Editar el perfil

A la pantalla principal, un cop autenticats, l'usuari haurà de poder accedir al seu perfil, el qual podrà editar **en qualsevol moment**.

Com a mínim, l'usuari haurà de poder canviar la seva foto de perfil, la seva descripció i indicar si està interessat en el llenguatge Java o en C++. El primer cop que l'usuari inicii sessió, s'haurà **d'obligar** a aquest a que triï una foto de perfil, escrigui una descripció i triï el llenguatge preferit. S'haurà de controlar que aquestes dades siguin emplenades correctament.

Opcionalment, es pot fer que un perfil contingui altres camps com una llista de hobbies, una cançó preferida... Inclús que pugui triar ambdós llenguatges com a interès.

2.2.4. Acceptar o descartar usuaris

A la pantalla principal, un cop autenticats, s'aniran mostrant perfils d'altres usuaris del sistema i l'usuari podrà anar acceptant o descartant aquests usuaris. De cada usuari s'haurà de mostrar la foto, el nom de pila i l'edat. Tot i així, l'usuari haurà de poder accedir a la resta de camps del perfil pitjant un botó.

Si en algun moment l'usuari accepta a un altre usuari i aquest havia acceptat prèviament al primer, haurà d'aparèixer un missatge informatiu (per exemple, "Has fet match amb l'usuari xsole") i a partir d'aquest moment ambdós usuaris podran xatejar en cas que ho desitgin.

Els usuaris han d'aparèixer de manera aleatòria i tenint en compte que ha d'existir un interès mutu en quant al llenguatge de programació, és a dir, si tu has triat Java com a llenguatge preferit, tan sols t'hauran d'aparèixer usuaris que hagin triat també aquesta opció. No es podrà repetir un usuari fins que no s'hagin mostrat tots els possibles. Lògicament, no hauran d'aparèixer ni els usuaris que ja han estat acceptats ni amb els que s'hagi fet match.

És important diferenciar entre els usuaris Normals i els Premium. A un usuari Normal li apareixeran usuaris de manera aleatòria, mentre que a un usuari Premium, li apareixeran primer tots els usuaris que l'han acceptat prèviament i un cop hagi acceptat o descartat a aquests, ja començaran a aparèixer la resta de manera aleatòria.

Opcionalment, podrem aplicar filtres per edat.

Opcionalment, l'usuari podrà acceptar o descartar usuaris fent drag&drop.

2.2.5. Xatejar

De moment, un cop l'usuari inicia sessió pot: acceptar i descartar altres usuaris i editar el seu perfil.

A més, també haurà de poder accedir a una llista de tots els usuaris amb els que ha fet match en algun moment. Accedint a un d'aquests usuaris, hauran de poder xatejar en temps real. S'hauran de guardar com a mínim els últims 20 missatges de cada xat entre dos usuaris.

Des del xat, qualsevol dels dos usuaris ha de poder desfer el match en cas que així ho desitgi, de manera que cap dels dos podrà ja xatejar amb l'altre. Si ho considereu oportú, podeu esborrar els missatges existents entre ambdós usuaris.

Opcionalment, que es guardin il·limitats missatges.

2.2.6. Tancar sessió

Des de qualsevol pantalla de l'aplicació (i en qualsevol moment), els usuaris han de poder tancar la sessió.

Recordeu que qualsevol element compartit com pot ser una foto, una descripció, els missatges d'un xat... Han d'estar sincronitzats en temps real entre tots els usuaris.

3. Consideracions

De forma addicional a les especificacions anteriors, cal tenir en compte les restriccions següents.

Estructures de dades

No és necessari que codifiqueu les estructures de dades, com per exemple una llista dinàmica. Podeu usar les classes del paquet `java.util` de l'API de Java, com les classes `LinkedList<E>`, `ArrayList<E>`, `HashMap<K,V>`, `TreeMap<K,V>` i/o d'altres.

D'aquesta manera obtindreu, "gratuïtament", estructures de dades amb funcionalitats interessants, lliures d'errors i ràpides, amb el que agilitzareu molt el procés de codificació de la pràctica.

Dibuix de gràfiques

Com ja s'ha comentat, per tal de dibuixar les gràfiques no es pot fer ús de cap llibreria externa al SDK de Java (JDK). Cal utilitzar els mecanismes que ofereix la llibreria AWT/SWING de Java i escalar els eixos en funció dels valors màxim i mínim del conjunt de dades que es representen.

Disseny gràfic

L'aspecte visual tan del programa servidor com client és totalment lliure, sigueu creatius. Investigueu el potencial que us ofereix Java, utilitzeu menús, separadors, desplegables, colors, creeu imatges personalitzades pels botons, etcètera. Tot disseny és implementable.

En programació, la clau sovint resideix en l'abstracció, pel que podeu intentar generalitzar components amb l'objectiu de reutilitzar-los.

Procés de desenvolupament

Abans de posar les mans sobre el teclat i començar a codificar, penseu. Analitzeu què cal implementar i dissenyeu quines classes i quines relacions tindrà el sistema, agafeu paper i llapis o utilitzeu l'eina StarUML per tal de crear un diagrama de classes UML base.

Segurament aquest diagrama no serà el definitiu, i anirà canviant a mida que avanceu el desenvolupament, però d'aquesta manera assentareu unes bases sòlides pel desenvolupament del sistema i us estalviareu hores de codificació i molts maldecaps.

El desenvolupament d'un projecte informàtic real típicament comprèn les fases: presa de requeriments, especificació, anàlisi, disseny, implementació, proves i implantació. Podeu pensar que les dues fases inicials d'aquesta seqüència ja han estat realitzades per l'equip docent i que ara us passem el relleu per tal que acabeu el sistema.

El no compliment d'algun dels punts especificats en aquest enunciat suposarà la no acceptació de la pràctica i la devolució de la mateixa.

La detecció de copia comportarà suspendre tots el integrants dels grups implicats, tant els que han copiat com els que han estat copiats, i perdre l'opció de presentar-se a la pròxima convocatòria ordinària. (veure normativa de la universitat)