

PROJECTE

APLICACIÓ XAT

Andreu Reyes & Quim Navarro



Índex

1. Introducció

- Breu introducció al projecte
- Objectiu del projecte
- Breu descripció de les tecnologies utilitzades

2. Requeriment 1

- Tecnologia Comunicacion Utilitzada.
- Notificar als usuaris l'entrada o sortida d'un usuari.
- Enviar missatges públics a tot el grup i privats entre dos usuaris.
- Atendre diversos clients simultàniament.

3. Requeriment 2

- Interfície intuïtiva (menú, barra d'eines i dreceres de teclat).
- Mostrar la llista d'usuaris connectats en qualsevol moment.
- Desenvolupament de components.
- Desenvolupar un instal·lador per a la distribució de l'aplicació.

4. Requeriment 3

- Utilitzar un bean/component propi basat en JFrame.
- Emmagatzemar missatges, dia/hora, i usuaris dels missatges.
- Permetre la selecció de missatges d'un dia en concret.

5. Eines Utilitzades

- Chat GPT
- Youtube
- GitHub
- StackOverFlow
- NetBeans

6. Conclusió

- Recapitulació dels objectius assolits
- Possibles millores futures

1. Introducció

Breu Introducció al Projecte:



El projecte consisteix en el desenvolupament d'una aplicació de xat utilitzant Java Swing per a la interfície gràfica i Java Socket per a la comunicació entre clients i servidor. L'aplicació permetrà als usuaris connectar-se, enviar missatges en temps real i consultar converses passades. El sistema inclou funcionalitats de seguretat, com l'encryptació de les dades i l'emmagatzematge segur de contrasenyes. A més, s'implementarà un component d'accés a dades per a emmagatzemar els missatges utilitzant MongoDB.

Objectiu del Projecte:



L'objectiu principal del projecte és crear una plataforma de xat eficient i segura per a l'ús intern d'una empresa, permetent la comunicació entre els treballadors de manera instantània i fàcil d'utilitzar. A través de l'aplicació, els usuaris podran intercanviar informació de forma pública i privada, mantenint un historial de les converses per a consultes futures. A més, el projecte busca demostrar la integració de diferents tecnologies Java per a desenvolupar una aplicació completa i funcional amb una interfície amigable i atractiva.

2. Requeriment 1

■ Tecnologia comunicacions utilitzada.

El protocol de comunicació usat en l'exemple de codi proporcionat és **TCP (Transmission Control Protocol)**. Aquesta conclusió es basa en les següents observacions:

Ús de ServerSocket i Socket:

En Java, les classes `ServerSocket` i `Socket` s'utilitzen per implementar connexions de xarxa orientades a la connexió, que és una característica del protocol TCP.

Orientació a la connexió:

El servidor crea un `ServerSocket` per escoltar les connexions entrants i accepta connexions creant un nou `Socket` per a cada client que es connecta. Aquesta és una operació típicament associada amb TCP, on s'estableix una connexió persistent entre el client i el servidor.

Fiabilitat i ordre:

TCP és conegut per ser un protocol fiable que assegura que les dades arribin en l'ordre correcte i sense pèrdua. El codi utilitza aquesta fiabilitat per enviar i rebre missatges encriptats entre el client i el servidor.

Control de flux i error:

Tot i que no es veu directament en el codi, TCP gestiona automàticament el control de flux i la correcció d'errors, garantint una comunicació robusta.

Comunicació bidireccional:

Un cop la connexió està establerta, tant el client com el servidor poden enviar i rebre dades de manera contínua, la qual cosa és característic del protocol TCP.

Notificar als usuaris l'entrada o sortida d'un usuari.

Aquest codi defineix un fil per notificar als usuaris d'un xat quan un nou usuari es connecta. A la classe “**Comprovar Estat Client**” s'executa un fil separat i envia un missatge a tots els usuaris connectats, informant-los que un nou usuari s'ha unit al xat, excepte al nou usuari que se li envia la llista d'usuaris ja connectats.

Estructura del Codi

```
public static class ComprovarEstatClient extends Thread {

    private final Usuari usuari;

    public ComprovarEstatClient(Usuari usuari) {
        this.usuari = usuari;
    }

    @Override
    public void run() {
        try {
            for (Usuari user : usuaris) {
                PrintWriter out = new PrintWriter(user.getSocket().getOutputStream(), true);
                if (usuari.getNomUsuari().equals(user.getNomUsuari())) {
                    // Enviar la llista d'usuaris ja connectats al nou usuari
                    for (Usuari u : usuaris) {
                        if (!u.getNomUsuari().equals(usuari.getNomUsuari())) {
                            out.println(u.getNomUsuari());
                            out.println(encryptarMissatge(" s'ha unit al xat"));
                        }
                    }
                } else {
                    // Notificar als usuaris ja connectats sobre el nou client
                    out.println(usuari.getNomUsuari());
                    out.println(encryptarMissatge(" s'ha unit al xat"));
                }
            }
            System.out.println(usuari.getNomUsuari() + " s'ha unit");
        } catch (Exception e) {
            System.out.println("Error al unir-se");
        }
    }
}
```

Constructor:

Inicialitza el fil amb l'usuari que s'ha connectat.

Mètode run:

Recorre la llista d'usuaris connectats.

Envia un missatge a cada usuari connectat (excepte al nou) informant que l'usuari s'ha unit, al nou usuari se li envia el llistat dels usuaris connectats.

Mostra un missatge al servidor confirmant la connexió del nou usuari.

Aquest missatge es mostra en un component del xat per informar a tots els usuaris que un nou participant s'ha connectat, i activa el estat online a la llista d'usuaris.

Vista d'entrada d'un usuari:

————— Andreu s'ha unit al xat —————

- Enviar missatges públics a tot el grup i privats entre dos usuaris.

Aquest codi defineix un fil per enviar missatges. La classe “**Realitzar Enviaments**” envia missatges privats o públics als usuaris connectats, segons sigui necessari.

Estructura del Codi

```
public static class RealitzarEnviaments extends Thread {

    private final String missatge;
    private final String nom;
    private final boolean missPrivat;
    private final String nomR;

    public RealitzarEnviaments(String missatge, String nom, boolean missPrivat, String nomR) {
        this.missatge = missatge;
        this.nom = nom;
        this.missPrivat = missPrivat;
        this.nomR = nomR;
    }

    @Override
    public void run() {
        try {
            String missatgeEncriptat = encriptarMissatge(missatge);
            if (missPrivat) {
                // Enviar missatge privat
                for (Usuari usu : usuaris) {
                    if ((usu.getNomUsuari().equals(nom) || usu.getNomUsuari().equals(nomR))
                        && (usu.getReceptor().equals(nom) || usu.getReceptor().equals(nomR))) {
                        // L'usuari rebrà el missatge en cas de que observi el xat de qui li envia el missatge
                        Socket socket = usu.getSocket();
                        PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
                        System.out.println("Enviant missatge privat a " + usu.getNomUsuari());
                        // Enviar missatge als usuaris connectats
                        out.println(nom);
                        out.println(missatgeEncriptat);
                        out.flush();
                    }
                }
            } else {
                // Enviar missatge grup
                System.out.println("Enviant missatge a tots els usuaris del grup DAM");
                for (Usuari usu : usuaris) {
                    if (usu.getReceptor().equals("DAM")) {
                        // Si l'usuari no està mirant el grup el missatge no se li envia
                        Socket socket = usu.getSocket();
                        PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
                        // Enviar missatge als usuaris connectats
                        out.println(nom);
                        out.println(missatgeEncriptat);
                        out.flush();
                    }
                }
            }
        } catch (Exception e) {
            System.out.println("Error al enviar el missatge");
        }
    }
}
```

Constructor:

Obté el missatge a enviar, qui envia el missatge, si el missatge és privat i a qui va dirigit el missatge.

Mètode run:

Encriptació: Encripta un missatge utilitzant la funció **encriptarMissatge()**.

```
// Encriptar missatge amb AES
private static String encriptarMissatge(String message) throws Exception {
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, clauAES);
    byte[] encryptedBytes = cipher.doFinal(message.getBytes("UTF-8"));
    return Base64.getEncoder().encodeToString(encryptedBytes);
}
```

Missatges Privats:

Recorre la llista d'usuaris (usuaris).

Comprova si l'usuari és el destinatari o l'emissor i comprova que l'usuari destinatari estigui actiu al xat d'on ve el missatge.

Envia el missatge encriptat als usuaris implicats en la comunicació privada.

Missatges Públics:

Recorre la llista d'usuaris.

Comprova si l'usuari està actiu al grup "DAM".

Envia el missatge encriptat a tots els usuaris del grup sempre que aquests siguin dins del xat de DAM.

Aquest codi garanteix que els missatges es distribueixin correctament als destinataris, siguin en comunicacions privades o en grups, assegurant que només els usuaris autoritzats rebin els missatges corresponents.

Missatge Grup:

FitxerEditaVisualitzaOpcionsAjuda

■ Atendre diversos clients simultàniament.

Aquest codi mostra com el servidor de xat és capaç d'atendre múltiples clients simultàniament utilitzant fils. A continuació, s'explica com s'implementa aquesta funcionalitat.

Estructura del Codi

```
public class Servidor {

    private static final int PORT = 7878;
    private static final MongoServeis dbManager = MongoServeis.getInstance();
    private static final ArrayList<Usuari> usuaris = new ArrayList<>();
    private static SecretKey clauAES; // Clau AES compartida
    private static KeyPair serverClauRSA; // Clau RSA del servidor
    private static PublicKey clientClauPublica; // Clau pública del client

    public static void main(String[] args) {
        try {
            // Generar clau RSA del servidor
            KeyPairGenerator clauRSA = KeyPairGenerator.getInstance("RSA");
            clauRSA.initialize(2048);
            serverClauRSA = clauRSA.genKeyPair();

            // Generar clau AES compartida
            KeyGenerator novaClauAES = KeyGenerator.getInstance("AES");
            novaClauAES.init(256);
            clauAES = novaClauAES.generateKey();

            // Obtindre la IP local per al servidor
            InetAddress localIP = InetAddress.getLocalHost();

            ServerSocket serverSocket = new ServerSocket(PORT, 0, localIP);
            System.out.println("Servidor del xat en línia..." + localIP.getHostAddress() + ":" + PORT);

            while (true) {
                Socket socket = serverSocket.accept();

                boolean usuRepetit = false;

                BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
                PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

                // Guardar credencials
                String nom = in.readLine();
                System.out.println(nom + " s'ha connectat");
                if (usuaris.isEmpty()) {
                    for (Usuari usu : usuaris) {
                        if (nom.equals(usu.getNomUsuari())) {
                            usuRepetit = true;
                            break;
                        }
                    }
                }

                if (usuRepetit) {
                    Usuari usuari = new Usuari(nom, socket);
                    usuaris.add(usuari);

                    // Enviar que pot iniciar sessió
                    out.println("OK");

                    // Obtener clau pública del client
                    byte[] byteKey = Base64.getDecoder().decode(in.readLine().trim());
                    X509EncodedKeySpec X509publicKey = new X509EncodedKeySpec(byteKey);
                    KeyFactory kf = KeyFactory.getInstance("RSA");
                    clientClauPublica = kf.generatePublic(X509publicKey);

                    // Xifrar clau AES amb clau pública del client i enviar-li
                    Cipher cipher = Cipher.getInstance("RSA");
                    cipher.init(Cipher.ENCRYPT_MODE, clientClauPublica);
                    byte[] encryptedKey = cipher.doFinal(clauAES.getEncoded());
                    String encryptedAESKey = Base64.getEncoder().encodeToString(encryptedKey);
                    out.println(encryptedAESKey);

                    // Generar fil per comprovar connexions
                    new ComprovarEstatClient(usuari).start();
                    // Generar fil per rebre missatge
                    new Handler(socket, nom).start();
                } else {
                    out.println("Repetit");
                    System.out.println("Usuari ja loguejat");
                }
            }
        } catch (IOException | InvalidKeyException | NoSuchAlgorithmException | InvalidKeySpecException | BadPaddingException | IllegalBlockSizeException | NoSuchPaddingException e) {
            System.out.println("Error en el servidor: " + e.getMessage());
        }
    }
}
```

El servidor escolta connexions entrants en un port específic (PORT).

Quan un client es connecta, el servidor verifica que l'usuari que s'intenta connectar no es a la llista d'usuaris ja connectats, si aquest no hi és l'afegeix, accepta la connexió (`serverSocket.accept()`) i crea un nou Socket per comunicar-se amb el client.

Creació d'Objectes Usuari i Emmagatzematge:

Es crea un nou objecte Usuari per representar el client connectat, emmagatzemant el nom d'usuari i el socket.

L'usuari es guarda a la llista usuaris per mantenir un registre dels usuaris connectats.

Enviament de la Clau AES:

Es genera una clau AES compartida.

La clau pública del client es rep i s'utilitza per encriptar la clau AES.

La clau AES encriptada és enviada al client per assegurar les comunicacions.

Creació i Inici de Fils:

ComprovarEstatClient: Un nou fil que s'encarrega de notificar a tots els usuaris del xat que un nou usuari s'ha unit i al nou usuari li dona la llista del usuaris connectats. Això es fa cridant a `new ComprovarEstatClient(usuari).start();`.

Handler: Un altre fil creat per gestionar la comunicació amb el client recentment connectat. Això es fa cridant a `new Handler(socket, nom).start();`.

Implementació de Fils per Atendre Diversos Clients

Els fils permeten que el servidor pugui gestionar múltiples connexions simultàniament. Cada vegada que un nou client es connecta, es creen aquest dos nous fils.

Aquesta arquitectura assegura que el servidor pot atendre diversos clients de manera concurrent, utilitzant fils per gestionar cadascuna de les connexions de forma independent.

3. Requeriment 2

Interfície intuïtiva (menú, barra d'eines i dreceres de teclat).

Tenim un menú a la barra de dalt, el qual té unes propietats que veurem a continuació, aquestes propietats canvien segons si es la pantalla de login i registre o la pantalla de xatejar:



Fitxer Edita Visualitza Opcions Ajuda

També tenim una barra d'eines:

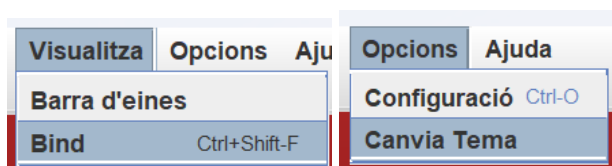
On podem canviar el tema de la interfície, obrir la informació de l'aplicació i obrir el bind.



Aquí tenim les dreceres de teclat, les quals si fas la combinació de tecles faran una funció, en aquest cas són les següents:

- **Ctrl + Q = Sortir.(A les dos pantalles)**
- **Ctrl + D = Eliminar dades del inputs.(Pantalla login)**
- **Ctrl + D = Netejar el xat.(Pantalla xat)**
- **Ctrl + V = Pegar el text del portapapers al input.(Pantalla xat)**
- **Ctrl + Enter = Enviar missatge. (Pantalla xat)**
- **Ctrl + shift + F = Obrir el component Bind.(A les dos pantalles)**
- **F1 = Guia d'usuari.(A les dos pantalles)**
- **Ctrl + I = Informació de l'aplicació.(A les dos pantalles)**
- **Ctrl + P = Veure llistat d'usuaris actius. (Pantalla xat)**

Imatges d'algunes opcions del menú:



Mostrar la llista d'usuaris connectats en qualsevol moment.

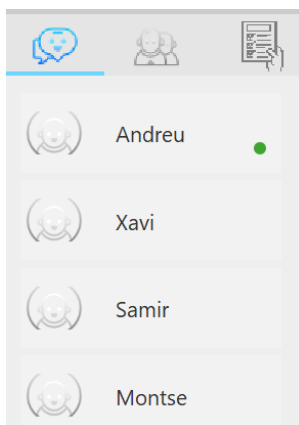
Aquest codi gestiona els canvis de connexió dels usuaris en un xat. Quan un usuari s'uneix o es desconnecta, actualitza la llista d'usuaris connectats i mostra una notificació al cos del xat, assegurant que tots els usuaris estiguin informats en temps real sobre aquests canvis.

Estructura del Codi

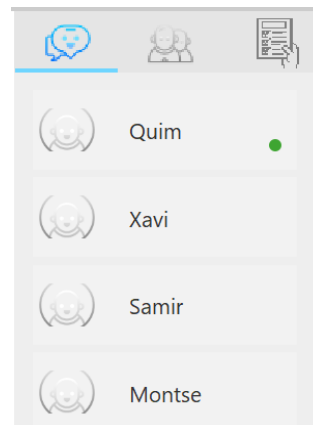
```
if (missatge.equals(" s'ha unit al xat") || missatge.equals(" s'ha desconnectat")) {  
    // Missatge per informar dels usuaris que es connecten o desconnecten al xat  
    SwingUtilities.invokeLater(() -> menuE.setActiu(nom, missatge.equals(" s'ha unit al xat")));  
    if(!nomUsuari.equals(nom)){  
        if(missatge.equals(" s'ha unit al xat")){  
            usuarisConectats.add(nom);  
        } else {  
            usuarisConectats.remove(nom);  
        }  
        SwingUtilities.invokeLater(() -> xatBody.afegirEstat(nom + missatge));  
    }  
}
```

Vista de la llista del usuaris(Online/Offline):

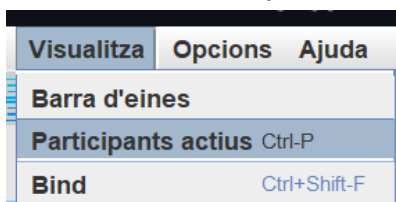
(Vista desde Usuari Quim):



(Vista desde Usuari Andreu):

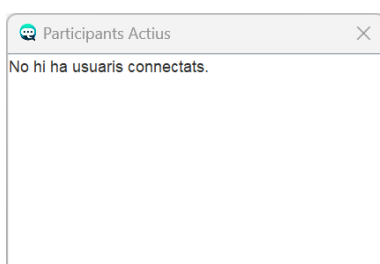


També hi ha una opció al menú:

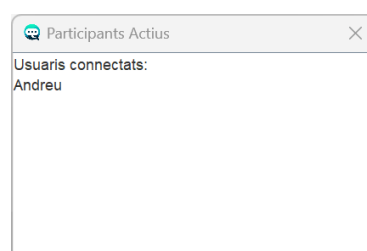


On es veu :

(En cas de no haver hi usuaris)



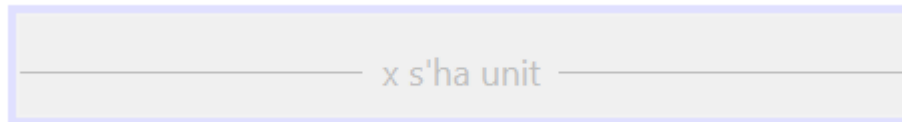
(En cas d'haver hi usuaris)



Desenvolupament de components.

Component XatEstat:

És senzill però efectiu per mostrar un missatge d'estat en una interfície gràfica, acompanyat de línies decoratives a banda i banda. Aquest tipus de component podria ser útil en una aplicació de xat o missatgeria per mostrar notificacions sobre l'activitat dels usuaris, com ara quan un usuari s'ha unit a la conversa.



Estructura del Codi

```
package com.projecte.swing;

/**
 *
 * @author Usuario
 */
public class XatEstat extends javax.swing.JLayeredPane {

    public XatEstat() {
        initComponents();
    }

    public void setEstat(String estat){
        lbEstat.setText(estat);
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        lbEstat = new javax.swing.JLabel();
        linia1 = new com.projecte.components.Linia();
        linia2 = new com.projecte.components.Linia();

        lbEstat.setForeground(new java.awt.Color(191, 191, 191));
        lbEstat.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        lbEstat.setText("x s'ha unit");

        linia1.setForeground(new java.awt.Color(191, 191, 191));

        linia2.setForeground(new java.awt.Color(191, 191, 191));

        setLayer(lbEstat, javax.swing.JLayeredPane.DEFAULT_LAYER);
        setLayer(linia1, javax.swing.JLayeredPane.DEFAULT_LAYER);
        setLayer(linia2, javax.swing.JLayeredPane.DEFAULT_LAYER);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(linia1, javax.swing.GroupLayout.DEFAULT_SIZE, 114, Short.MAX_VALUE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(lbEstat)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(linia2, javax.swing.GroupLayout.DEFAULT_SIZE, 115, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addGap(10, 10, 10)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                        .addComponent(lbEstat, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(linia1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(linia2, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
    }
}

// Variables declaration - do not modify
private javax.swing.JLabel lbEstat;
private com.projecte.components.Linia linia1;
private com.projecte.components.Linia linia2;
// End of variables declaration
}
```

Mètode setEstat

Aquest mètode permet actualitzar el text del JLabel lbEstat que mostra l'estat actual.

Components Declarats

lbEstat: Un JLabel que mostra el missatge d'estat.

linia1 i linia2: Dos components personalitzats de tipus Linia que provenen del paquet com.projecte.components.

Configuració dels Components

El text del JLabel lbEstat es configura per ser de color gris i centrat.

Les línies linia1 i linia2 també es configuren per tenir el color gris.

Els components es col·loquen dins de la JLayeredPane a la capa per defecte.

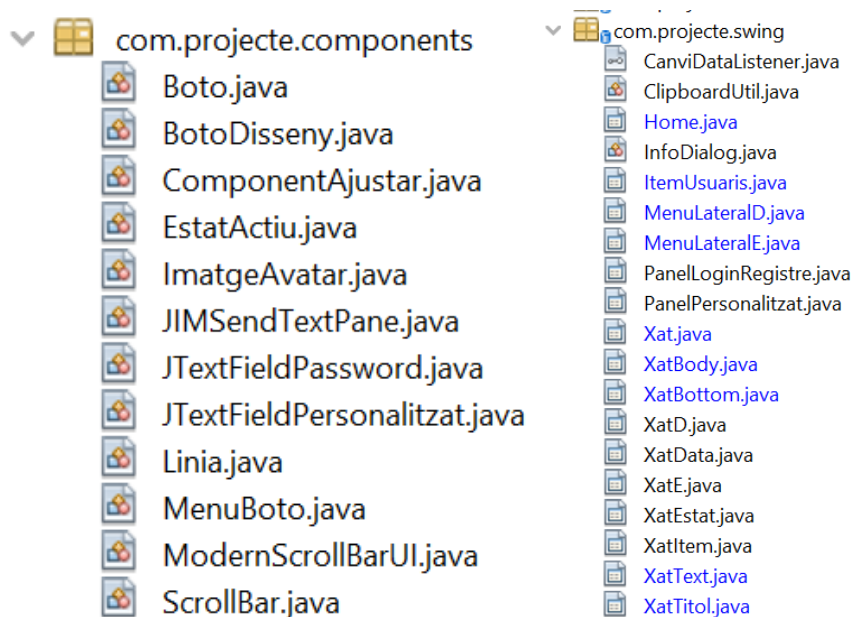
Disseny de la Disposició (Layout)

El GroupLayout s'utilitza per definir la disposició dels components en el XatEstat.

En la disposició horitzontal, linia1 ocupa l'espai a l'esquerra, seguit del lbEstat al centre, i linia2 a la dreta.

En la disposició vertical, tots els components estan alineats i tenen una alçada ajustada.

Altres components utilitzats:



- Desenvolupar un instal·lador per a la distribució de l'aplicació.

Aquest codi configura el projecte Maven per crear un arxiu JAR executable amb les següents especificacions:

Estructura del Codi

Identificació del projecte:

Defineix els identificadors únics del projecte (groupId, artifactId, version) i el tipus d'empaquetat (jar).

```
<groupId>com.mycompany</groupId>
<artifactId>JavaXat</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>
```

Dependències:

Especifiquen les llibreries necessàries per al projecte, com timingframework, miglayout, mongo-java-driver, jcalendar i AbsoluteLayout.

```
<dependencies>
  <dependency>
    <groupId>TimingFramework-0.55</groupId>
    <artifactId>TimingFramework-0.55</artifactId>
    <version>0.55</version>
  </dependency>
  <dependency>
    <groupId>miglayout-4.0</groupId>
    <artifactId>miglayout-4.0</artifactId>
    <version>4.0</version>
  </dependency>
  <dependency>
    <groupId>mongo-java-driver-3.12.14</groupId>
    <artifactId>mongo-java-driver-3.12.14</artifactId>
    <version>3.12.14</version>
  </dependency>
  <dependency>
    <groupId>jcalendar-1.4.jar</groupId>
    <artifactId>jcalendar-1.4.jar</artifactId>
    <version>1.4</version>
  </dependency>
  <dependency>
    <groupId>org.netbeans.external</groupId>
    <artifactId>AbsoluteLayout</artifactId>
    <version>RELEASE190</version>
  </dependency>
</dependencies>
```

Propietats del projecte:

Configura l'encoding, la versió del compilador (source i target) i la classe principal (mainClass).

```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <maven.compiler.source>20</maven.compiler.source>
  <maven.compiler.target>20</maven.compiler.target>
  <exec.mainClass>com.mycompany.javaxat.JavaXat</exec.mainClass>
</properties>
```

Configuració del procés de build:

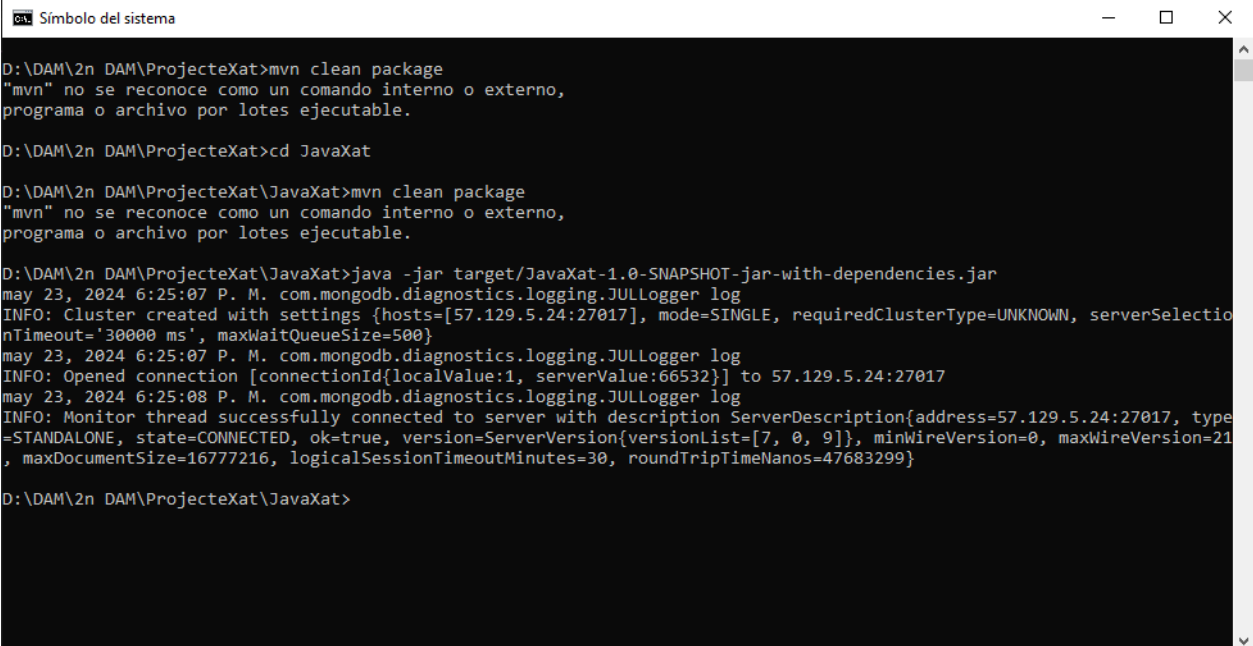
Utilitza el maven-shade-plugin per crear un JAR executable amb totes les dependències incloses (jar-with-dependencies). També especifica els recursos (imatges) que s'han d'incloure en el JAR.

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-shade-plugin</artifactId>
      <version>3.2.4</version>
      <executions>
        <execution>
          <phase>package</phase>
          <goals>
            <goal>shade</goal>
          </goals>
          <configuration>
            <createDependencyReducedPom>false</createDependencyReducedPom>
            <shadedArtifactAttached>true</shadedArtifactAttached>
            <shadedClassifierName>jar-with-dependencies</shadedClassifierName>
            <transformers>
              <transformer implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
                <manifestEntries>
                  <Main-Class>com.projecte.main.LoginRegistre</Main-Class>
                  <Build-Number>123</Build-Number>
                </manifestEntries>
              </transformer>
            </transformers>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>

  <resources>
    <resource>
      <directory>src/main/java/com/projecte/icones</directory>
      <includes>
        <include>/**/*.png</include>
        <include>/**/*.jpg</include>
        <include>/**/*.ico</include>
      </includes>
    </resource>
  </resources>
</build>
```


Procés per crear l'executable:

Desde el cmd al path on es troba el nostre projecte i escriurem **"java -jar nomDelArxiu"**.



```
Símbolo del sistema

D:\DAM\2n DAM\ProjecteXat>mvn clean package
"mvn" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

D:\DAM\2n DAM\ProjecteXat>cd JavaXat

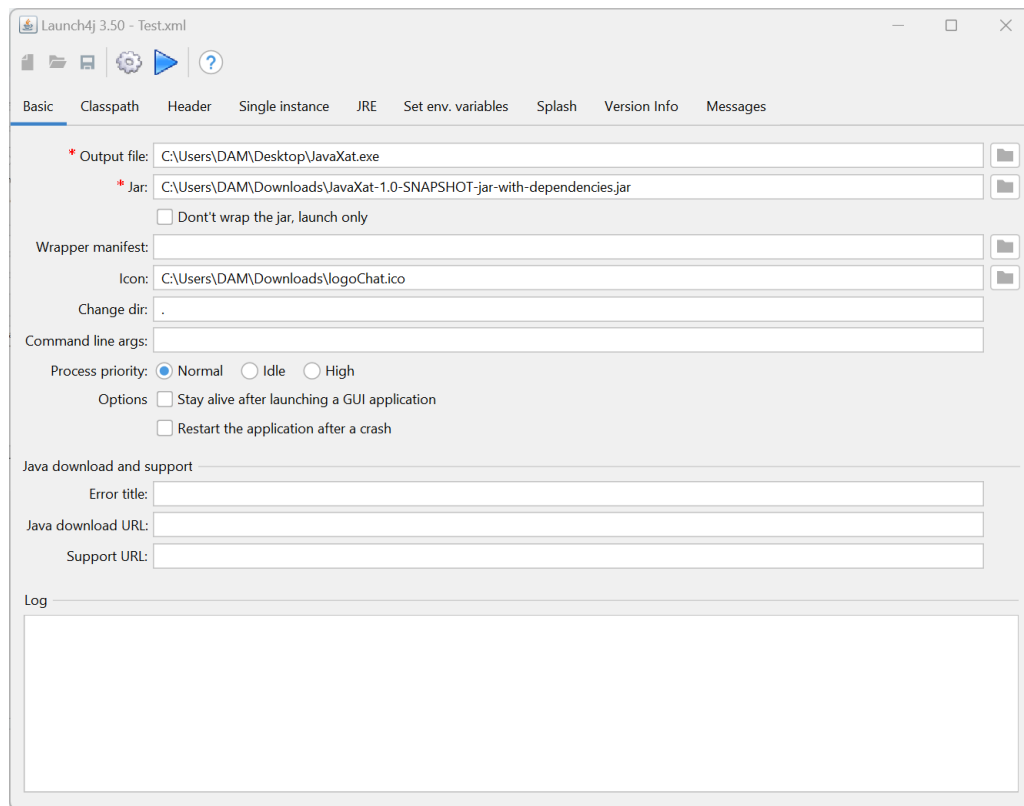
D:\DAM\2n DAM\ProjecteXat\JavaXat>mvn clean package
"mvn" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

D:\DAM\2n DAM\ProjecteXat\JavaXat>java -jar target/JavaXat-1.0-SNAPSHOT-jar-with-dependencies.jar
may 23, 2024 6:25:07 P. M. com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[57.129.5.24:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=500}
may 23, 2024 6:25:07 P. M. com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:66532}] to 57.129.5.24:27017
may 23, 2024 6:25:08 P. M. com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=57.129.5.24:27017, type=STANDALONE, state=CONNECTED, ok=true, version=ServerVersion{versionList=[7, 0, 9]}, minWireVersion=0, maxWireVersion=21, maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30, roundTripTimeNanos=47683299}

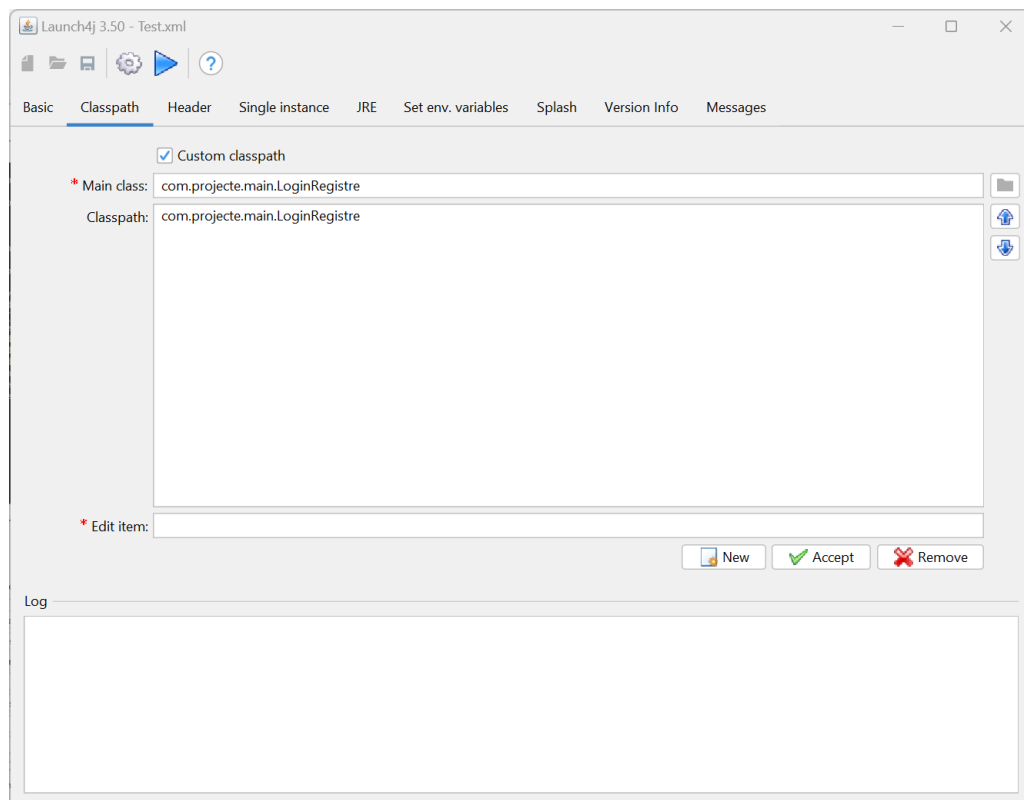
D:\DAM\2n DAM\ProjecteXat\JavaXat>
```

Un cop tinguem l'executable obrirem l'eina **Launch4j** on convertirem el nostre .jar en un .exe.

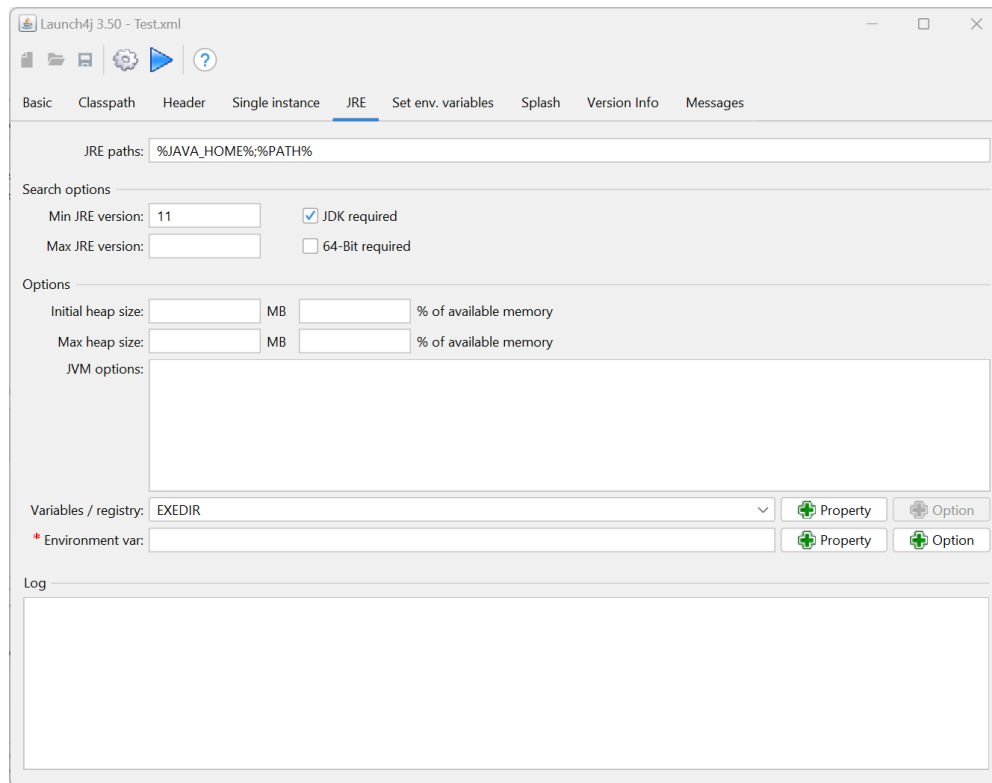
Configuració pàgina Basic.



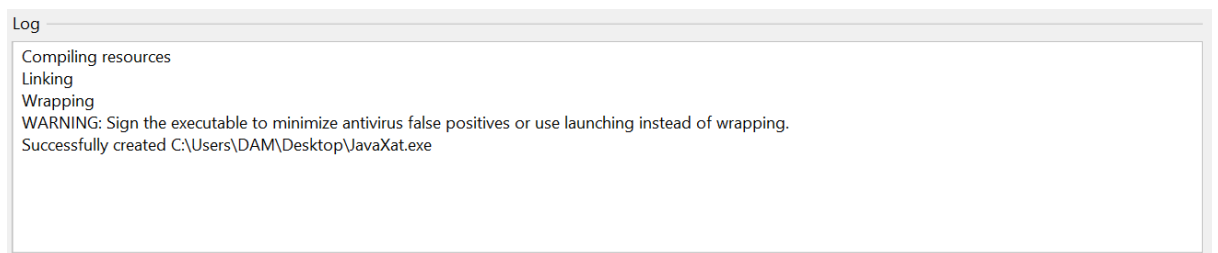
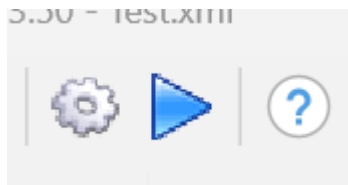
Configuració pàgina Classpath.



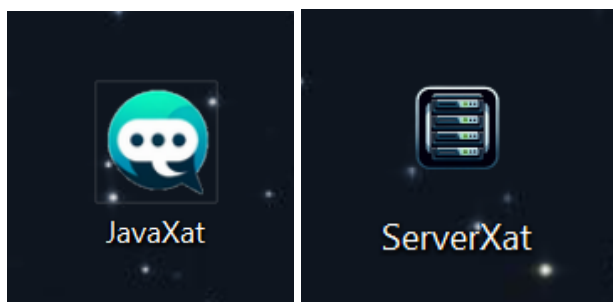
Configuració pàgina JRE.



Per últim li donem a l'engranatge on en generem l'arxiu .exe amb un arxiu xml.



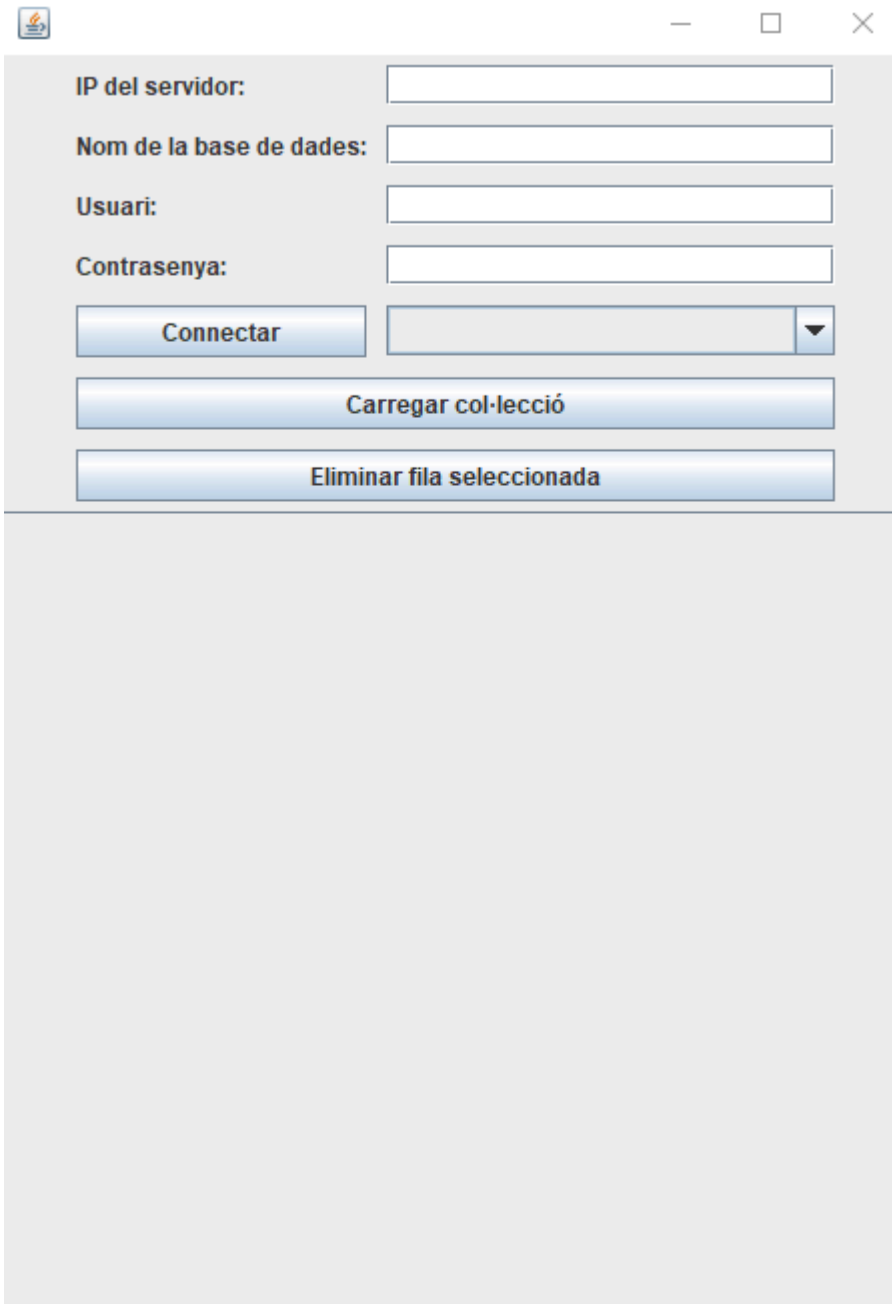
Per la part del servidor realitzarem els mateixos passos. Executables de l'aplicació.



4. Requeriment 3

- Utilitzar un bean/component propi basat en JFrame.

El nostre component personalitzat és un element dissenyat per ser utilitzat amb qualsevol base de dades MongoDB. La seva funcionalitat principal és obtenir automàticament les col·leccions de la base de dades i mostrar-les. Un cop mostrades, la informació d'aquestes col·leccions es pot modificar i eliminar segons sigui necessari.



The image shows a Java Swing window titled "MongoDB" with a standard Mac OS X title bar (red, yellow, and green buttons). The window has a light gray background and contains the following elements:

- Four text input fields with labels: "IP del servidor:", "Nom de la base de dades:", "Usuari:", and "Contrasenya:".
- A "Connectar" button.
- A dropdown menu.
- A "Carregar col·lecció" button.
- An "Eliminar fila seleccionada" button.
- A large, empty gray rectangular area at the bottom of the window, intended for displaying the data from the MongoDB collections.

Ara omplirem les credencials necessàries. Al desplegable de les col·leccions, seleccionarem una col·lecció i farem clic a "carregar col·lecció". Això ens mostrarà la informació de la col·lecció seleccionada.

— □ ×

IP del servidor:

Nom de la base de dades:

Usuari:

Contrasenya:

Connectar

usuaris ▼

Carregar col·lecció

Eliminar fila seleccionada

_id	usuari	contrasenya
664bc3a7d6123954d356e4a2	Andreu	03ac674216f3e15c761ee1a5...
664bc3afd6123954d356e4a3	Quim	03ac674216f3e15c761ee1a5...
6651d9a8ee7b596b43fb2637	Xavi	03ac674216f3e15c761ee1a5...
6651d9c3ee7b596b43fb2638	Samir	03ac674216f3e15c761ee1a5...
6651d9d3ee7b596b43fb2639	Montse	03ac674216f3e15c761ee1a5...
6651ee90c2ac2c2dd4c5bbc8	Quinacla	03ac674216f3e15c761ee1a5...

Ara seleccionar una fila per editar, editarem l'usuari "Quinacla" i li ficarem "Oriol", fen un intro s'actualitza.

—□×

IP del servidor:

Nom de la base de dades:

Usuari:

Contrasenya:

Connectar


▼

Carregar col·lecció

Eliminar fila seleccionada

_id	usuari	contrasenya
664bc3a7d6123954d356e4a2	Andreu	03ac674216f3e15c761ee1a5...
664bc3afd6123954d356e4a3	Quim	03ac674216f3e15c761ee1a5...
6651d9a8ee7b596b43fb2637	Xavi	03ac674216f3e15c761ee1a5...
6651d9c3ee7b596b43fb2638	Samir	03ac674216f3e15c761ee1a5...
6651d9d3ee7b596b43fb2639	Montse	03ac674216f3e15c761ee1a5...
6651ee90c2ac2c2dd4c5bbc8	Oriol	03ac674216f3e15c761ee1a5...

Finalment, eliminarem l'usuari Oriol seleccionant la fila corresponent a aquest usuari i fent clic a "eliminar fila seleccionada".



—

□

×

IP del servidor:

57.129.5.24

Nom de la base de dades:

grup2

Usuari:

grup2

Contrasenya:

.....

Connectar

usuaris

Carregar col·lecció

Eliminar fila seleccionada

_id	usuari	contrasenya
664bc3a7d6123954d356e4a2	Andreu	03ac674216f3e15c761ee1a5...
664bc3afd6123954d356e4a3	Quim	03ac674216f3e15c761ee1a5...
6651d9a8ee7b596b43fb2637	Xavi	03ac674216f3e15c761ee1a5...
6651d9c3ee7b596b43fb2638	Samir	03ac674216f3e15c761ee1a5...
6651d9d3ee7b596b43fb2639	Montse	03ac674216f3e15c761ee1a5...

- Emmagatzemar missatges, dia/hora, i usuaris dels missatges.

Usuari

Estructura del Codi

Aquest codi crea un document amb el nom de l'usuari i el hash de la contrasenya, i l'insereix en una col·lecció de MongoDB. Això permet emmagatzemar de manera segura la informació dels usuaris en la base de dades.

```
public void desarUsuari(Usuari usuari) {  
    String hashedPassword = hashPassword(usuari.getContrasenya());  
    Document document = new Document("usuari", usuari.getNomUsuari())  
        .append("contrasenya", hashedPassword);  
    usuariosCollection.insertOne(document);  
}
```

Estructura de la col·lecció document Usuari:

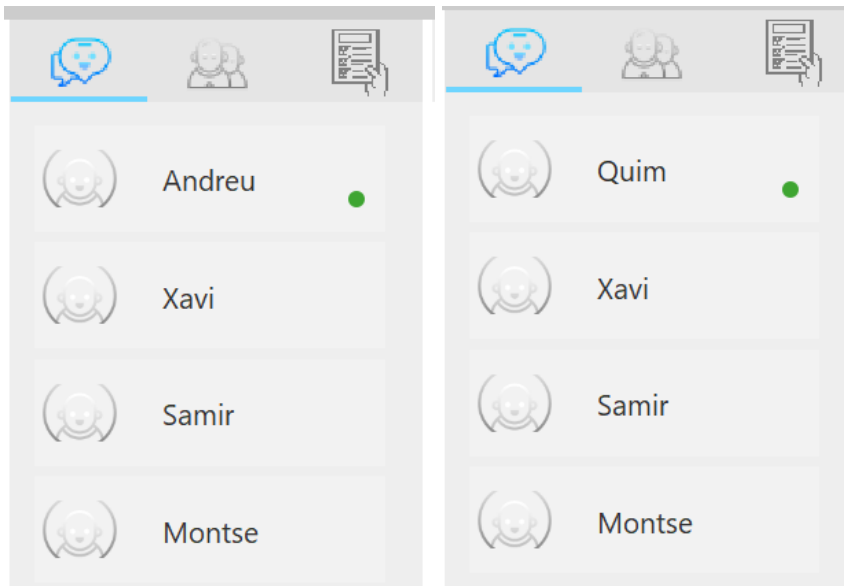
idObjecte: Per defecte de mongo.

usuari: Qui es registra.

contrasenya: Hash de la contrasenya.

```
_id: ObjectId('664bc3afd6123954d356e4a3')  
usuari : "Quim"  
contrasenya : "03ac674216f3e15c761ee1a5e255f067953623c9b389b4459e13f979d7c946f4"
```

Vista app usuaris registrats(exceptuant el personal):



Missatge

Estructura del Codi

Aquest codi crea un document amb la informació del missatge i l'insereix en una col·lecció de MongoDB per emmagatzemar-lo. Això permet guardar missatges en una base de dades per a un ús posterior, com ara historial de xats o anàlisi.

```
public void desarMissatge(Missatge missatge) {  
    Document document = new Document("usuari", missatge.getNomUsuari())  
        .append("missatge", missatge.getMissatge())  
        .append("dataHora", missatge.getDataHora())  
        .append("grup", missatge.getGrup());  
    missatgesCollection.insertOne(document);  
}
```

Estructura de la col·lecció Missatge:

idObjecte: Per defecte de mongo.

usuari: Qui envia el missatge.

missatge: Contingut del que s'envia.

dataHora: Moment quan s'envia.

grup: On s'envia.

```
_id: ObjectId('664d09dbbcb1ae60cf6e09cb')  
usuari: "Quim"  
missatge: "Hola"  
dataHora: 2024-05-21T20:53:47.397+00:00  
grup: "DAM"
```

Vista missatge en app:

De l'usuari que envia:

Hola

22:53

De l'usuari que rep:

Quim

Hola

22:53

- Permetre la selecció de missatges d'un dia en concret.

Estructura del Codi

Aquest codi permet recuperar eficientment tots els missatges privats y públics enviats per un usuari en un xat específic durant una data concreta, utilitzant la base de dades MongoDB. Aquesta funcionalitat és útil per a aplicacions de xat que necessiten accedir a l'historial de missatges per a una anàlisi posterior o per mostrar-lo als usuaris.

```
public List<Missatge> obtenirMissatgesPrivatsIData(String grup, String nom, Date data) {  
    List<Missatge> missatgesGrup = new ArrayList<>();  
    MongoCursor<Document> cursor = missatgesCollection.find(Filters.and(  
        Filters.eq("grup", grup),  
        Filters.eq("usuari", nom),  
        Filters.gte("dataHora", getStartOfDay(data)),  
        Filters.lt("dataHora", getEndOfDay(data))  
    )).iterator();  
    try {  
        while (cursor.hasNext()) {  
            Document doc = cursor.next();  
            String nomUsuari = doc.getString("usuari");  
            String missatge = doc.getString("missatge");  
            Date dataHora = doc.getDate("dataHora");  
            missatgesGrup.add(new Missatge(nomUsuari, missatge, dataHora, grup));  
        }  
    } finally {  
        cursor.close();  
    }  
    return missatgesGrup;  
}
```

Vista missatges diferents dies:
(21/05/2024):

Eines

Andreu

Xavi

Samir

Montse

DAM

Actiu

aaaaaaa

15:57

Andreu

Holaaa Quim, que tal?

17:42

Molt bé i tu?, que faras avui Andreu?

17:43

Hola

22:53

Andreu

Hola

22:53

Andreu

IEEEE

22:54

Andreu

Holaaa

22:54

Hola

22:54

Escriu un missatge aquí...

May

2024

M... Tue ... Thu Fri Sat Sun

18

1

2

3

4

5

19

6

7

8

9

10

11

12

20

13

14

15

16

17

18

19

21

20

21

22

23

24

25

26

22

27

28

29

30

31

(22/05/2024):

Eines

Andreu

Xavi

Samir

Montse

DAM

Actiu

Andreu

Hola bon dia

15:13

Hola

15:13

Andreu

AAAAAAA

23:18

EEEE

23:18

Andreu

SSSSSS

23:19

Escriu un missatge aquí...

May

2024

M... Tue ... Thu Fri Sat Sun

18

1

2

3

4

5

19

6

7

8

9

10

11

12

20

13

14

15

16

17

18

19

21

20

21

22

23

24

25

26

22

27

28

29

30

31

5. Eines utilitzades

Chat GPT:

S'ha utilitzat per a obtenir idees, solucionar problemes tècnics i generar fragments de codi específics relacionats amb Java Swing, sockets i MongoDB. Ha estat útil per resoldre dubtes puntuals i obtenir exemples concrets d'implementació.

Youtube:

S'han consultat tutorials i vídeos explicatius sobre l'ús de Java Swing per crear interfícies gràfiques, l'ús de sockets per a la comunicació en xarxa, i la integració amb MongoDB per a l'emmagatzematge de dades. Els vídeos han proporcionat una comprensió visual i detallada de les tecnologies.

GitHub:

S'ha utilitzat per gestionar el codi font del projecte, mantenint un registre de tots els canvis realitzats i facilitant la col·laboració entre els membres de l'equip. També s'ha consultat repositoris públics per obtenir exemples de projectes similars.

StackOverFlow:

S'han resolt dubtes tècnics específics mitjançant la consulta de preguntes i respostes existents. La comunitat de desenvolupadors ha proporcionat solucions concretes a problemes relacionats amb Java Swing, sockets i MongoDB.

NetBeans:

S'ha utilitzat com a entorn de desenvolupament integrat (IDE) per escriure, provar i depurar el codi del projecte. NetBeans ha facilitat la gestió del projecte, la integració de biblioteques i la construcció d'executables.

6. Conclusió

El desenvolupament de l'aplicació de xat ha estat un projecte ambiciós que ha permès integrar diverses tecnologies i habilitats de programació, amb l'objectiu de proporcionar una eina funcional i segura per a la comunicació interna dins d'una empresa. Durant aquest projecte, hem aconseguit complir tots els requeriments especificats, demostrant la nostra capacitat per desenvolupar aplicacions completes i robustes.

Recapitulació dels objectius assolits

Implementació de Funcionalitats Clau:

Hem desenvolupat un sistema de connexió segur que permet als usuaris iniciar sessió, enviar i rebre missatges en temps real.

Els missatges es poden enviar de manera pública a tot el grup o de manera privada entre dos usuaris, amb totes les comunicacions encriptades per garantir la seguretat.

Interfície d'Usuari Intuïtiva:

Hem creat una interfície gràfica d'usuari (GUI) atractiva i fàcil d'utilitzar, que inclou menús, barra d'eines i dreceres de teclat.

La GUI notifica als usuaris dels canvis de connexió i permet veure la llista d'usuaris connectats en temps real.

Gestió Eficient de Dades:

S'han implementat components personalitzats per a l'emmagatzematge i la gestió de dades, utilitzant MongoDB per garantir la persistència de les converses.

La funcionalitat de visualització permet seleccionar i veure els missatges d'un dia en concret, facilitant l'accés a l'història de xats.

Escalabilitat i Robustesa:

L'arquitectura del servidor, basada en fils, permet atendre múltiples clients simultàniament, assegurant una experiència fluida per a tots els usuaris connectats.

Possibles millores futures

Millores en la Interfície d'Usuari:

Assegurar una paleta de colors i tipografia consistent.

Millorar el mode fosc amb millor contrast i llegibilitat.

Assegurar disseny responsiu per diferents mides de pantalla.

Millores en l'Experiència d'Usuari:

Implementar transicions suaus entre pantalles.

Afegeix notificacions en temps real per nous missatges.

Permet reaccions als missatges amb emojis.

Millores Funcionals:

Afegeix una funció de cerca per trobar missatges o usuaris ràpidament.

Permet l'edició de missatges enviats.

Millora la gestió dels xats de grup amb controls d'administrador millorats.

Seguretat i Rendiment:

Implementar xifratge de cap a cap per a missatges.

Optimitzar l'aplicació per temps de càrrega més ràpids.

Implementar una gestió d'errors completa i missatges d'error clars.

Funcions Addicionals:

Integrar trucades de veu i vídeo.

Permetre que els usuaris estableixin actualitzacions d'estat.

Permetre temes personalitzats per a l'aplicació de xat.