

# Trabalho POO- ChatApp

Gabriel Borges e Matheus Giordani

Trabalho final da disciplina de Programação Orientada a  
Objetos

# Aplicativo de Chat em Java (WhatsApp-POO)

Interface inspirada no WhatsApp, feita com Java Swing

Foram implementadas 5 funções principais:

Criar grupo

Criar conversa

Enviar mensagem

Silenciar / ativar som da conversa

Buscar conversas pelo campo de busca

# Interfaces

MenuApp.java:

```
public JMenuBar criarMenu() { 1 usage
    JMenuBar mb = new JMenuBar();

    // menu arquivo
    JMenu arquivo = new JMenu( s: "Arquivo");
    JMenuItem novoChat = new JMenuItem( text: "Nova conversa");
    JMenuItem criarGrupo = new JMenuItem( text: "Criar grupo");
    JMenuItem sair = new JMenuItem( text: "Sair");

    arquivo.add(novoChat);
    arquivo.add(criarGrupo);
    arquivo.addSeparator();
    arquivo.add(sair);

    // menu editar
    JMenu editar = new JMenu( s: "Editar");
    JMenuItem silenciar = new JMenuItem( text: "Silenciar conversa");
    editar.add(silenciar);
```

# Interface

## PainelConversas.java:

```
public class PainelConversas extends JPanel { 2 usages
    private final JList<Conversa> listaConversas; 1 usage
    private final JTextField campoBusca = new JTextField(); 4 usages
    private final ControladorChat controlador; 2 usages

    public PainelConversas(JList<Conversa> listaConversas, ControladorChat controlador) { 1 usage
        super(new BorderLayout());
        this.listaConversas = listaConversas;
        this.controlador = controlador;
        this.setPreferredSize(new Dimension( width: 350, height: 700));

        // painel superior (busca)
        JPanel topoEsquerdo = new JPanel(new BorderLayout( hgap: 8, vgap: 8));
        topoEsquerdo.setBorder(new EmptyBorder( top: 8, left: 8, bottom: 8, right: 8));
        campoBusca.putClientProperty("JTextField.placeholderText", "Procurar conversas");
        topoEsquerdo.add(campoBusca, BorderLayout.CENTER);
        add(topoEsquerdo, BorderLayout.NORTH);

        //lista de Conversas
        listaConversas.setCellRenderer(new RenderizadorConversa());
        listaConversas.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        listaConversas.addListSelectionListener( ListSelectionEvent e -> {
            if (!e.getValueIsAdjusting()) {
                Conversa selecionada = listaConversas.getSelectedValue();
                controlador.abrirConversa(selecionada);
            }
        });
    };
}
```

# Interface

AplicativoChat.java:

```
JList<String> lista = new JList<>(nomes);
lista.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);

int op = JOptionPane.showConfirmDialog(
    parentComponent: this,
    new JScrollPane(lista),
    title: "Selecionar participantes",
    JOptionPane.OK_CANCEL_OPTION,
    JOptionPane.PLAIN_MESSAGE,
    getIconImage() != null ? new ImageIcon(getIconImage()).getScaledInstance( width: 48, height: 48, Image.SCA
);

if (op != JOptionPane.OK_OPTION) return;
```

# Funções

ControladorChat.java:

```
// enviar mensagem

public void enviarMensagem(String texto) { 1 usage
    if (conversaAtual == null || texto.isBlank()) return;

    Mensagem msg = new Mensagem(texto.trim(), Mensagem.Lado.EU);
    modeloMensagens.addElement(msg);
    conversaAtual.mensagens.add(msg);
    conversaAtual.ultimaMensagem = msg.texto;
    conversaAtual.ultimoHorario = LocalDateTime.now();
```

# Funções

ControladorChat.java:

```
// silenciar / dessilenciar conversa atual
public void alternarSilencio() { 2 usages
    if (conversaAtual == null) return;
    conversaAtual.silenciada = !conversaAtual.silenciada;

    if (callbackRepintarListaConversas != null) {
        callbackRepintarListaConversas.run();
    }
}
```

# Funções

ControladorChat.java:

```
// criar contato/conversa
public void criarConversa(String nome) { 1 usage
    if (nome == null || nome.isBlank()) return;
    Conversa conv = new Conversa(nome.trim());
    adicionarSeed(conv);
}
```

# Funções

ControladorChat.java:

```
public void criarGrupo(String nomeGrupo, List<Conversa> membros) { 1 usage
    if (nomeGrupo == null || nomeGrupo.isBlank()) return;
    if (membros == null || membros.isEmpty()) return;

    Conversa grupo = new Conversa( titulo: nomeGrupo + " (Grupo)", isGroup: true);
    adicionarSeed(grupo);
}
```

# Funções

ControladorChat.java:

```
// filtro da lista da esquerda
public void filtrarConversas(String textoBusca) { 1 usage
    String q = (textoBusca == null) ? "" : textoBusca.trim().toLowerCase();
    modeloConversas.clear();

    if (q.isEmpty()) {
        for (Conversa c : todasConversas) {
            modeloConversas.addElement(c);
        }
    } else {
        for (Conversa c : todasConversas) {
            boolean matchTitulo = c.titulo != null && c.titulo.toLowerCase().contains(q);
            boolean matchUltMsg = c.ultimaMensagem != null && c.ultimaMensagem.toLowerCase().contains(q);
            if (matchTitulo || matchUltMsg) {
                modeloConversas.addElement(c);
            }
        }
    }
}
```

# Referências:

Aula de Java 090 - JList, componente de lista. YouTube, 2025. Disponível em: [https://www.youtube.com/watch?v=UI\\_HnR3ZTd0](https://www.youtube.com/watch?v=UI_HnR3ZTd0). Acesso em: 10 nov. 2025.

YouTube, 2025. Disponível em:

<https://www.youtube.com/watch?v=Ghb0owCoaEc>. Acesso em: 11 nov. 2025.

JETBRAINS. IntelliJ IDEA - recurso “Autocomplete”. Disponível em:

<https://www.jetbrains.com/idea/>. Acesso em: 10 nov. 2025.

OPENAI. ChatGPT. Disponível em: <https://chat.openai.com/>. Acesso em: 16 nov. 2025.

MANUS. 2025. Disponível em: <https://manus.im/>. Acesso em: 11 nov. 2025.