



Desenvolvimento de ChatBot em Java Swing com Integração ao ChatGPT

Com o avanço da Inteligência Artificial, modelos de linguagem como o **ChatGPT** têm sido amplamente utilizados para desenvolver aplicações interativas.

Este artigo apresenta um **ChatBot desenvolvido em Java**, utilizando **Swing** para a interface gráfica, **OkHttp** para comunicação HTTP e **Gson** para manipulação de JSON.

A aplicação permite que o usuário insira perguntas, selecione o modelo de IA entre **GPT-3.5** e **GPT-4**, visualize a resposta em tempo real e gerencie a chave da API de forma dinâmica.

Este artigo fornece uma visão detalhada da arquitetura da aplicação, explicação técnica dos métodos e boas práticas de desenvolvimento.

2. Arquitetura da Aplicação

A aplicação é estruturada em um único pacote, mas modularizada para facilitar a manutenção e a escalabilidade.

2.1 Estrutura do Pacote

Pacote principal: `com.java.chatGPT`

- **Pergunta_chatGPT_app** → Classe principal responsável pela interface gráfica e controle da aplicação.
- **Credenciais** → Classe que armazena a URL da API da OpenAI.

2.2 Bibliotecas Utilizadas

- **okhttp-4.x.jar** → Para requisições HTTP.
- **gson-2.x.jar** → Para manipulação e parsing de JSON.

2.3 Diagrama de Fluxo da Aplicação

[Interface Gráfica] -> [Entrada do Usuário] -> [Comunicação com API] -> [Processamento da Resposta] -> [Exibição no Painel]



3. Implementação

A seguir, apresentamos a implementação da aplicação e a explicação detalhada de cada método.

3.1 Código-Fonte

```
package com.java.chatGPT;

import java.io.IOException;
import com.google.gson.Gson;
import com.google.gson.JsonObject;
import com.google.gson.JsonElement;
import com.google.gson.JsonParser;
import com.java.chatGPT.credenciais.Credenciais;
import okhttp3.*;

public class Pergunta_chatGPT_app extends javax.swing.JFrame {

    public Pergunta_chatGPT_app() {
        initComponents();
        txt_Pergunta.setText("");
        jTextPane_resposta.setText("");
        jTextField_API_SK.setText("");
    }

    private void initComponents() {
        txt_Pergunta = new javax.swing.JTextField();
        lbl_ChatGPT = new javax.swing.JLabel();
        btn_Perguntar_gpt3 = new javax.swing.JButton();
        btn_sair = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextPane_resposta = new javax.swing.JTextPane();
        btn_Limpar = new javax.swing.JButton();
        btn_Perguntar_gpt4 = new javax.swing.JButton();
        jTextField_API_SK = new javax.swing.JPasswordField();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Pergunte ao ChatGPT");

        lbl_ChatGPT.setText("ChatGPT");

        btn_Perguntar_gpt3.setText("Perguntar GPT-3.5");
        btn_Perguntar_gpt3.addActionListener(evt -> enviarPergunta("gpt-3.5-turbo"));

        btn_Perguntar_gpt4.setText("Perguntar GPT-4");
        btn_Perguntar_gpt4.addActionListener(evt -> enviarPergunta("gpt-4"));

        btn_Limpar.setText("Limpar");
        btn_Limpar.addActionListener(evt -> limparCampos());

        btn_sair.setText("Sair");
        btn_sair.addActionListener(evt -> dispose());

        jScrollPane1.setViewportView(jTextPane_resposta);
    }

    private void limparCampos() {
```



```
txt_Pergunta.setText("");
jTextPane_resposta.setText("");
jTextField_API_SK.setText("");
}

private void enviarPergunta(String modelo) {
    String chaveAPI = jTextField_API_SK.getText();
    String pergunta = txt_Pergunta.getText() + " em português";

    try {
        String resposta = getChatGPTResponse(pergunta, modelo, chaveAPI);
        String conteudo = extrairConteudoResposta(resposta);
        jTextPane_resposta.setText(conteudo);
    } catch (IOException e) {
        jTextPane_resposta.setText("Erro ao conectar com a API.");
    }
}

private static String getChatGPTResponse(String prompt, String modelo, String apiKey)
throws IOException {
    String API_URL = new Credenciais().getAPI_URL();
    OkHttpClient client = new OkHttpClient();

    String json = new Gson().toJson(new RequestBodyPayload(modelo, prompt));

    RequestBody body = RequestBody.create(MediaType.parse("application/json; charset=utf-8"), json);
    Request request = new Request.Builder()
        .url(API_URL)
        .post(body)
        .addHeader("Authorization", "Bearer " + apiKey)
        .addHeader("Content-Type", "application/json")
        .build();

    try (Response response = client.newCall(request).execute()) {
        if (!response.isSuccessful()) {
            throw new IOException("Erro na requisição: " + response);
        }
        return response.body().string();
    }
}

private static String extrairConteudoResposta(String jsonResponse) {
    JsonElement jsonElement = JsonParser.parseString(jsonResponse);
    JsonObject jsonObject = jsonElement.getAsJsonObject();
    return jsonObject.getAsJsonArray("choices")
        .get(0).getAsJsonObject()
        .getAsJsonObject("message")
        .get("content").getString();
}

static class RequestBodyPayload {
    String model;
    Message[] messages;

    RequestBodyPayload(String model, String prompt) {
        this.model = model;
        this.messages = new Message[]{new Message("user", prompt)};
    }
}
```



```
static class Message {
    String role;
    String content;

    Message(String role, String content) {
        this.role = role;
        this.content = content;
    }
}

public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(() -> new Pergunta_chatGPT_app().setVisible(true));
}

private javax.swing.JButton btn_Limpar, btn_Perguntar_gpt3, btn_Perguntar_gpt4, btn_sair;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JPasswordField jTextField_API_SK;
private javax.swing.JTextPane jTextPane_resposta;
private javax.swing.JLabel lbl_ChatGPT;
private javax.swing.JTextField txt_Pergunta;
}
```

4. Explicação Técnica do Código

4.1 Interface Gráfica

- initComponents() → Configura os componentes gráficos da aplicação.
- limparCampos() → Reseta os campos de entrada e saída.
- enviarPergunta(String modelo) → Captura a pergunta do usuário, formata a entrada e envia para a API.

4.2 Comunicação com a API

- getChatGPTResponse(String prompt, String modelo, String apiKey):
 - Configura e executa uma requisição HTTP POST para a API da OpenAI.
 - Utiliza **OkHttp** para envio da requisição.
 - Inclui cabeçalhos de autenticação e o payload JSON.
- extrairConteudoResposta(String jsonResponse):
 - Analisa o JSON de resposta da API.
 - Extrai e retorna a mensagem gerada pelo ChatGPT.

4.3 Estrutura do JSON

- RequestBodyPayload:
 - Define o modelo e a estrutura das mensagens enviadas ao ChatGPT.
 - "model" especifica se será usado GPT-3.5 ou GPT-4.
 - "messages" contém o histórico da conversa.



5. Conclusão Técnica

5.1 Aprendizados

- Uso de **Swing** para criar interfaces gráficas.
- Consumo de APIs REST usando **OkHttp**.
- Manipulação de JSON com **Gson**.
- Estrutura modular para escalabilidade.

Para o projeto completo, acesse https://github.com/perucello/Java_ChatGPT_Swing/

5.2 Melhorias Futuras

- **Histórico de mensagens** para conversas contínuas.
- **Melhoria na interface gráfica** para uma melhor experiência do usuário.
- **Autenticação segura da API** com armazenamento criptografado.

Com essa implementação modular e escalável, a aplicação pode ser expandida para diferentes casos de uso, incluindo **assistentes virtuais e sistemas de automação de atendimento**.

EducaCiência FastCode para a comunidade