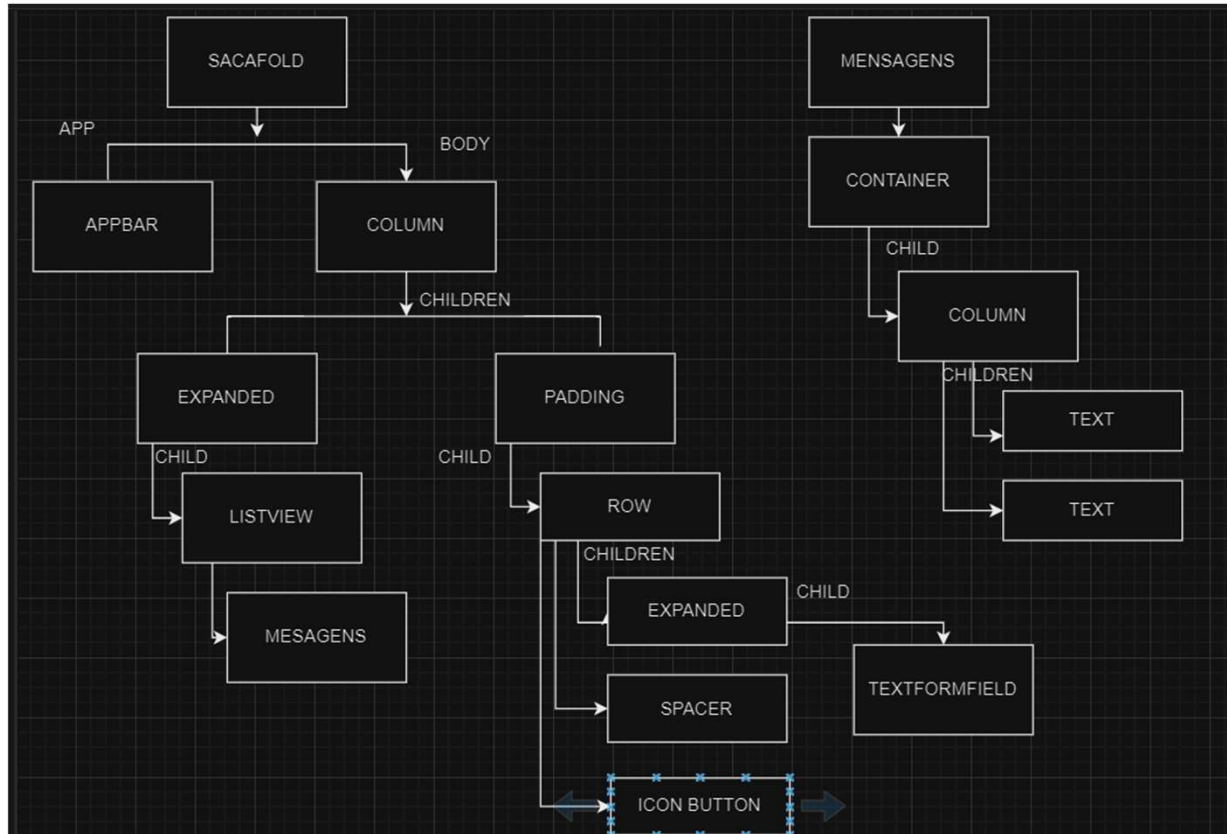


PROJETO Gemini Chat Bot

PROJETO Gemini Chat Bot



Main.dart

The image shows an IDE interface with the following components:

- Project Explorer (Left):** Displays the project structure for 'gemini'. The 'lib' directory is expanded, showing 'chat_screen.dart' and 'main.dart'.
- Editor (Center):** Shows the content of 'main.dart' with the following code:

```
1 import 'package:flutter/material.dart';
2 import 'package:gemini/chat_screen.dart';
3
4 void main() async{
5   runApp(const MyApp());
6 }
7
8 class MyApp extends StatelessWidget {
9   const MyApp({super.key});
10
11   // This widget is the root of your application.
12   @override
13   Widget build(BuildContext context) {
14     return const MaterialApp(
15       title: 'Gemini AI ChatBot',
16       home: ChatScreen(),
17     ); // MaterialApp
18   }
19 }
```
- Running Devices (Right):** Shows a virtual Pixel 3a device running the application. The app's title bar reads 'Gemini Chat Bot'. The screen displays a white background with a text input field at the bottom labeled 'Enter your message' and a black circular button with a white right-pointing arrow.
- Bottom Bar:** Indicates the current file path as 'gemini > lib > main.dart' and shows settings for 'CRLF', 'UTF-8', and '2 spaces'.

Chat_screen.dart

Como obter a API key

<https://aistudio.google.com/app/apikey>

```
lib > chat_screen.dart
1 import 'package:flutter/material.dart';
2 import 'package:google_generative_ai/google_generative_ai.dart';
3 import 'package:intl/intl.dart';
4
5 class ChatScreen extends StatefulWidget {
6   const ChatScreen({super.key});
7
8   @override
9   State<ChatScreen> createState() => _ChatScreenState();
10 }
11
12 class _ChatScreenState extends State<ChatScreen> {
13   final TextEditingController _userMessage = TextEditingController();
14   static const apiKey = "";
15   final model = GenerativeModel(model: 'gemini-pro', apiKey: apiKey);
16
17   final List<Message> _messages = [];
18
19   Future<void> sendMessage() async {
20     final message = _userMessage.text;
21     _userMessage.clear();
22
23     setState(() {
24       // Add user message to the chat
25       _messages
26         .add(Message(isUser: true, message: message, date: DateTime.now()));
27     });
28
29     // Send the user message to the bot and wait for the response
30     final content = [Content.text(message)];
31     final response = await model.generateContent(content);
32     setState(() {
33       // Add bot's response to the chat
34       _messages.add(Message(
```

Chat_screen.dart

```
12 class _ChatScreenState extends State<ChatScreen> {
21   Future<void> sendMessage() async {
35     // Add bot's response to the chat
36     _messages.add(Message(
37       isUser: false, message: response.text ?? "", date: DateTime.now())); // Message
38   });
39 }
40
41 @override
42 Widget build(BuildContext context) {
43   return Scaffold(
44     appBar: AppBar(
45       title: const Text('Gemini Chat Bot'),
46     ), // AppBar
47     body: Column(
48       mainAxisAlignment: MainAxisAlignment.end,
49       children: [
50         Expanded(
51           child: ListView.builder(
52             itemCount: _messages.length,
53             itemBuilder: (context, index) {
54               final message = _messages[index];
55               return Messages(
56                 isUser: message.isUser,
57                 message: message.message,
58                 date: DateFormat('HH:mm').format(message.date),
59               ); // Messages
60             },
61           ), // ListView.builder
62         ), // Expanded
63         Padding(
64           padding:
65             const EdgeInsets.symmetric(horizontal: 8.0, vertical: 15),
66           child: Row(
67             children: [
68               Expanded(
```

Chat_screen.dart

```
68 Expanded(  
69   flex: 15,  
70   child: TextFormField(  
71     controller: _userMessage,  
72     decoration: InputDecoration(  
73       border: OutlineInputBorder(  
74         borderRadius: BorderRadius.circular(50)), // OutlineInputBorder  
75       label: const Text("Enter your message"),  
76     ), // InputDecoration  
77   ), // TextFormField  
78 ), // Expanded  
79 const Spacer(),  
80 IconButton(  
81   padding: const EdgeInsets.all(15),  
82   iconSize: 30,  
83   style: ButtonStyle(  
84     backgroundColor: MaterialStateProperty.all(Colors.black),  
85     foregroundColor: MaterialStateProperty.all(Colors.white),  
86     shape: MaterialStateProperty.all(  
87       const CircleBorder(),  
88     ),  
89   ), // ButtonStyle  
90   onPressed: () {  
91     sendMessage();  
92   },  
93   icon: const Icon(Icons.send),  
94 ) // IconButton  
95 ],  
96 ), // Row  
97 ) // Padding  
98 ],  
99 )); // Column // Scaffold  
100 }  
101 }
```


Chat_screen.dart

```
100 }
101 }
102
103 class Messages extends StatelessWidget {
104   final bool isUser;
105   final String message;
106   final String date;
107   const Messages(
108     {super.key,
109     required this.isUser,
110     required this.message,
111     required this.date});
112
113   @override
114   Widget build(BuildContext context) {
115     return Container(
116       width: double.infinity,
117       padding: const EdgeInsets.all(15),
118       margin: const EdgeInsets.symmetric(vertical: 15).copyWith(
119         left: isUser ? 100 : 10,
120         right: isUser ? 10 : 100,
121       ),
122       decoration: BoxDecoration(
123         color: isUser
124           ? const Color.fromARGB(255, 9, 48, 79)
125           : Colors.grey.shade300,
126         borderRadius: BorderRadius.only(
127           topLeft: const Radius.circular(10),
128           bottomLeft: isUser ? const Radius.circular(10) : Radius.zero,
129           topRight: const Radius.circular(10),
130           bottomRight: isUser ? Radius.zero : const Radius.circular(10),
131         ), // BorderRadius.only
132       ), // BoxDecoration
133       child: Column(
134         crossAxisAlignment: CrossAxisAlignment.start,
```

Chat_screen.dart

```
103 class Messages extends StatelessWidget {
114   Widget build(BuildContext context) {
115     // BoxDecoration
133     child: Column(
134       crossAxisAlignment: CrossAxisAlignment.start,
135       children: [
136         Text(
137           message,
138           style: TextStyle(color: isUser ? Colors.white : Colors.black),
139         ), // Text
140         Text(
141           date,
142           style: TextStyle(color: isUser ? Colors.white : Colors.black),
143         ), // Text
144       ],
145     ), // Column
146   ); // Container
147 }
148 }
149
150 class Message {
151   final bool isUser;
152   final String message;
153   final DateTime date;
154
155   Message({
156     required this.isUser,
157     required this.message,
158     required this.date,
159   });
160 }
161
```


Adicionar as dependências da imagem

