FACULDADE DE TECNOLOGIA DE CARAPICUÍBA

CURSO TECNOLÓGICO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ADENILSON ELIAS DA SILVA
GABRIEL CICCOTTI MONTEIRO DA SILVA
LUIZ ROBERTO ANDRADE DE OLIVEIRA
TACIO DOS SANTOS SOUZA

FATEQUINO

Equipe de Interação

CARAPICUÍBA/SP
2019

SUMÁRIO

1 INTRODUÇÃO

Este trabalho é referente a disciplina de Tópicos Especiais em Informática. Onde a proposta inicial da disciplina consiste em desenvolver um robô chamado Fatequino. Com base nesse robô, foram divididos alguns grupos, onde cada grupo iria focar em uma particularidade:: Visão, Web e Mecânica e Interação, sendo o último o foco de nosso grupo.

2 PROJETO TÉCNICO

Com o objetivo de facilitar a parte da interação com o robô, o que nosso grupo propôs para o professor durante as entregas foi criar um chatbot. Esse chatbot, desenvolvido para dispositivos móveis na linguagem Dart com o framework Flutter, iria funcionar via Web e Bluetooth: os estudantes iriam perguntar coisas cotidianas da faculdade, como se o professor veio, ou o horário de funcionamento da secretaria etc.

2.1 REGRAS DE NEGÓCIO

O sistema deve estar de acordo com as seguintes regras de negócio (RN) para estar de acordo com as especificações.

- RN01 - o sistema deverá responder adequadamente às perguntas do usuário.
- RN02 - o sistema irá aprender de acordo com o uso.
- RN03 - o sistema deverá funcionar com o uso da internet e do Bluetooth.

2.2 REQUISITOS FUNCIONAIS

Nesta seção são apresentados a seguir os requisitos funcionais (RF) que o sistema deve conter para seu correto funcionamento e que atenda à necessidade de gerenciamento da produção de veículos e controle de pedidos.

- RF01 - O sistema irá dar uma resposta de acordo com a pergunta do usuário.
- RF02 - O sistema irá aprender de acordo com o uso.

2.3 REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais (RNF) esperados do sistema para atender às exigências de qualidade são:

- RNF01 - o sistema funcionará via web e bluetooth;
- RNF02 - o sistema será utilizado por alunos da Fatec.
- RNF03 - o sistema será desenvolvido em Dart com o framework Flutter.
- RNF04 - o sistema será desenvolvido para dispositivos móveis.

2.4 DESCRIÇÃO NUMERADA

A seguir será apresentada a descrição numerada de como o sistema funcionará.

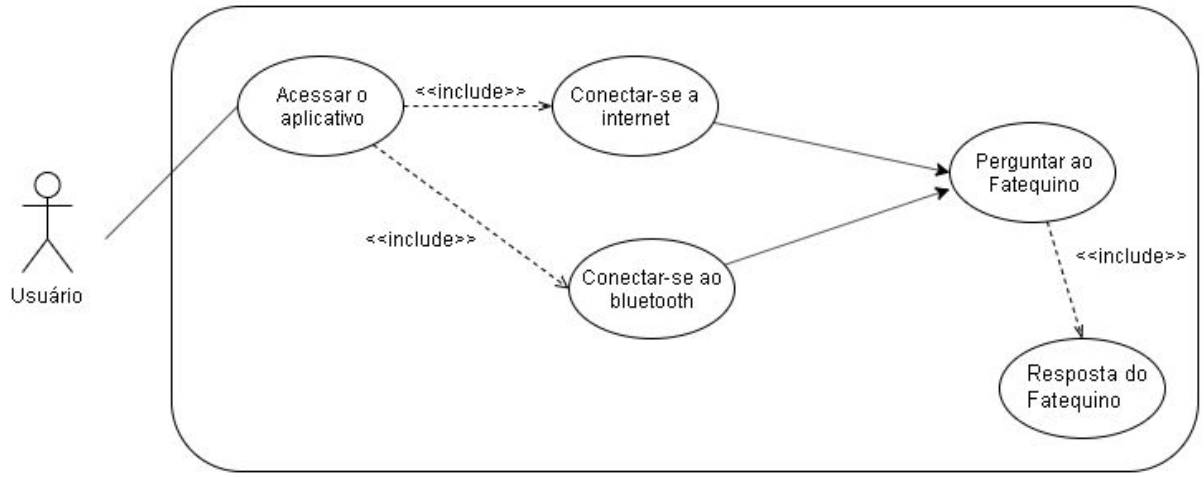
1 - O usuário irá escolher se ele quer utilizar via Web ou via Whatsapp.
2 - O usuário irá perguntar ao Fatequino sua dúvida.
3 - O Fatequino responderá adequadamente.
4 - O Fatequino irá aprender com a resposta do usuário.

2.5 DESCRIÇÃO CONTINUA

O usuário irá escolher se vai utilizar o aplicativo por bluetooth ou via Web. No caso, de ser via Web, ele perguntará ao Fatequino e ele responderá adequadamente. Em modo bluetooth, o usuário terá que estar perto do robô e parelhar seu aparelho à ele. Após isso, fará sua pergunta.

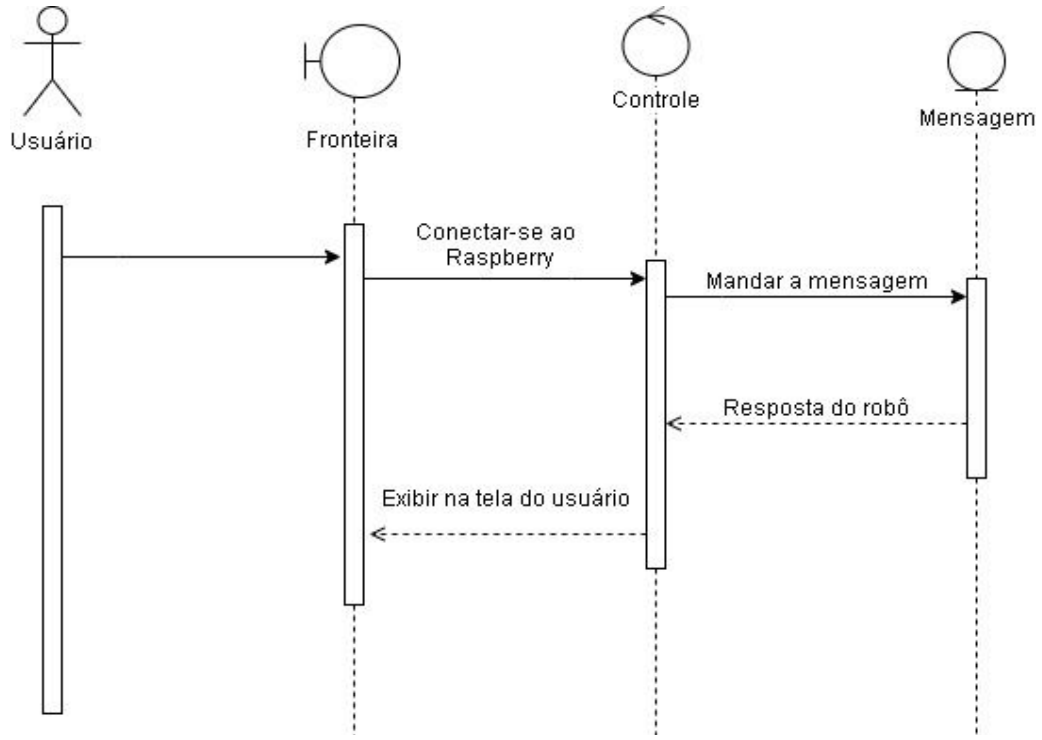
2.6 DIAGRAMA DE CASOS DE USO

A seguir é apresentado o Diagrama de Casos de Uso (DCU) demonstrando a interação dos atores com o sistema e suas funcionalidades.



2.7 DIAGRAMA DE SEQUÊNCIA

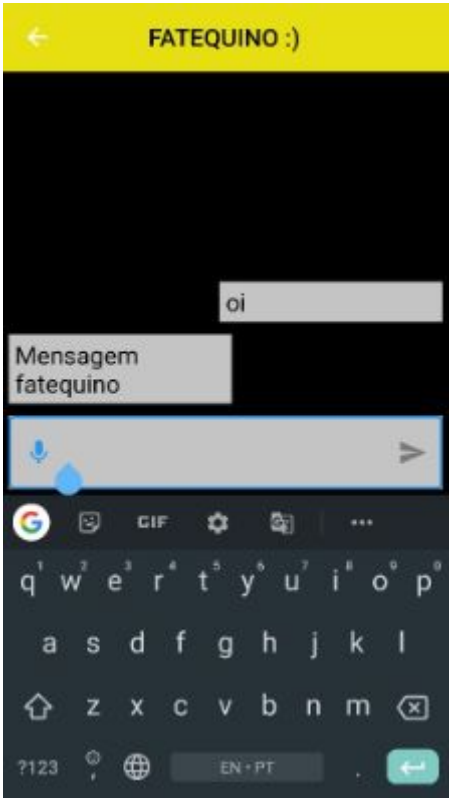
Com a finalidade de representar graficamente o fluxo da informação pelo sistema e como é realizada a interação entre o ator com a fronteira e desta com o controle temos o Diagrama de Sequência (DS).



2.8 PROTÓTIPO NÃO FUNCIONAL

O aplicativo ficou com o seguinte design:

1. Tela de Login



2. *Tela de Chat*

2.9 LINGUAGEM DE PROGRAMAÇÃO

Para o desenvolvimento do sistema foram utilizadas a linguagem de programação Dart, focado no framework Flutter.

2.9.1 Dart / Flutter

Dart é uma linguagem de script voltada à web desenvolvida pela Google em 2011 com o objetivo de substituir o JavaScript como a linguagem principal embutida nos navegadores. Programas nesta linguagem podem tanto serem executados em uma máquina virtual quanto compilados para JavaScript

Flutter é um SDK de código aberto criado pelo Google e lançado a segunda metade de 2018 para o desenvolvimento de aplicativos para Android, iOS, Desktop ou Web.

2.9.2 Porque escolhemos flutter/dart?

Flutter simplifica a criação de app para iOS e Android sendo possível fazer um app para as duas plataforma com apenas um código e tendo desempenho quase que nativo em ambas.

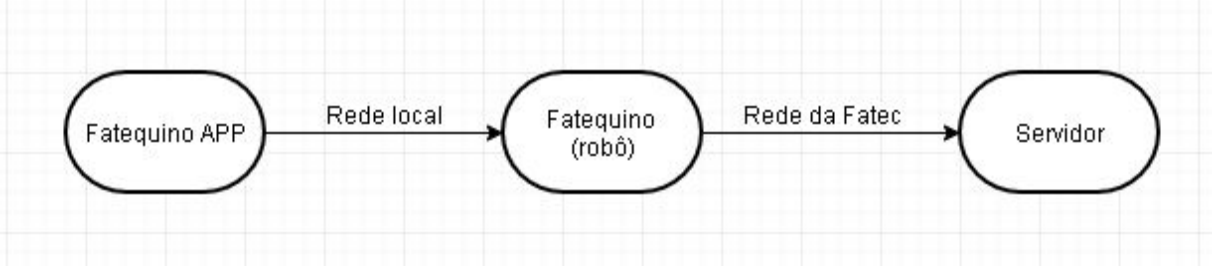
2.10 REPOSITÓRIO

O código ficou versionado em um repositório do Gitlab, com o seguinte link.

https://gitlab.com/adenilson.elias2/fatequinoapp

O código está devidamente comentado, com as explicações de cada funcionalidade.

2.11 FLUXO DE COMUNICAÇÃO



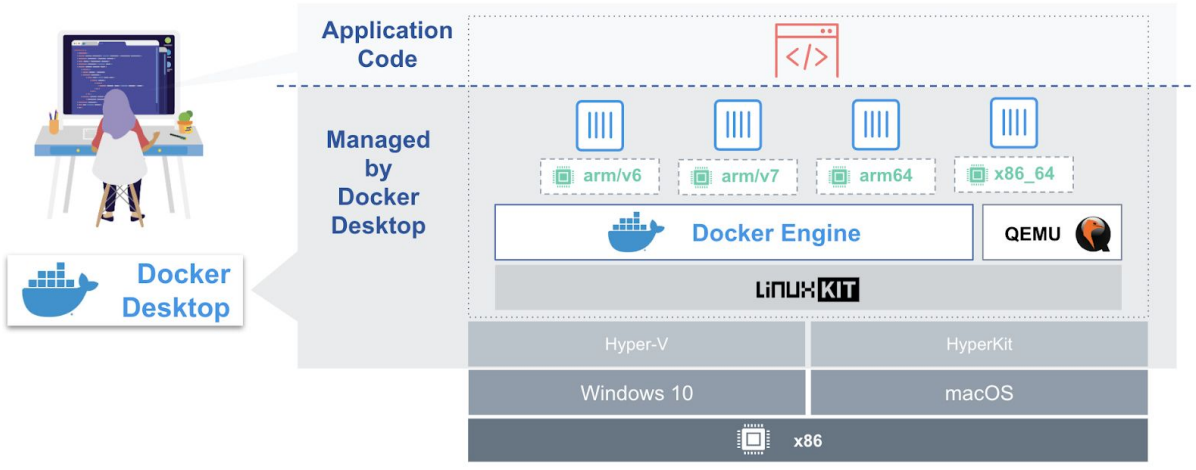
O aplicativo do Fatequino, através da rede local gerada pelo robô, envia as informações ao Robô, e este através da rede própria da Fatec envia a mensagem para o servidor. O servidor processa a mensagem, devolve para o robô e o robô a devolve para o aplicativo.

2.12 SPACY + BLISS X ARM

O SpaCy consiste em uma biblioteca de desenvolvimento com código aberto para processamento avançado de linguagem natural escrita nas linguagens de programação Python e Cython. A biblioteca SpaCy faz uso das bibliotecas Bliss e Cython, utilizada para escrever extensões para a linguagem Python com a linguagem C o que permite grandes melhorias de performance para a biblioteca. Entretanto, segundo as issues #3541, #3861, do projeto SpaCy no Github, #1803, do projeto do ChatterBot no Github, a biblioteca Bliss não possui suporte para a arquitetura de processadores ARM, utilizada pelo processador do Raspberry Pi, bem como o código escrito em linguagem C utilizado pela biblioteca SpaCy. Foi utilizada no Fatequino através da biblioteca ChatterBot, que faz uso do SpaCy. O fato de o SpaCy não rodar sobre arquitetura ARM tornou impossível rodar programas escritos com a biblioteca SpaCy diretamente sobre o Raspberry PI.

2.12.1 Solução 1: Docker Buildx

O Docker é uma poderosa ferramenta para criação e gestão de containers o que permite criar uma virtualização da aplicação mitigando erros de incompatibilidade entre máquinas em ambientes de homologação, desenvolvimento e produção. Em Abril de 2019 o Docker anunciou uma parceria com a ARM para acelerar a adoção de containers sobre arquitetura ARM, e lançou em caráter experimental uma ferramenta para builds em múltiplas arquiteturas.



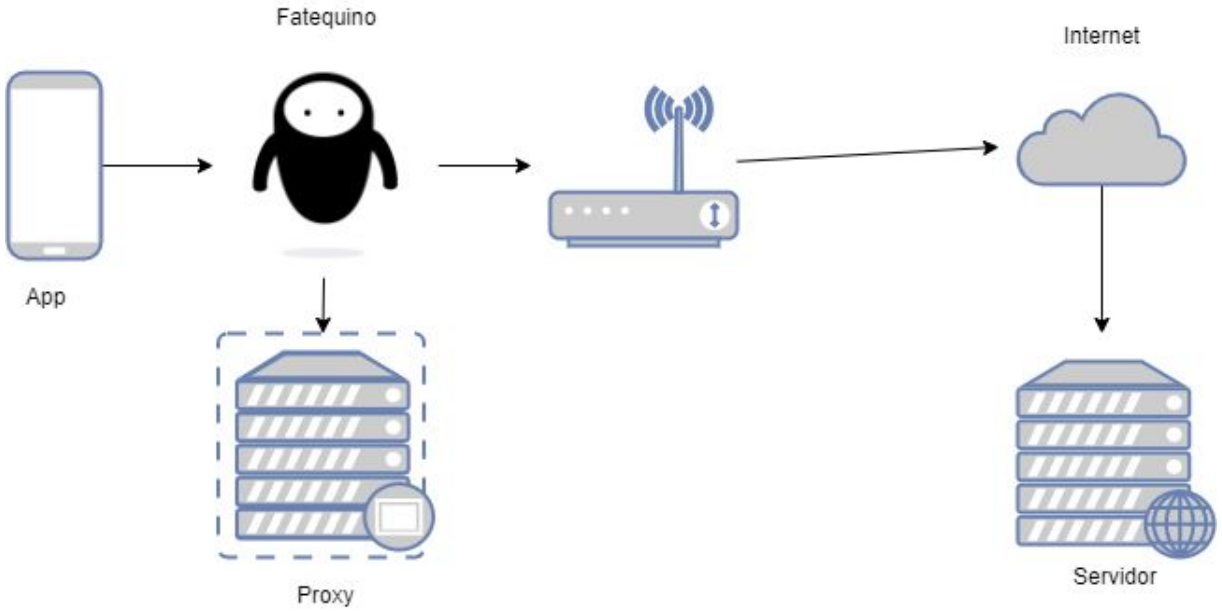
O Docker Buildx permite criar e executar imagens em arquitetura x86 sobre arquitetura ARMv7 fazendo uso do qemu como ferramenta de emulação. Infelizmente por ainda estar funcionando em caráter experimental essa solução se mostrou pouco estável e não chegou a funcionar corretamente no Raspberry Pi.

2.12.2 Solução 2: Raspberry redirecionar para servidor externo.

Optamos por, ao invés de manter um processamento local para a realização da parte de interação com o Fatequino, manter o Raspberry Pi conectado a uma rede wi-fi para quando for receber uma requisição, ele redireciona a requisição para um servidor externo, o mesmo utilizado para manter o bot de presença na WEB. Está solução mostrou-se interessante também por resolver a questão de compartilhamento de conhecimento entre o chatbot presente na Web e o chatbot presente no robô.

2.13 FUNCIONAMENTO DO BLUETOOTH

A utilização do Bluetooth traz algumas desvantagens dentre elas: a maior dificuldade de programação e uma maior limitação de conexões simultâneas. Por isso optamos por realizar a instalação de uma nova placa wifi no raspberry pi o que permitiria ao Fatequino funcionar como um Hotspot WiFi que compartilharia a conexão com a internet para comunicação com chatbot. Para evitar que os alunos usassem a conexão do Fatequino como acesso a internet, seria instalado um proxy no raspberry para limitar as requisições ao servidor com a API do chatbot, e o acesso a api de reconhecimento de voz do google, necessária para o reconhecimento de voz presente no app.



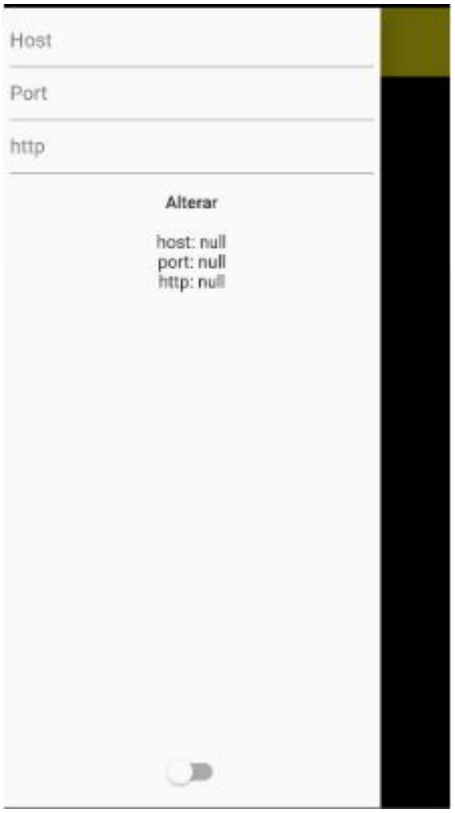
2.14 FUNCIONAMENTO DA API

A API foi desenvolvida em Python e tinha como objetivo receber a mensagem do aplicativo e devolve a resposta do Chatterbot (biblioteca utilizada do Python para auxiliar na criação do Chatbot).

https://github.com/fatequino-manha/INTERACAO

2.14 MANUAL DA CONEXÃO DO FATEQUINO

1. Clique no botão de menu branco localizado na esquerda



2. No menu que será aberto, terão três opções:
    - **Host:** aqui será colocado o IP da máquina que está rodando o código do Python.
    - **Port:** aqui será colocado a porta, no caso, utilizamos a **8080.**
    - **http**: aqui será digitado “**http**”.

Após inserir essas informações clique em “**Alterar**”.

O **botão** localizado no final da página, funciona da seguinte forma: quando ativado, o aplicativo não conversa com a API, o Fatequino sempre devolve uma mensagem específica (sempre a mesma mensagem). Quando desligado, ele conversa com a API do Python.

PS: Optamos por deixar todo o menu desse jeito para facilitar o processo de depuração de código durante o período de desenvolvimento. Em uma versão final, essa configuração não seria necessária, pois as informações de conexão com a API seriam constantes.

2.15 UTILIZAÇÃO DO MONGODB

O banco de dados não-relacional, MongoDB foi utilizado para salvar a base de conhecimento do chatbot. O MongoDB foi escolhido por ser o mais utilizado banco de dados não relacional, apresentar boa performance e ter uma integração nativa com a biblioteca Chatterbot.

3 ANEXOS

O link dos repositórios foram colocados durante a documentação. Porém, para evitar perdas futuras. Todo o código está disponibilizado aqui.

3.1 BackEnd

cli.py

```
1.  from firstbot import chatbot
2.  while True:
3.      try:
4.          bot_input = chatbot.get_response(input())
5.          print(bot_input)
6.
7.      except(KeyboardInterrupt, EOFError, SystemExit):
8.          break
```

Dockerfile

```
1.  FROM python:3.7-alpine
2.
3.  COPY . /app
4.
5.  WORKDIR /app
6.
7.  RUN pip install flask chatterbot pymongo
8.
9.  EXPOSE 8080
10.
11. CMD ["python", "server.py"]
```

firstbot.py

```
1.
2.  from chatterbot import ChatBot
3.  from chatterbot.trainers import ListTrainer
4.
5.  chatbot = ChatBot("Fatequino",
```

```
6.      storage_adapter='chatterbot.storage.MongoDatabaseAdapter',
7.      database_uri='mongodb://localhost:27017/chatterbot-db',
8.      logic_adapters=["chatterbot.logic.BestMatch"])
9.
10.
11. conversa = [
12.     "Oi",
13.     "Olá, eu sou o Fatequino!\nComo posso te ajudar?",
14.     "O professor Mario Marques está na faculdade?",
15.     "Sim!\nO professor Mario Marques está na sala 108.",
16.     "Obrigado.",
17.     "Qualquer dúvida é só me chamar!\nBoa aula!",
18.     "Eu vou passar?",
19.     "Com um BOT lindo como eu claro que sim ; )",
20.     "Eu vou passar em topicos?",
21.     "Se eu estou te respondendo creio que sim"
22. ]
23.
24. treino = ListTrainer(chatbot)
25. treino.train(conversa)
26.
27.
28.
29. # resposta = chatbot.get_response("Oi")
30. # print(resposta)
```

requirements.txt

```
1.  blis==0.2.4
2.  certifi==2019.9.11
3.  chardet==3.0.4
4.  Click==7.0
5.  cymem==2.0.2
6.  Flask==1.1.1
```

```
idna==2.8
itsdangerous==1.1.0
Jinja2==2.10.3
MarkupSafe==1.1.1
mathparse==0.1.2
murmurhash==1.0.2
nltk==3.4.5
numpy==1.17.2
Pint==0.9
plac==0.9.6
preshed==2.0.1
pymongo==3.9.0
python-dateutil==2.7.5
PyYAML==3.13
requests==2.22.0
six==1.12.0
SQLAlchemy==1.2.19
srsly==0.1.0
thinc==7.0.8
tqdm==4.36.1
urllib3==1.25.5
wasabi==0.2.2
Werkzeug==0.16.0

```

server.py

```
from flask import Flask, jsonify, request
from firstbot import chatbot

app = Flask(__name__)

@app.route('/', methods=['POST'])
def post():
    data = request.get_json()
```

```
9.       text = data['text']
10.      resp = chatbot.get_response(text)
11.      # print(type(resp))
12.      return jsonify(dict(text=str(resp))), 200
13.
14.  app.run(host='0.0.0.0', port=8080, debug=True)
```

3.2 Aplicativo

Main.dart

```
1. import 'package:fatequino_app/screen/ChoiceScreen.dart';
2. import 'package:flutter/material.dart';
3.
4. void main() => runApp(MyApp());
5.
6. class MyApp extends StatelessWidget {
7.   /**  Aqui é a raiz da aplicação, onde o app começa.*/
8.   @override
9.   Widget build(BuildContext context) {
10.       return MaterialApp(
11.         title: 'Fatequino app',
12.         debugShowCheckedModeBanner: false,
13.         home: ChoiceScreen(),
14.       );
15.     }
16.   }
```

ChatScreen.dart

```
1. import 'package:fatequino_app/screen/ChoiceScreen.dart';
2. import 'package:fatequino_app/screen/widgets/mensage.dart';
3. import 'package:flutter/material.dart';
4. import 'package:speech_recognition/speech_recognition.dart';
5. import 'style.dart' as style;
```

```
6. import 'package:fatequino_app/services/Apis_call.dart' as api;
7.
8. class ChatScreen extends StatefulWidget {
9.   /**  Tela do chat */
10.     @override
11.     State<StatefulWidget> createState() {
12.       return _ChatScreen();
13.     }
14.   }
15.
16.   class _ChatScreen extends State<ChatScreen> {
17.               static      TextEditingController      txt      =
   TextEditingController();
18.     static FocusNode _focusNode = new FocusNode();
19.     static List<Widget> _chatMensagem = [];
20.     String _text = "";
21.     SpeechRecognition _speech;
22.
23.     bool _speechRecognitionAvaliable = false;
24.
25.     String _currentLocale;
26.
27.     bool _isListening = false;
28.
29.     @override
30.     Widget build(BuildContext context) {
31.       return SafeArea(
32.         child: Scaffold(
33.           appBar: AppBar(
34.             backgroundColor: style.secundaryColor,
35.             title: Text(
36.               "FATEQUINO :)",
37.               style: TextStyle(color: Colors.black),
38.             ),
39.             centerTitle: true,
```

```
40.           ),
41.           backgroundColor: Colors.black,
42.           body: Column(
43.             children: <Widget>[
44.               Expanded(
45.                 child: SingleChildScrollView(
46.                   reverse: true,
47.                   child: Column(
48.                                                mainAxisAlignment:
   MainAxisAlignment.start,
49.                                               crossAxisAlignment:
   CrossAxisAlignment.start,
50.                     children: _chatMensagem,
51.                   ),
52.                 ),
53.               ),
54.               SingleChildScrollView(
55.                 child: Container(
56.                     margin: EdgeInsets.all(5),
57.                     color: style.colorPrimary,
58.                     child: Form(
59.                       child: TextFormField(
60.                         onChanged: (e) {
61.                           setState(() {
62.                             _text = e;
63.                           });
64.                         },
65.                         maxLines: 3,
66.                         minLines: 1,
67.                         controller: txt,
68.                         focusNode: _focusNode,
69.                         keyboardType: TextInputType.multiline,
70.                         decoration: InputDecoration(
71.                           border: OutlineInputBorder(),
72.                           suffixIcon: sendMensagemButton(),
```

```
                          prefixIcon: audiobutton(),
                        ),
                      ),
                    )),
              ),
            ],
          ),
        ),
      );
    }

    Widget sendMensagemButton() {
      /** Widget do botão de enviar mensagem */
      return IconButton(
        icon: Icon(Icons.send),
        color: Colors.yellow,
        onPressed: _text.trim().isNotEmpty
            ? () {
                /** if para escolher se app esta no modo teste ou ele envia algo para api */
                if (!api.ligado) {
                                    api.sendMensagem(mensagem: this._text.trim()).then((value) {
                    Widget meu;
                    Widget fatequino;
                    meu = Mensagem(
                      mensagem: this._text,
                      minha: true,
                    );
                    fatequino = Mensagem(
                      mensagem: value,
                      minha: false,
                    );
                    setState(() {
                      _chatMensagem.add(meu);
```

```
106.                     _chatMensagem.add(fatequino);
107.                     cleanInput();
108.                   });
109.                 }).catchError((erro) {
110.                   debugPrint(erro.toString());
111.                   Navigator.pushReplacement(
112.                     context,
113.                     MaterialPageRoute(
114.                       builder: (context) => ChoiceScreen(),
115.                     ),
116.                   );
117.                 });
118.               } else {
119.                 _chatMensagem.add(
120.                   Mensagem(
121.                     mensagem: this._text,
122.                     minha: true,
123.                   ),
124.                 );
125.                 _chatMensagem.add(
126.                   Mensagem(
127.                     mensagem: "Mensagem fatequino",
128.                     minha: false,
129.                   ),
130.                 );
131.                 cleanInput();
132.               }
133.             }
134.           : null,
135.     );
136.   }
137.
138.   void cleanInput() async {
139.     /** Função para limpa o campo de input quando a mensagem
   é enviada */
```

```
140.      Future.delayed(Duration(milliseconds: 100), () {
141.        this.setState(() {
142.          txt.clear();
143.          this._text = "";
144.        });
145.      });
146.    }
147.
148.    Widget audiobutton() {
149.      /** Widget do botão de gravar audio */
150.      _speech = SpeechRecognition();
151.      _speech.setAvailabilityHandler((bool result) {
152.        setState(() {
153.          _speechRecognitionAvaliable = result;
154.        });
155.      });
156.
157.      _speech.setCurrentLocaleHandler((String locale) {
158.        setState(() {
159.          _currentLocale = locale;
160.        });
161.      });
162.
163.      _speech.setRecognitionStartedHandler(() {
164.        setState(() {
165.          _isListening = true;
166.        });
167.      });
168.
169.      _speech.setRecognitionResultHandler((String texto) {
170.        setState(() {
171.          txt.text = texto;
172.          _text = texto;
173.        });
174.      });
```

```
175.
176.      _speech.setRecognitionCompleteHandler(() {
177.        setState(() {
178.          _isListening = false;
179.        });
180.      });
181.
182.      _speech.activate().then((res) {
183.        setState(() {
184.          _speechRecognitionAvaliable = res;
185.        });
186.      });
187.      return IconButton(
188.          icon: Icon(_isListening ? Icons.stop : Icons.mic),
189.          onPressed: () {
190.                                          _speech.listen(locale:
    _currentLocale).then((value) {
191.              print("$value");
192.            });
193.          });
194.    }
195.  }
```

ChoiceScreen.dart

```
1. import 'package:fatequino_app/screen/ChatScreen.dart';
2. import 'package:fatequino_app/screen/widgets/drawer.dart';
3. import 'package:flutter/material.dart';
4. import 'style.dart' as style;
5.
6. class ChoiceScreen extends StatelessWidget {
7.   /** Tela de escolha da comunicação (Wifi / bluetooth) */
8.   @override
9.   Widget build(BuildContext context) {
10.      return SafeArea(
11.        child: Scaffold(
```

```
12.          drawer: Drawer(
13.            child: DrawerCuston(),
14.          ),
15.          appBar: AppBar(
16.            backgroundColor: style.secundaryColor,
17.            title: Text(
18.              "FATEQUINO :)",
19.              style: TextStyle(
20.                color: Colors.black,
21.              ),
22.            ),
23.            centerTitle: true,
24.          ),
25.          backgroundColor: Colors.black,
26.          body: Container(
27.            child: Column(
28.              mainAxisAlignment: MainAxisAlignment.center,
29.              crossAxisAlignment: CrossAxisAlignment.center,
30.              children: <Widget>[
31.                Container(
32.                  margin: EdgeInsets.all(10),
33.                  alignment: Alignment.center,
34.                  child: Text(
35.                    "Conversa com \n\t\t\to fatequino: ",
36.                                        style:  TextStyle(color:
   style.colorPrimary, fontSize: 30),
37.                    maxLines: 2,
38.                  ),
39.                ),
40.                Container(
41.                  margin: EdgeInsets.all(5),
42.                  child: IconButton(
43.                    onPressed: () {
44.                      Navigator.push(
45.                        context,
```

```
46.                       MaterialPageRoute(
47.                         builder: (context) => ChatScreen(),
48.                       ),
49.                     );
50.                 },
51.                 icon: Icon(Icons.wifi),
52.                 color: Colors.black,
53.               ),
54.               color: style.colorPrimary,
55.             ),
56.             Container(
57.               margin: EdgeInsets.all(10),
58.               child: Text(
59.                 "Web",
60.                 style: TextStyle(
61.                   color: style.colorPrimary,
62.                   fontSize: 30,
63.                 ),
64.               ),
65.             ),
66.             Container(
67.               margin: EdgeInsets.all(5),
68.               child: IconButton(
69.                 icon: Icon(
70.                   Icons.bluetooth,
71.                   color: Colors.black,
72.                 ),
73.                 onPressed: () {},
74.               ),
75.               color: style.colorPrimary,
76.             ),
77.             Container(
78.               margin: EdgeInsets.all(10),
79.               child: Text(
80.                 "Bluetooth",
```

```
                    style: TextStyle(
                      color: style.colorPrimary,
                      fontSize: 30,
                    ),
                  ),
                ),
                Container(
                  child: Text(
                    "Versão 0.10",
                                        style:  TextStyle(color:
  style.colorPrimary),
                  ),
                )
              ],
            ),
          ),
        ),
      );
    }
  }
```

style.dart

```
library fatequino.style;

import 'package:flutter/material.dart';

Color colorPrimary = Color(0xffC4C4C4);
Color secundaryColor = Color(0xffE8DF12);
```

drawer.dart

```
import 'package:flutter/material.dart';
import 'package:fatequino_app/services/Apis_call.dart' as api;

class DrawerCuston extends StatefulWidget {
  /** Drawer de configuraçoes do fatequino para conexão
```

```
6.    * Classe não permanente apenas usada para conectar a api.
7.   */
8.   String host;
9.   String port;
10.    String httpP;
11.    @override
12.    _DrawerCustonState createState() => _DrawerCustonState();
13.  }
14.
15.  class _DrawerCustonState extends State<DrawerCuston> {
16.    @override
17.    Widget build(BuildContext context) {
18.      return Container(
19.        margin: EdgeInsets.all(5),
20.        child: Column(
21.          children: <Widget>[
22.            TextField(
23.              decoration: InputDecoration(hintText: "Host"),
24.              onChanged: (e) => widget.host = e,
25.            ),
26.            TextField(
27.              decoration: InputDecoration(hintText: "Port"),
28.              onChanged: (e) => widget.port = e,
29.            ),
30.            TextField(
31.              decoration: InputDecoration(hintText: "http"),
32.              onChanged: (e) => widget.httpP = e,
33.            ),
34.            FlatButton(
35.              child: Text("Alterar"),
36.              onPressed: () {
37.                setState(() {
38.                  api.host = widget.host;
39.                  api.port = widget.port;
40.                  api.httpP = widget.httpP;
```

```
                });
              },
            ),
            Text("host: ${api.host}"),
            Text("port: ${api.port}"),
            Text("http: ${api.httpP}"),
            Spacer(),
            Switch(
              onChanged: (e) {
                api.ligado = e;
              },
              value: api.ligado,
            )
          ],
        ),
      );
    }
  }
```

mensage.dart

```
import 'package:flutter/material.dart';
import 'package:fatequino_app/screen/style.dart' as style;

class Mensagem extends StatelessWidget {
  /** Widget de mensagem */
  Mensagem({Key key, this.mensagem, this.minha = false});
  String mensagem;
  bool minha;

    @override
    Widget build(BuildContext context) {
      return Align(
              alignment: this.minha ? Alignment.topRight : Alignment.topLeft,
        child: Container(
```

```
15.           color: style.colorPrimary,
16.           width: MediaQuery.of(context).size.width * 0.5,
17.           margin: EdgeInsets.all(5),
18.           padding: EdgeInsets.all(5),
19.           child: Text(
20.             "${this.mensagem}",
21.                style: TextStyle(color: Colors.black, fontSize:
   20),
22.           ),
23.         ),
24.       );
25.     }
26.   }
```

Apis_call.dart

```
1. library fatequino.api;
2.
3. /** Biblioteca para comunicação com as api do fatequino */
4.
5. import 'package:http/http.dart' as http;
6. import 'dart:async';
7. import 'dart:convert';
8.
9. String _host = null;
10.   String _port = null;
11.   String _httpP = null;
12.   bool ligado = false;
13.
14.   String get httpP => _httpP;
15.
16.   set httpP(value){
17.     _httpP = value;
18.   }
19.
20.   String get host => _host;
```

```
21.
22.   set host(value) {
23.     _host = value;
24.   }
25.
26.   String get port => _port;
27.   set port(value) {
28.     _port = value;
29.   }
30.
31.   Future<String> sendMensagem({String mensagem}) async {
32.     print("$host,$port");
33.                     var         response         =         await
   http.post('${_httpP}://${_host}:${_port}/',
34.         body: '{"text":"$mensagem"}',
35.         headers: {"content-type": "application/json"});
36.     if (response.statusCode != 200) {
37.       throw ("${response.body}");
38.     }
39.     var fatequinoResposta = jsonDecode(response.body);
40.     print(response.body);
41.
42.     return fatequinoResposta['text'];
43.   }
```