INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO CAMPUS CUBATÃO
TÉCNICO EM INFORMÁTICA INTEGRADO AO ENSINO MÉDIO

ARTHUR HERNANDES SILVA DE SOUZA
FELIPE DA SILVA SERRALHEIRO GIGANTE
JOÃO BATISTA RUFINO DA SILVA JUNIOR
JULIO BRANDASSE DE ABREU LIMA
KAUÃ PAULINO MARQUES
SABRINA VITÓRIA PEREIRA DE SOUZA RIBEIRO

Chatbot

CUBATÃO
2022

ARTHUR HERNANDES SILVA DE SOUZA
FELIPE DA SILVA SERRALHEIRO GIGANTE
JOÃO BATISTA RUFINO DA SILVA JUNIOR
JÚLIO BRANDASSE DE ABREU LIMA
KAUÃ PAULINO MARQUES
SABRINA VITÓRIA PEREIRA DE SOUZA RIBEIRO

Chatbot

Projeto de Sistemas apresentado ao xxxxxxx em Xxxxxxxxxxxxx da Instituto Federal de São Paulo, como requisito parcial à obtenção do título de xxxxxxxxxxx em Xxxxxxxxxxxx. Área de concentração: Xxxxxxxx.

Orientador: Prof. Maurício Neves Asenjo.

CUBATÃO
2022

ARTHUR HERNANDES SILVA DE SOUZA
FELIPE DA SILVA SERRALHEIRO GIGANTE
JOÃO BATISTA RUFINO DA SILVA JUNIOR
JULIO BRANDASSE DE ABREU LIMA
KAUÃ PAULINO MARQUES
SABRINA VITÓRIA PEREIRA DE SOUZA RIBEIRO

Chatbot

Projeto de Sistemas apresentado ao xxxxxxx em Xxxxxxxxxxxxx da Instituto Federal de São Paulo, como requisito parcial à obtenção do título de xxxxxxxxxxx em Xxxxxxxxxxxx. Área de concentração: Xxxxxxxx

Aprovada em: ___/___/_____.

BANCA EXAMINADORA

_____

Prof. Dr. Xxxxxxxxx Xxxxxxx (Orientador)

Instituto Federal de São Paulo (IFSP)

_____

Prof. Dr. Xxxxxxxxx Xxxxxxx

Instituto Federal de São Paulo (IFSP)

SUMÁRIO

1. APRESENTAÇÃO DA APOSTILA 14
2. APRESENTAÇÃO DO PROJETO 6
3. LINGUAGEM PYTHON 15
4. UM POUCO DA HISTÓRIA 7
5. INTERPRETADOR 7
6. OBJETIVOS 16
7. PREPARANDO O AMBIENTE 16
8. NOSSA PRIMEIRA LINHA DE COMANDO 18
9. VARIÁVEIS E TIPOS DE DADOS 18
10. OPERADORES 19
11. OPERADORES ARITMÉTICOS 19
12. OPERADORES DE COMPARAÇÃO 20
13. ENTRADAS DO USUÁRIO 21
14. ESTRUTURAS LÓGICAS E CONDICIONAIS 21
15. ESTRUTURA DE REPETIÇÃO 22
16. FOR 22
17. WHILE 23
18. FUNÇÕES 23
19. COLEÇÕES 24
20. BIBLIOTECAS USADAS 26
20.1. SPEECH RECOGNITION 26
20.2. CHATTERBOT 27
21. DESENVOLVIMENTO DO CHATBOT 20
22. REFERÊNCIAS 22

LISTA DE FIGURAS

Figura 1. Site Python. **8**
Figura 2. Tela instalação Python. **9**
Figura 3. Tela Python instalado. **9**
Figura 4. Chatbot do Poderoso **20**
Figura 5. Implementações no código **21**
Figura 6. Iniciação dos métodos **22**
Figura 7. Bibliotecas **23**
Figura 8. Front-end **24**
Figura 9. Layout **25**

LISTA DE TABELAS

Tabela 1. Operadores Aritméticos. **19**
Tabela 2. Operadores de Comparação **12**

1. APRESENTAÇÃO DA APOSTILA

Essa apostila tem o objetivo de apresentar alguns conceitos, de forma clara e concreta, a fim de poupar o leitor de questões que não convém, e abordar temas de forma didática, evoluindo o aprendizado geral.

Esperamos que aproveite tal material e contribua para o seu progresso do mesmo, através de comentários, sugestões e críticas. Todos serão bem-vindos.

A apostila deve auxiliar em futuros projetos e ser atualizada. Indique a apostila para um amigo, baixando-a em PDF, para propagar conhecimento.

Esse material foi desenvolvido com base em conhecimentos do curso e será exposto gratuitamente exclusivamente pelo Instituto Federal. Todos os direitos são reservados aos alunos e Instituto Federal. O uso indevido desse material é vedado.

Para uso comercial desse material, consulte o Instituto Federal previamente.

2. APRESENTAÇÃO DO PROJETO

O módulo irá abordar a criação de um **Chatbot** com o intuito de elaborar um programa de inclusão social, que poderá fornecer informações para pessoas com deficiência visual, sem a necessidade do auxílio de outra pessoa, sendo esse nosso principal foco, de incluir e integrar a acessibilidade no contexto da informática e tecnologia.

Vamos abordar o processo, a estrutura e as preocupações especiais para o desenvolvimento de um software de código aberto que poderá ser usado para a integração com outros sistemas, fazendo o uso de uma **API** que será produzida de acordo com um padrão, e disponibilizada para consumo.

O projeto tem a finalidade de minuciar o processo de criação de um Chatbot conversacional e documentar sua criação, apresentando detalhes relevantes da teoria através das etapas.

3. LINGUAGEM PYTHON

Python é uma linguagem de programação muito forte, clara, de alto nível e orientada a objetos. Ao utilizar o **Python**, usufruímos de vários recursos, tais como **a redução de manutenção do código, programação modularizada e reutilização de código,** vem com uma vasta biblioteca padrão, permitindo várias ações comuns de programação, como a conexão de servidores e mais, permite o desenvolvimento em vários sistemas operacionais, os diversos tipos de dados são fortemente tipados, entre outros.

Por ter essa série de recursos, é uma linguagem que tem se expandido, pois é compatível e dá assistência a várias outras linguagens.

4. UM POUCO DA HISTÓRIA

O Python foi criado como uma sucessora da linguagem ABC, afim de ser uma linguagem voltada para programadores. A linguagem ABC foi estabelecida para ser de uso para não programadoras, mas possuía diversas limitações, o que deixou o público um pouco frustrados.

Em solução a essas limitações, *Guido Van Rossum* e alguns outros desenvolvedores criaram a linguagem Python, em 1990, na Holanda, que abrangia diversos componentes e funcionalidades, como listas Python, uso obrigatório de indentação, declarações básicas, entre outros.

5. INTERPRETADOR

O Python é uma linguagem interpretada, como também uma linguagem compilada. O interpretador Python é quem lê os programas e executa as instruções.

A fundo, esse interpretador faz o papel de traduzir um programa de linguagem Python para uma **linguagem de máquina**, permitindo assim, a execução das instruções, isso tudo em tempo de execução, ou seja, passa também pelo processo de compilação. O ***CPython*** faz todos esses passos de forma automática.

6. OBJETIVOS

Normalmente, Python é usada para realizar tarefas de *back-end*, ou seja, tarefas como a realização de consultas no banco de dados, tratamento de regras de negócio, criação de planilhas etc. E por ser uma linguagem de fácil compreensão, foi ascendida no meio científico, possibilitando que, diversas tarefas, fossem realizadas de modo eficiente.

7. PREPARANDO O AMBIENTE

Para começar a desenvolver nossas aplicações, precisamos instalar o Python, pois não vem por padrão no Windows, e para isso, vamos consultar a página de download oficial do Python, através do link: https://www.python.org/downloads/.

Vamos fazer o download da versão mais recente do executável, de acordo com as configurações do sistema operacional.

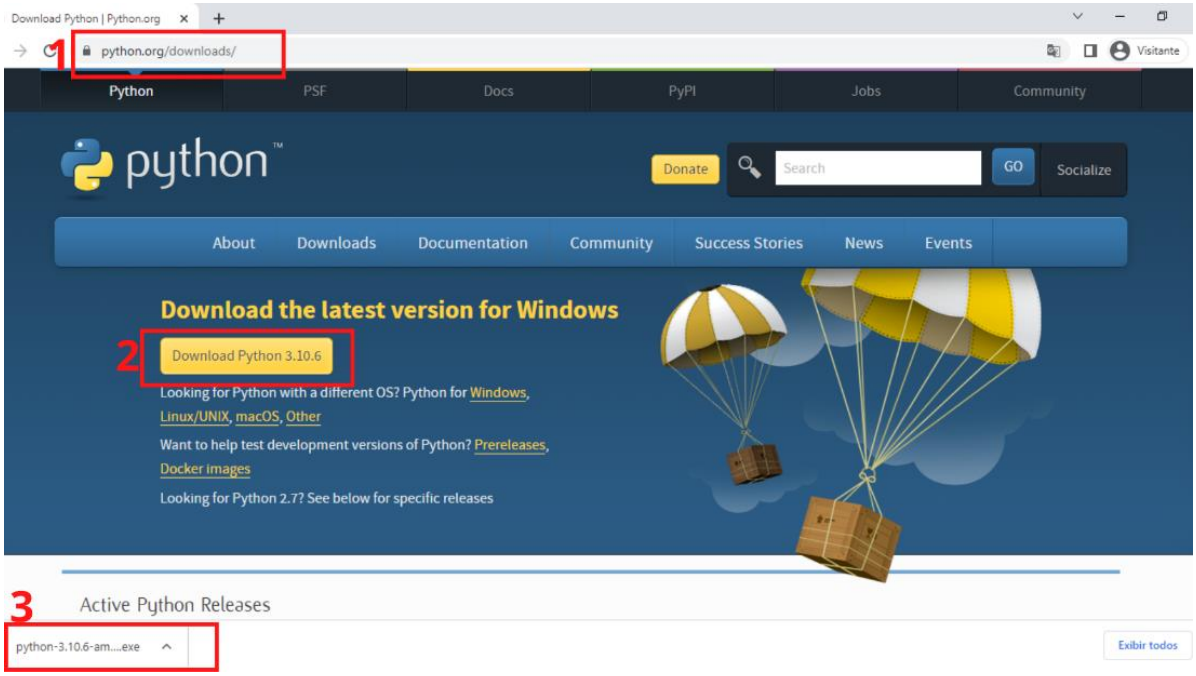


Figura 1. Site Python.

Ao terminar o download, clique duas vezes sobre o arquivo na barra inferior do seu navegador.

Logo após, outra aba será aberta, onde vamos iniciar a instalação do Pyhton. Marque a opção de **“Add Python <versão>”** e depois clique em **“Install now”**. Caso a sua máquina peça permissão, por favor, aceite.

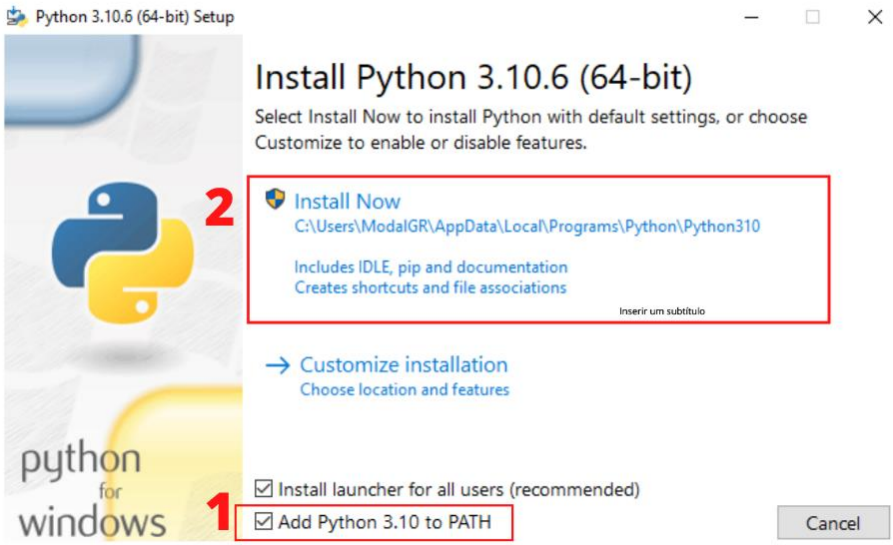


Figura 2. Tela instalação Python.

Ao aparecer essa tela, a instalação do Python estará bem-sucedida.

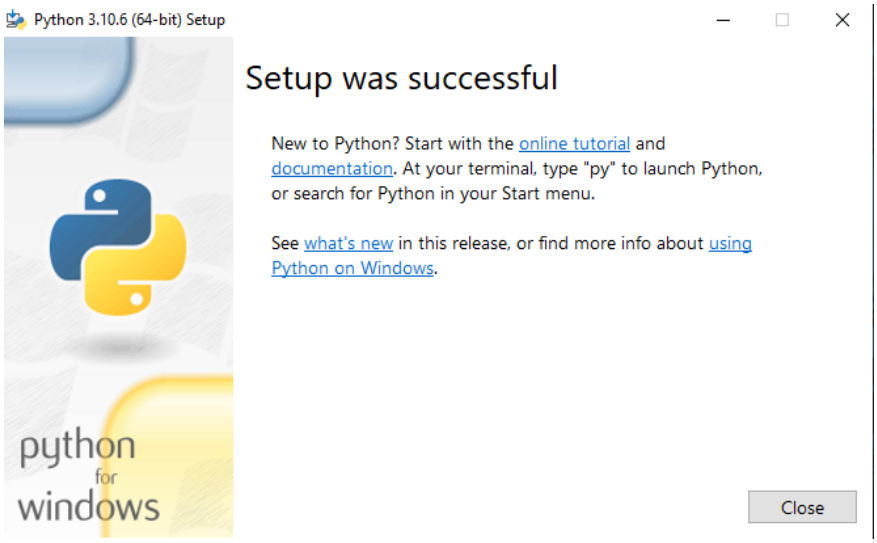


Figura 3. Tela Python instalado.

Além do Python, vamos utilizar uma IDE, ou seja, um ambiente de desenvolvimento integrado, ao qual vamos desenvolver nossas aplicações. Para isso, recomendamos o uso do **Visual Studio Code**, e você pode fazer o download através do link: https://code.visualstudio.com/. Depois de realizar a instalação, podemos dar início ao nosso desenvolvimento.

8. NOSSA PRIMEIRA LINHA DE COMANDO

Com o Visual Studio Code aberto, vamos criar um novo arquivo **main.py**. Ao abrir o arquivo de texto, vamos escrever nossa primeira linha de comando, conforme o exemplo a seguir. Para executar nosso código, abra o terminal e navegue até o arquivo. Depois usamos o comando **pyhton3**.

Entrada:
```
print('Olá Mundo!')
```

Saída:
```
Olá Mundo!
```

9. VARIÁVEIS E TIPOS DE DADOS

Agora vamos abordar os principais tipos internos básicos da biblioteca padrão Python, ou seja, recursos nativos, que são os números e sequências de texto *(strings)*.

Dentro do nosso script, podemos declarar variáveis para usá-las ao decorrer do nosso desenvolvimento. Variáveis são referências que reserva espaços de memória, através de um nome. Quando precisamos usá-la, basta chamar.

Variáveis são criadas de acordo com um tipo, ou seja, se precisamos utilizar um texto, chamamos esse tipo de **str (string),** que sempre estão entre aspas (simples ou duplas). Se precisamos de números, utilizamos **int (inteiros), float (ponto flutuante ou decimal)**, além disso, podemos utilizar outros tipos, como **bool (boleanos), list (lista).** Cada um deles tem suas particularidades.

Pelo Python ser altamente tipado, o interpretador se responsabiliza em declarar o tipo da variável e mudar o tipo se for alterado ao longo do programa, assim, não devemos ter preocupação em relação isso.

A seguir, vamos ver alguns exemplos:

```
Entrada:
nome = 'Maria'
idade = 14
media_notas = 9.5
materias_do_curso = ['Português', 'Matemática', 'Geografia']
aprovada = True
print(type(nome))
print(type(idade))
print(type(media_notas))
print(type(materias_do_curso))
print(type(aprovada))
```

```
Saída:
<class 'str'>
<class 'int'>
<class 'float'>
<class 'list'>
<class 'bool'>
```

10. OPERADORES

Operadores são símbolos reservados utilizados para realizar cálculos ou comparações.

11. OPERADORES ARITMÉTICOS

Operadores Aritméticos são símbolos que tem a função de realizar cálculos.

OPERADORES	DESCRIÇÃO
+	Adição
-	Subtração
*	Multiplicação
**	Exponenciação
/	Divisão
//	Divisão interna
%	Resto da divisão

Tabela 1. Operadores Aritméticos.

Segue alguns exemplos:

```
Entrada:
x = 10;
y = 2;
print(x + y)
print(x - y)
print(x * y)
print(x / y)
print(x**y)
print(x % y)
```

```
Saída:
12
8
20
5.0
100
0
```

12. OPERADORES DE COMPARAÇÃO

São operadores que tem a função de realizar a comparação entre valores. Sendo eles:

OPERADORES	DESCRIÇÃO
==	Igual a
!=	Diferente de
<	Menor do que
>	Maior do que
<=	Menor ou igual a
>=	Maior ou igual a

Tabela 2. Operadores de Comparação

13. ENTRADAS DO USUÁRIO

Podemos capturar os valores que o usuário digita. Para isso, utilizamos a função **input(),** que espera o usuário entrar com algum valor, e quando apertar o **ENTER,** o valor é processado e exibido na tela.

```
Entrada:
nome = input('Digite o seu nome: ')
print('O nome digitado é: ' + nome)
```

```
Saída:
Digite o seu nome: Maria
O nome digitado é: Maria
```

14. ESTRUTURAS LÓGICAS E CONDICIONAIS

As estruturas lógicas e condicionais são utilizadas quando precisamos verificar condições que determinam se o bloco será ou não executado.

As estruturas lógicas, presumem que, dois conjuntos de instruções que poderão ser executados a partir de uma condição. Ou seja, se a condição X for satisfeita, um bloco é executado, caso contrário, outro bloco é executado.

Em Python, utilizamos as palavras reservadas **If-Else** para esses tipos de problemas, vejamos um exemplo a seguir:

```
Entrada:
valor1 = 10
valor2 = 20
if(valor1 > valor2)
    print("O valor1 é maior que o valor2")
else
    print("O valor2 é maior que o valor1")
```

```
Saída:
O valor2 é maior que o valor1
```

15. ESTRUTURA DE REPETIÇÃO

As estruturas de repetição são responsáveis pela execução de um determinado conjunto de ações, que pode ser executado nenhuma ou várias vezes, dependendo da condição de expressão de controle, que é caracterizada por um resultado booleano, ou seja, verdadeiro ou falso.

Geralmente, uma variável de controle também é criada, com o objetivo de mapear o número de iterações nem um laço de repetição. Abordaremos duas principais e mais utilizadas estruturas de repetição.

16. FOR

Utilizamos o For quando precisamos iterar ou percorrer um certo conjunto de dados. Assim, é executado um conjunto de iterações sobre cada item. Expressamos um exemplo a seguir:

```
Entrada:
frutas = ['Abacaxi', 'Maçã', 'Banana']
for item in frutas:
   print(item)
```

```
Saída:
Abacaxi
Maçã
Banana
```

17. WHILE

Utilizamos o While quando queremos executar um determinado bloco enquanto a condição estiver sendo satisfeita, vejamos a seguir:

```
Entrada:
contador = 0
while contador < 3:
   print(f'Contador = {contador}')
   contador += 1
```

```
Saída:
Contador = 1
Contador = 2
Contador = 3
```

18. FUNÇÕES

As funções em Python nada mais são do que códigos estruturados em blocos, identificados por um nome e com a capacidade de receber parâmetro.

Resumidamente, o objetivo de uma função é simplificar a codificação isolando uma determinada tarefa que irá ser usada com grande frequência, tornando assim o corpo do código mais organizado e fluido.

Eis a seguir alguns exemplos de funções altamente requisitadas para automatizar tarefas.

```
Entrada:
def soma(a,b):
return (a+b)
def show(msg):
print msg
```

```
Uso:
P = soma(2,3)
 print P
Sáida: 5
```

Nesse sentido, existem diversas funções para atender diferentes requisitos provenientes de diferentes necessidades, contudo, todos seguem a mesma estrutura.

19. COLEÇÕES

As coleções consistem no agrupamento de vários itens e valores em um único aspecto ou unidade. Normalmente as coleções envolvem aspectos como: adição, pesquisa e entre outros tipos de tarefa.

Dentre a grande massa de opções, é importante salientar algumas das opções mais usadas na plataforma, que são os dicionários, listas e tuplas.

Listas:

Lista é uma coleção de valores indexada, separadas por virgula e dentro de colchetes. Sua utilidade consiste no armazenamento de diversos itens em uma única variável.

```
Entrada:
lista = ['Ovos', 'Peixes', 'Manteiga']
print(lista)
```

```
Saída:
['Ovos', 'Peixes', 'Manteiga']
```

Tuplas:

Um tipo de estrutura semelhante a uma lista, mas cuja particularidade é a imutabilidade. Isso significa que quando uma tupla é criada, seus elementos não podem ser adicionados, alterados ou removidos. A tupla corresponde a um dos problemas da lista que seria a capacidade de alteração livre de seus dados.

Sua sintaxe corresponde ao uso dos parênteses em sua estrutura ao invés dos colchetes que são usados nas listas.

```
Entrada:
tupla = ("Ovos", "Peixe", "Manteiga", )
print (tupla)
```

Observe que a declaração é finalizada com uma vírgula. É a vírgula que é realmente define uma tupla

```
Saída:
("Ovos", "Peixes", "Manteiga")
```

Dicionários:

Um dicionário é uma lista de associações composta por uma chave (de tipo imutável) e estruturas correspondentes às chaves que podem ou não ser mutáveis.

Dessa forma, dicionários em Python são conjuntos de chave-valor (chaves com dados). A sintaxe para a criação de dicionários em Python atende por {chave1 : valor1, chave2 : valor2, … }.

```
Entrada
> dicionario = {'Peixe' : 1, 'Ovos' : 2}
> dicionario
> {'Peixe' : 1, 'Ovos' : 2}
print(len(dicionario))
```

Note que a função "len()" retorna somente o tamanho da lista (o número de chaves):

```
Saída:
> 2
```

Destaca-se que os nomes das chaves não podem ser alterados (‘chave1’:1). São imutáveis. Porém, os valores atribuídos a ela são mutáveis (‘chave1’:1), podem assumir valores de qualquer natureza. Também são abertos para serem apagados, acrescentados etc.

20. BIBLIOTECAS USADAS

20.1. SPEECH RECOGNITION

A biblioteca ***SpeechRecognition*** foi desenvolvida para realizar reconhecimento de voz. O próprio termo Speech Recognition se traduzido para o português tem o significado de reconhecimento de voz. Essa é uma área interdisciplinar da ciência da computação e linguística computacional que tem como objetivo o reconhecimento e a tradução da linguagem falada para em texto por computadores. Utilizamos essa biblioteca para fazer a transformação de um áudio em um texto.

O ato de falar nada mais é do que criar vibrações no ar. Por meio de um conversor analógico-digital (ADC), essas vibrações podem ser convertidas em dados digitais que um computador pode entender. Para fazer isso, ele coleta ou digitaliza o som medindo com precisão a onda em intervalos de frequência. O algoritmo filtra o áudio digitalizado para remover o ruído indesejado e, às vezes, o separa em diferentes bandas de

frequência (a frequência é o comprimento das ondas sonoras que os humanos percebem como diferenças de tom). Este algoritmo também ajusta o som para um nível de volume constante. As pessoas nem sempre falam na mesma velocidade, então o áudio também é ajustado para corresponder à velocidade das amostras de áudio do modelo já armazenadas na memória do sistema.

Após esta etapa inicial, o sinal é dividido em pequenos segmentos de alguns centésimos de segundo ou, conforme o caso, milissegundos. O algoritmo então compara esses segmentos com fonemas conhecidos no respectivo idioma. Um fonema é o menor elemento da linguagem, ou seja, a representação dos sons que fazemos para formar palavras e expressões significativas.

O último passo é examinar os fonemas no contexto dos outros fonemas ao seu redor. Para fazer isso, o algoritmo analisa estatisticamente os fonemas capturados e os compara a uma grande biblioteca de palavras, frases e sentenças conhecidas. O programa determina o que o usuário provavelmente disse e envia como texto ou emite um comando de computador.

20.2. CHATTERBOT

ChatterBot é uma biblioteca python que gera uma resposta à entrada do usuário. Ele usa diferentes algoritmos de *machine learning* para gerar respostas diferentes.

Isso facilita para o usuário criar um chatbot usando a biblioteca do chatterbot para obter respostas mais precisas. O design do chatbot permite que o bot se comunique em vários idiomas, incluindo espanhol, alemão, inglês e vários idiomas regionais. Os algoritmos de *machine learning* também facilitam o aprimoramento do bot usando a entrada do usuário.

21. DESENVOLVIMENTO DO CHATBOT

```python
#import kivy
from logging import exception
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.core.window import Window
from chatterbot import ChatBot
from chatterbot.trainers import ListTrainer
from chatterbot.trainers import ChatterBotCorpusTrainer
import pyttsx3
import speech_recognition as sr

class Manager(ScreenManager):
    pass

class MenuInitial(Screen):
    pass

class HomeScreen(Screen):

    bot = ChatBot('Chatbot do Poderoso')

    list = ChatterBotCorpusTrainer(bot)
    list.train("./frases.yml")

    speak = pyttsx3.init('sapi5')
    voices = speak.getProperty('voices')
    speak.setProperty('voice', voices[2].id)

    rec = sr.Recognizer()

    def __init__ (self, tarefas=[], **kwargs):
        super().__init__(**kwargs)
        for tarefa in tarefas:
            self.ids.box.add_widget(Message(text=tarefa))

```

Figura 4. Chatbot do Poderoso.

```
    #implementa o esc para voltar
    def back(self, window, key, *args):
        if key == 27:
            App.get_running_app().root.current = 'menuinitial'
            return True

    #implementa o esc para sair da aplicação
    def on_pre_leave(self):
        return Window.unbind(on_keyboard=self.back)

    # método que permite adicionar labels dinamicamente
    def addWidget(self):
        texto = self.ids.texto.text
        self.ids.box.add_widget(Message(text=texto))
        self.ids.texto.text = ''

#define o espaço da mensagem na GUI + adiciona o X de exclusão de labels
class Message(BoxLayout):
    def __init__(self, text='', **kwargs):
        super().__init__(**kwargs)
        self.ids.label.text = text

#classe main
class Main(App):
    def build(self):
        return Manager()

Main().run()

```

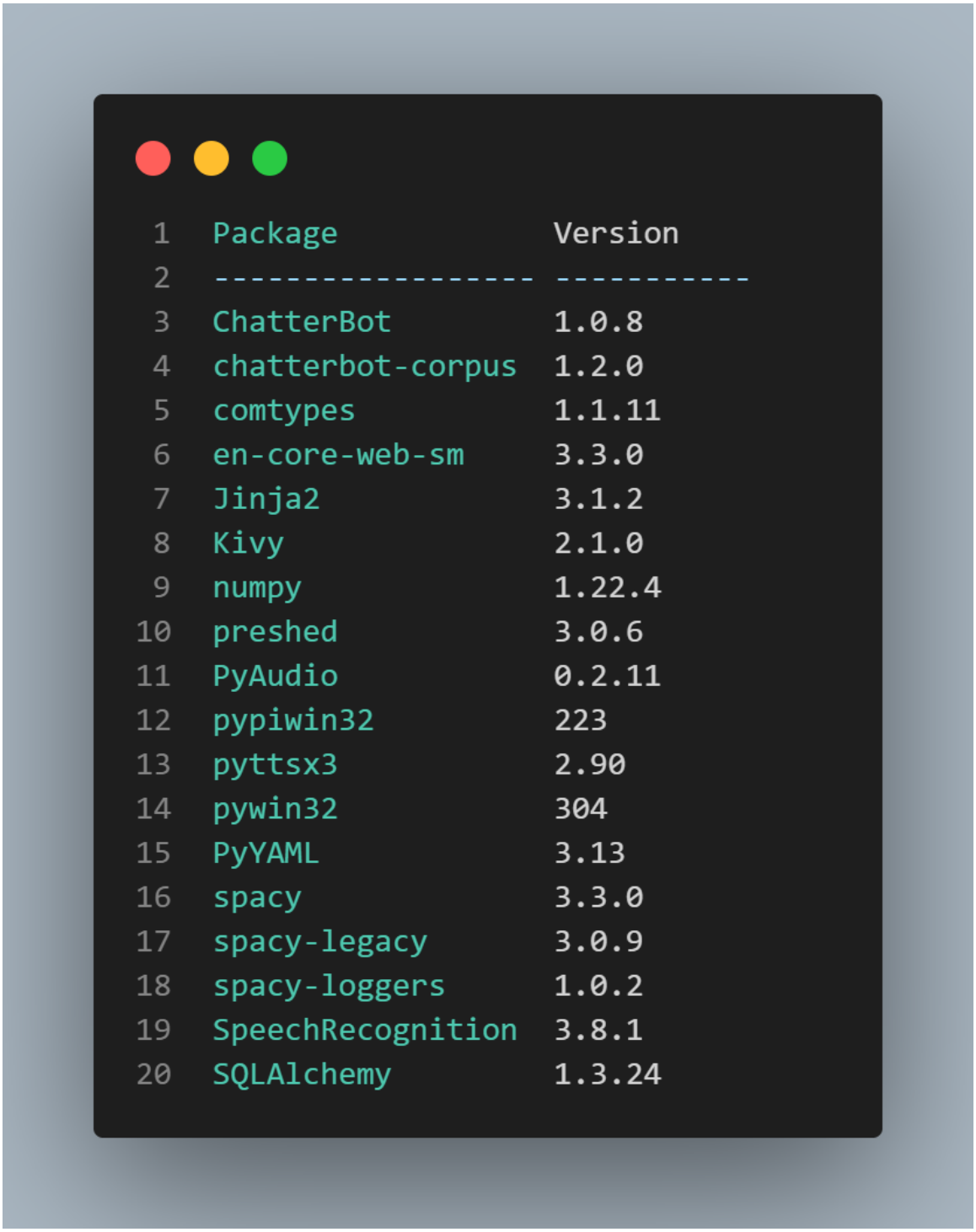
Figura 5. Implementações no código.

Essa parte do código, basicamente, faz a instancia de duas das nossas bibliotecas principais, sendo elas Chatterbot (utilizada para construir o bot através de funções e métodos pré-programados) e Pyttsx3 (utilizada essencialmente para sintetizar a voz do bot e tornar viável a participação de pessoas com alguma deficiencia visual).

```python
# método que utiliza o chatbot
    def mensagem(self, msg, *args):
        self.ids.box.add_widget(Message(text=msg))

        #sintetiza a voz e a reproduz
        resp = str(self.bot.get_response(msg))
        self.speak.say(resp)
        self.speak.runAndWait()
        self.ids.box.add_widget(Message(text=resp))
        self.ids.texto.text = ''

    def mensagem_audio(self, *args):
        with sr.Microphone() as fonte:
            frase = self.rec.listen(fonte)
            texto = self.rec.recognize_google(frase, language='pt')

        self.ids.box.add_widget(Message(text=texto))
        resp = str(self.bot.get_response(texto))
        self.speak.say(resp)
        self.speak.runAndWait()
        self.ids.box.add_widget(Message(text=resp))
        self.ids.texto.text = ''

    #implementa o esc para voltar
    def on_pre_enter(self):
        Window.bind(on_keyboard=self.back)
```

Figura 6. Iniciação dos métodos.

Esse trecho do código faz a iniciação dos métodos do Chatbot, bem como atribui as respostas fornecidas pelo nosso bot para a biblioteca de sintetização de voz. Nessa parte está contida a lógica principal do bot, onde são analisadas cada entrada de dados do usuário e é processada através de uma função, gerando uma resposta (saída) de acordo com o conteúdo fornecido.



```
Package            Version
------------------ -----------
ChatterBot         1.0.8
chatterbot-corpus  1.2.0
comtypes           1.1.11
en-core-web-sm     3.3.0
Jinja2             3.1.2
Kivy               2.1.0
numpy              1.22.4
preshed            3.0.6
PyAudio            0.2.11
pypiwin32          223
pyttsx3            2.90
pywin32            304
PyYAML             3.13
spacy              3.3.0
spacy-legacy       3.0.9
spacy-loggers      1.0.2
SpeechRecognition  3.8.1
SQLAlchemy         1.3.24
```

Figura 7. Bibliotecas.

Essa é a lista de bibliotecas principais, sendo os pilares fundamentais para o funcionamento do chatbot. Foi especificado a versão de cada biblioteca utilizada a fim de evitar conflitos caso a aplicação seja executada em outro ambiente de desenvolvimento.

```
<Manager>:
    MenuInitial:
        name: 'menuinitial'
    HomeScreen:
        name: 'homescreen'

<MenuInitial>:
    BoxLayout:
        orientation: 'vertical'
        padding: 150
        spacing: 25
        Image:
            source: 'images/chatbot.png'
            size_hint_y: None
            height: 220
            allow_stretch: True
        Button:
            text: 'Comece a Conversar!'
            on_release: app.root.current = 'homescreen'
        Button:
            text: 'Sair do Aplicativo'
            on_release: app.stop()

```

Figura 8. Front-end.

```
<HomeScreen>:
    BoxLayout:
        orientation: 'vertical'
        ActionBar:
            ActionView:
                ActionPrevious:
                    title: 'Chatbot'
                    on_release: app.root.current = 'menuinitial'
                ActionButton:
                    text: 'Sair'
                    on_release: app.stop()
        ScrollView:
            BoxLayout:
                id:box
                orientation:'vertical'
                size_hint_y:None
                height: self.minimum_height
        BoxLayout:
            size_hint_y: None
            height: 50
            TextInput:
                id: texto
            Button:
                text:'>'
                size_hint_x: None
                width: 50
                on_release: root.mensagem(texto.text)
            Button:
                text:'AUDIO'
                size_hint_x: None
                width: 50
                on_release: root.mensagem_audio(texto.text)

<Message>:
    size_hint_y: None
    height: 120
    Label:
        id:label
        font_size: 30
    Button:
        text: 'X'
        size_hint_x: None
        width: 50
        on_release: app.root.get_screen('homescreen').ids.box.remove_widget(root)
```

Figura 9. Layout.

Esses trechos de código realizam a estruturação do layout da aplicação. Sua principal função é estilizar e constituir a disposição dos elementos na tela, através de posicionamento dinâmico e ajuste de tabelas.

22. REFERÊNCIAS

https://www.deficienteonline.com.br/acessibilidade-para-deficientes-adaptacoes -e-normas___1.html

META. Meta for Developers, 2022. Introdução. Disponível em: https://developers.facebook.com/docs/messenger-platform/introduction/. Acesso em: 25 de mar. De 2022.

REJANE, Kely e JUNIOR, Ciro. Chatbot: uma visão geral sobre aplicações inteligentes, Revista Sítio Novo Instituto Federal de Tocantis. Disponível em: https://sitionovo.ifto.edu.br/index.php/sitionovo/article/view/140. Acesso em: 24 de mar. 2022.

PEREIRA, Thiago. Desenvolvimento de um sistema do tipo Chatbot para o curso de Sistemas de Informação. Repositório Institucional UFSC, 2021-05-06. Seção 2.2 Processamento da Linguagem Natural (PLN), pagina 21). Disponível em: https://repositorio.ufsc.br/handle/123456789/223935. Acesso em: 25 de mar. de 2022.

SHIRAISHI, Guilherme de Farias; YODA, Fernanda Sayuri; LOURENÇO, Valter Cavalcante. Para o high-tech ser high touch: um estudo exploratório com chatbots. Revista Administração em Diálogo - RAD, São Paulo, v. 20, n. ja/abr. 2020, 2020. Disponível em: https://revistas.pucsp.br/index.php/rad/article/view/40774/31505 DOI: 10.23925/2178-0080.2020v22i1.40774. Acesso em: 25 de mar. de 2022.

SILVA, Rafael João da. Batendo um papo com a informação: o uso dos chatbots para a recuperação da informação e a contribuição da Ciência da Informação nesse processo. 2020. Universidade de São Paulo, São Paulo, 2020. Disponível em: https://www.teses.usp.br/teses/disponiveis/27/27163/tde-10032021-013140/. Acesso em: 25 de mar. de 2022.

LABAKY, Josué. Introdução a Python. 2003. UNESP, Campus de Ilha Solteira. Disponível em: https://dcc.ufrj.br/~fabiom/python/pythonbasico.pdf. Acesso em: 25 de mar. de 2022.

BORGES, Luiz Eduardo. Python para Desenvolvedores: Aborda Python 3.3. 2014. Disponível em https://books.google.com.br/books?hl=pt-BR&lr=&id=eZmtBAAAQBAJ&oi=fnd&pg=PA14&dq=cole%C3%A7%C3%B5es+python&ots=VERrwnHfdp&sig=62DXXaVEpV2_0rzYzEBQKLVR5DQ#v=onepage&q=cole%C3%A7%C3%B5es%20python&f=false. Acesso em: 25 de mar. de 2022.