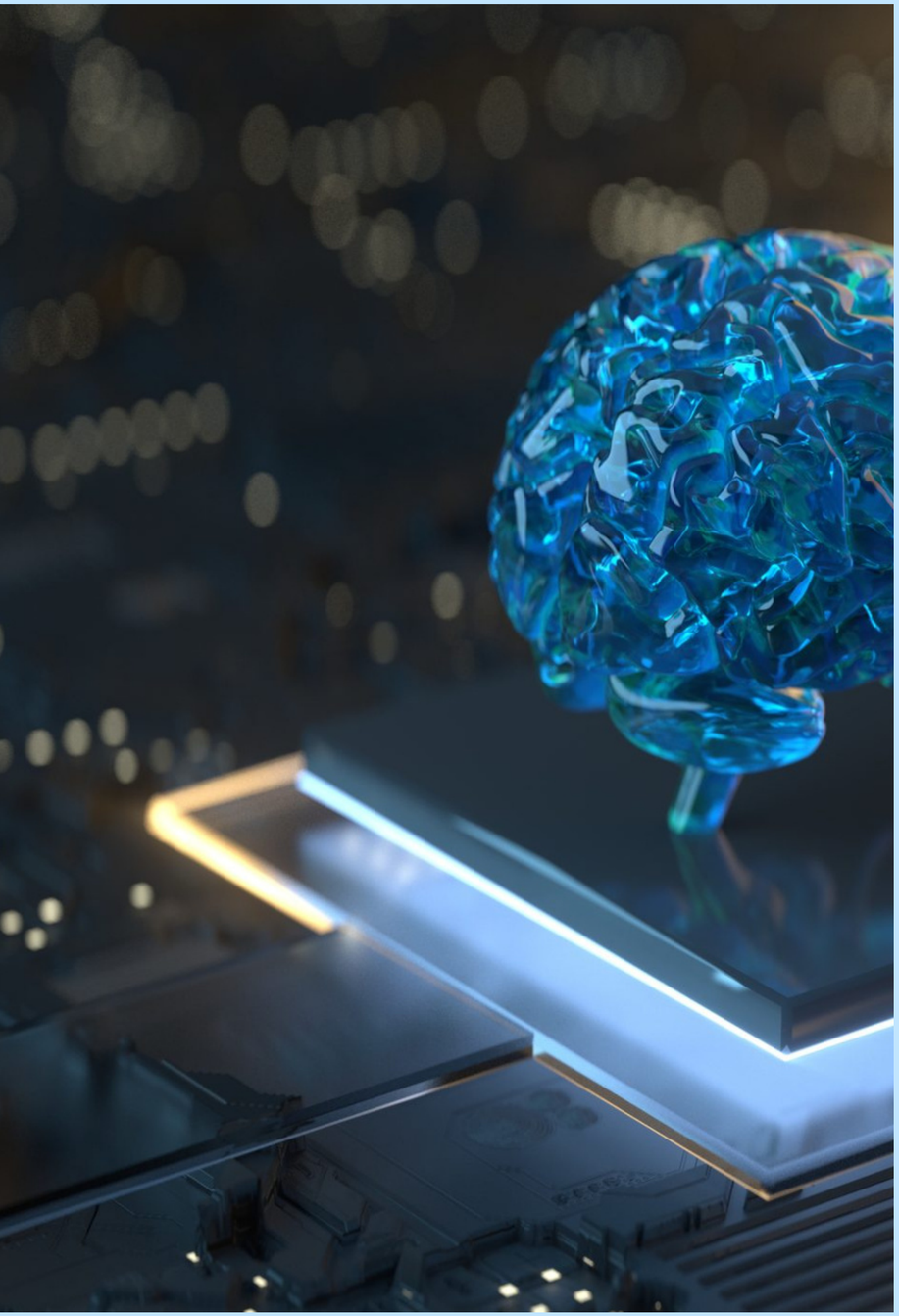
HEALTHCARE CHATBOT

AVEC NLP

EnduraBot

Réalisée par :
Aya



PLAN DE PRÉSENTATION

1. **INTRODUCTION :**
   - C'EST QUOI UN CHATBOT ?
   - CHATBOT EN SANTÉ
2. **PRÉSENTATION DE ENDURABOT**
3. **ANALYSE ET CONCEPTION**
4. **RÉALISATION**
5. **DEMO**
6. **CONCLUSION**

UN CHATBOT, C'EST QUOI ?

Un **chatbot** est un programme informatique qui simule une conversation humaine via des commandes vocales ou des chats textuels ou les deux.

Chatbot, abréviation de chatterbot, est une fonctionnalité d'intelligence artificielle (IA) qui peut être intégrée et utilisée dans toutes les principales applications de messagerie. Il a été initialement créée dans un cadre “médical”, ELIZA (1966) fut le premier chatterbot : une “ thérapeute virtuelle” était née .

LES CHATBOTS EN SANTÉ

Les chatbots de santé alimentés par l'IA sont capables de traiter des demandes simples avec facilité et offrent aux utilisateurs un moyen pratique de rechercher des informations.

Avantages des chatbots dans le domaine de la santé

1

Disponibilité 24 heures sur 24 et 7 jours sur 7

2

Réduction des temps d'attente

3

Réduction des coûts des soins

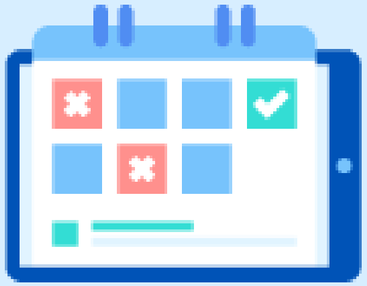
4

Amélioration de la satisfaction des clients

5 USE CASES OF CHATBOTS IN HEALTHCARE

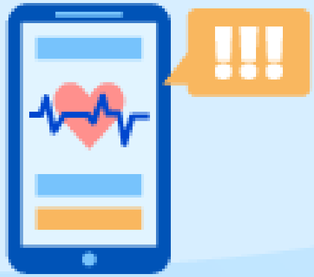
Providing informational support

Assistance in refilling



Scheduling appointments

Collecting patient information



Providing medical assistance

PRÉSENTATION DE ENDURABOT

EnduraBot est un nom qui combine « endurance » et « bot », suggérant un chatbot qui peut fournir un soutien et une assistance durables dans le **secteur de la santé**.

Le nom implique un sentiment de fiabilité, de durabilité et de résilience, qui sont toutes des qualités souhaitables pour **un chatbot** de soins de santé.

FONCTIONNALITÉS

01

Compréhension des requêtes et détection des symptômes

EnduraBot peut détecter les symptômes de la personne à partir des réponses qu'elle fournit.

02

Réponses médicales précises et personnalisées (Conseils d'auto-soins)

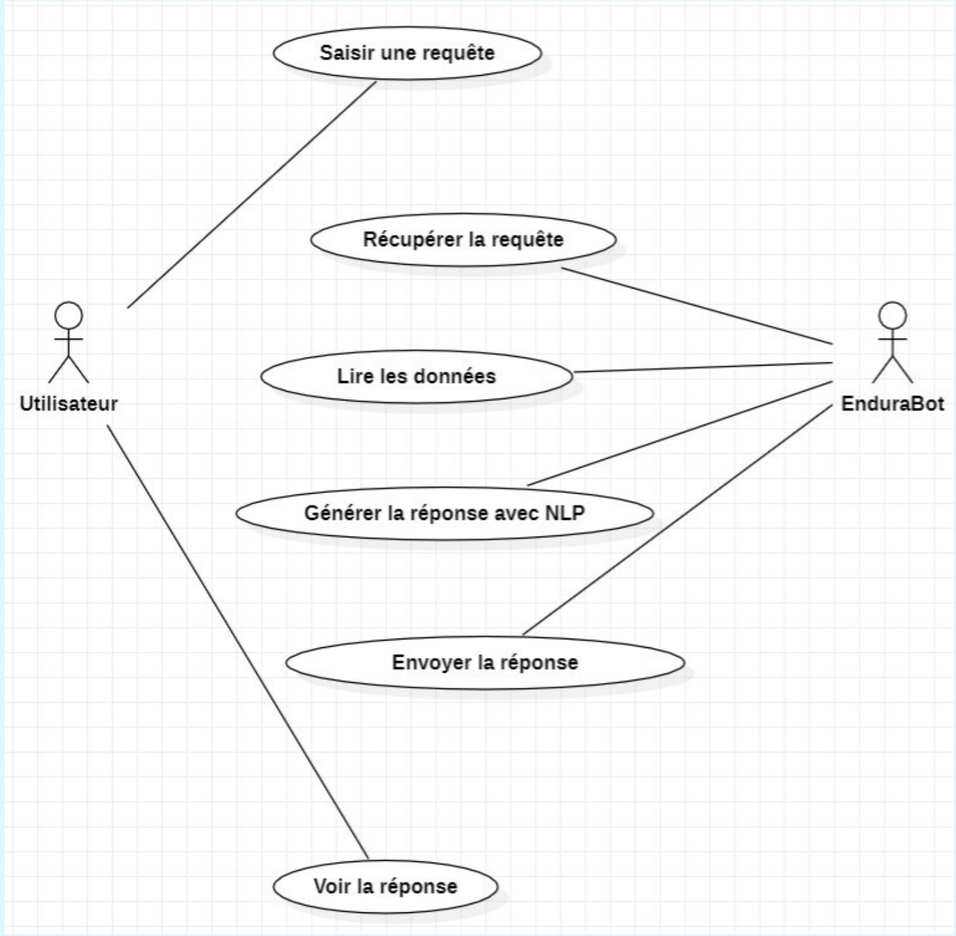
EnduraBot peut fournir des conseils pour le soulagement de symptômes et recommander des médicaments, des remèdes simples et des mesures de prévention.

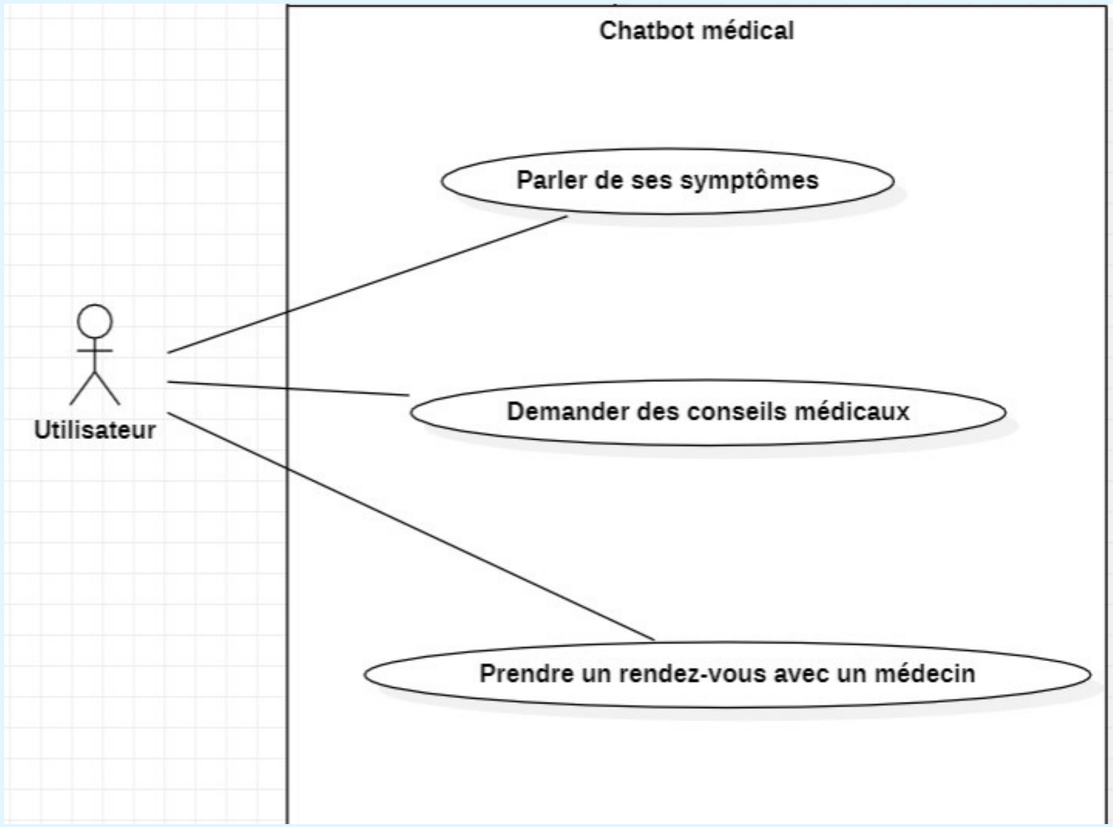
03

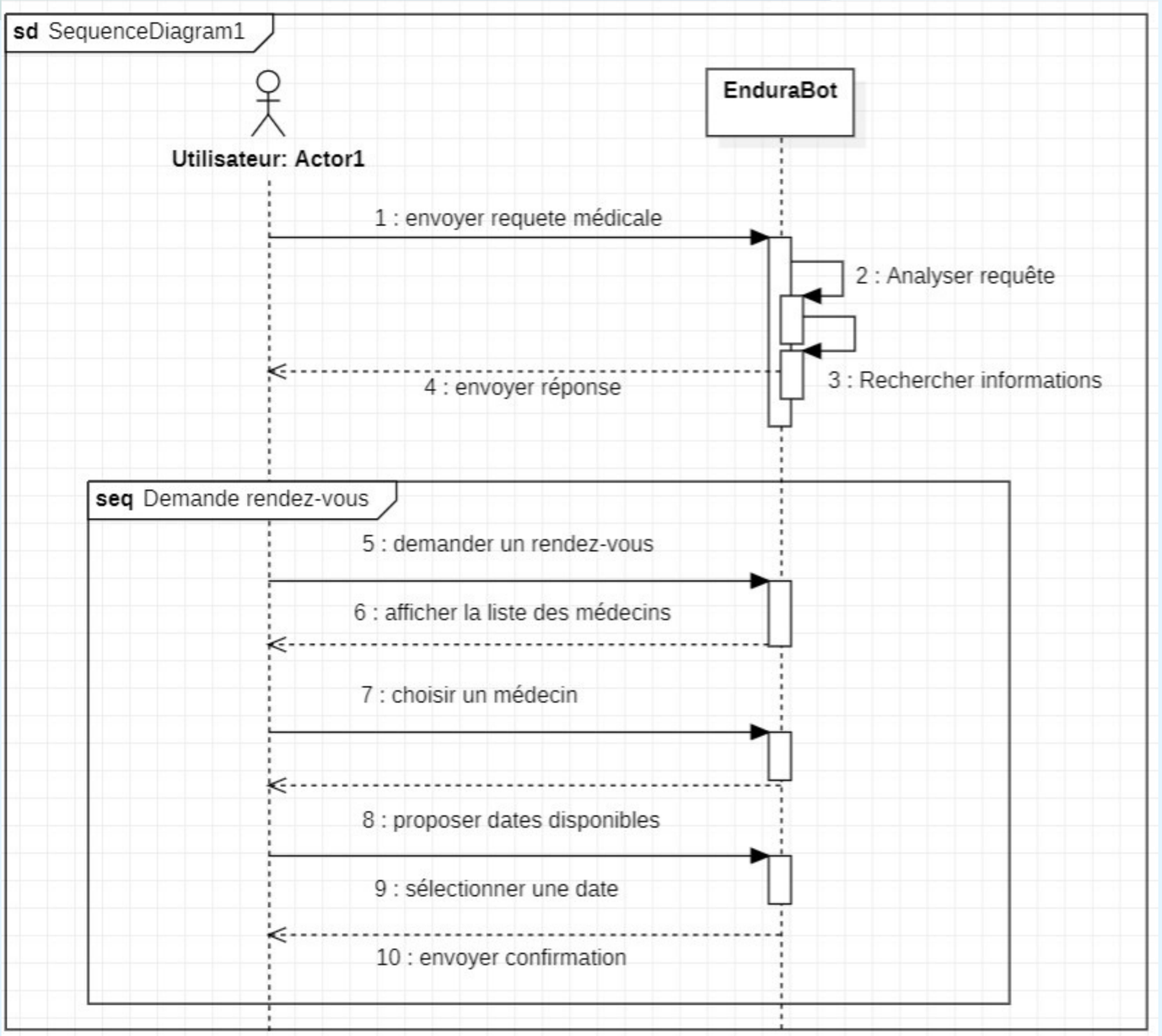
Prise des rendez-vous médicaux

EnduraBot permet aux utilisateurs de planifier facilement des rendez-vous avec des professionnels de santé, tels que des médecins.

ANALYSE ET CONCEPTION






sd SequenceDiagram1
Utilisateur: Actor1
EnduraBot
1 : envoyer requete médicale
2 : Analyser requête
3 : Rechercher informations
4 : envoyer réponse
seq Demande rendez-vous
5 : demander un rendez-vous
6 : afficher la liste des médecins
7 : choisir un médecin
8 : proposer dates disponibles
9 : sélectionner une date
10 : envoyer confirmation

RÉALISATION

TECHNOLOGIES & OUTILS

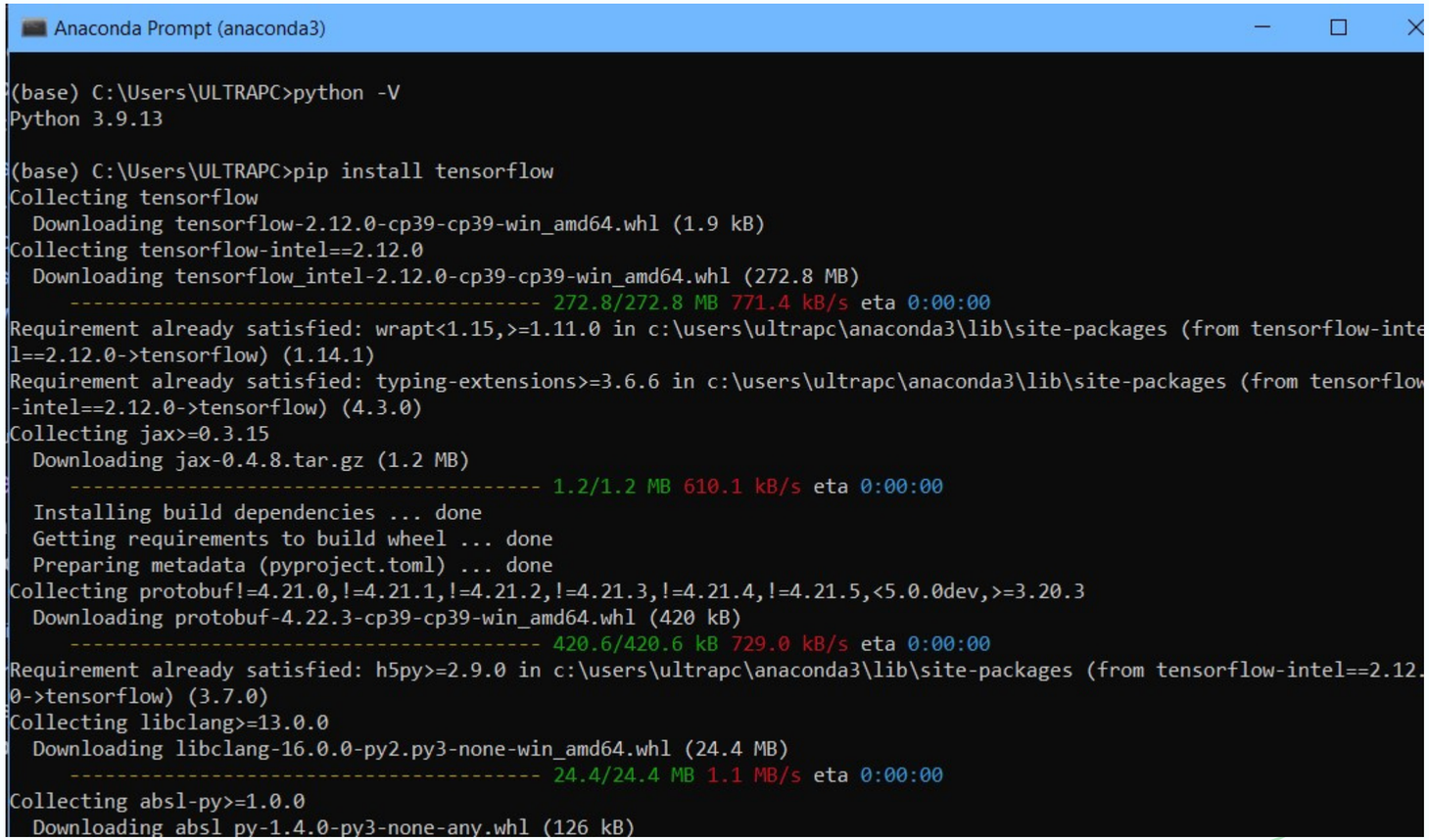
Programming language

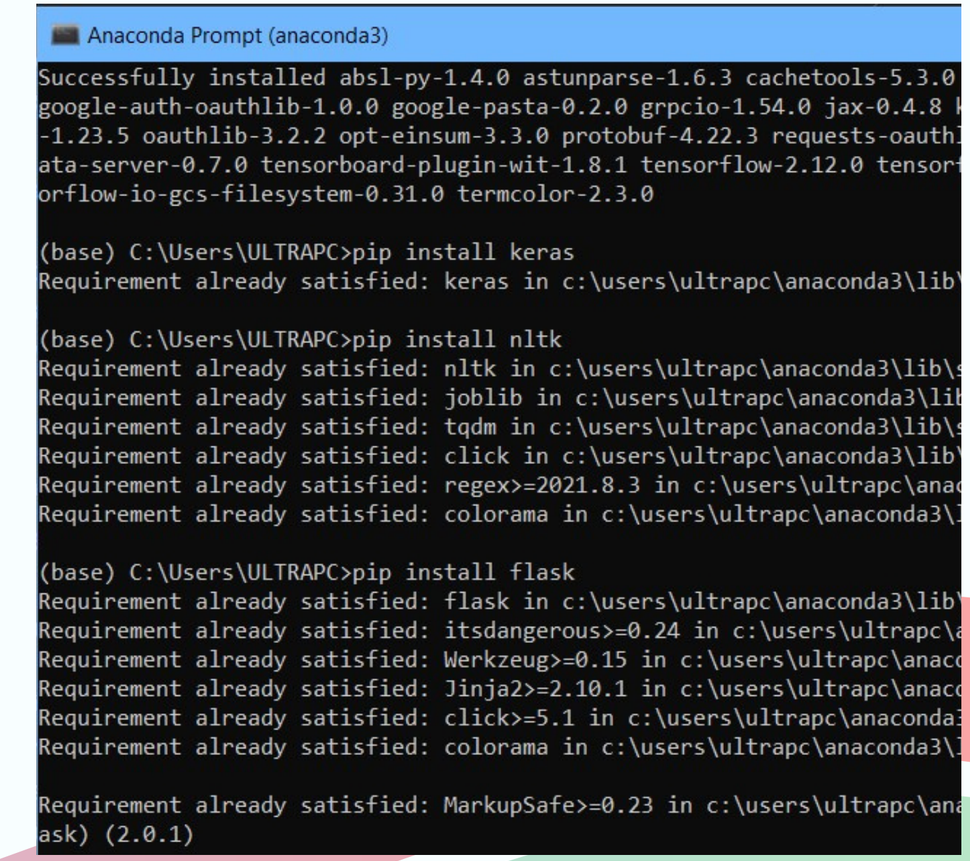
IDE

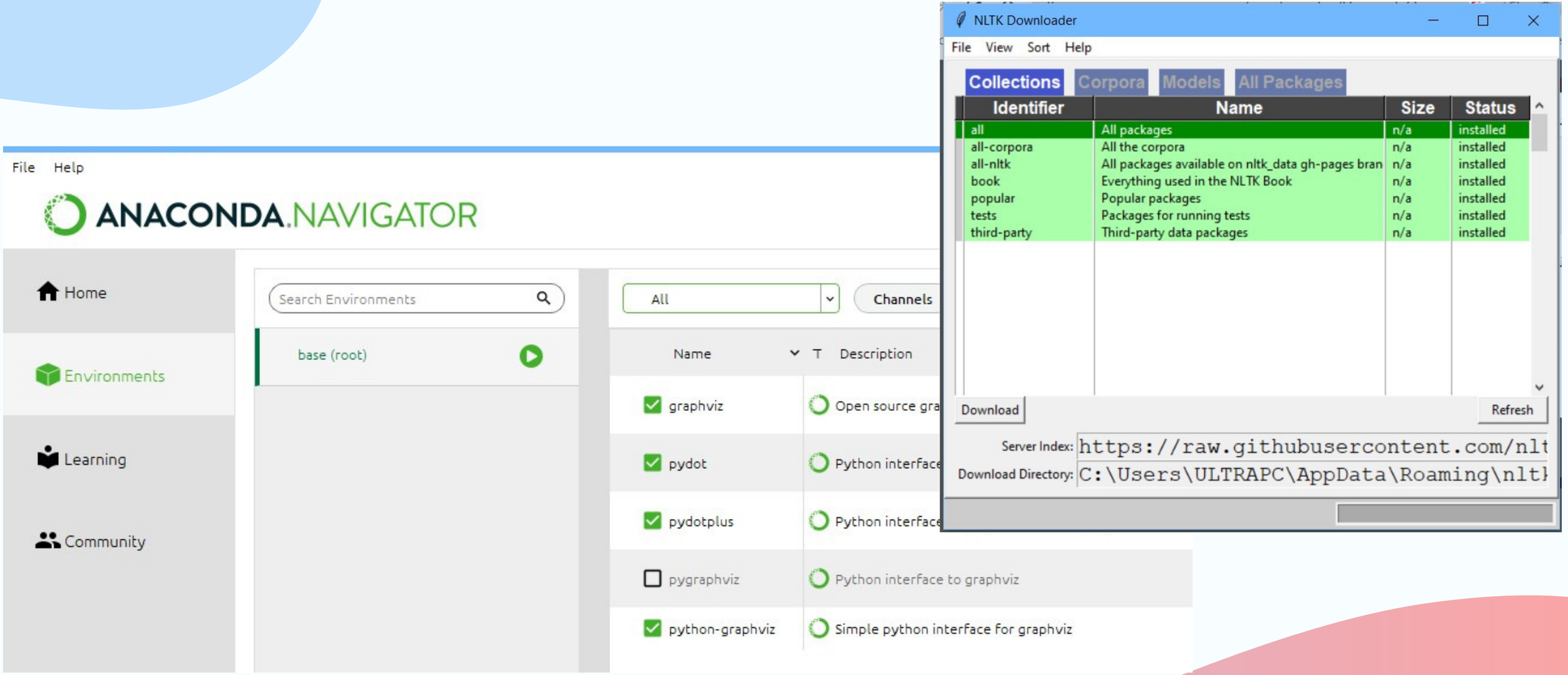
ML & NN Libraries

Web application framework

Installation de : tensorflow, keras, pickle, flask






NLTK Downloader
File View Sort Help
Collections Corpora Models All Packages
Identifier Name Size Status
all All packages n/a installed
all-corpora All the corpora n/a installed
all-nltk All packages available on nltk_data gh-pages bran n/a installed
book Everything used in the NLTK Book n/a installed
popular Popular packages n/a installed
tests Packages for running tests n/a installed
third-party Third-party data packages n/a installed
Download
Refresh
Server Index: https://raw.githubusercontent.com/nlt
Download Directory: C:\Users\ULTRAPC\AppData\Roaming\nltk
File Help
ANACONDA.NAVIGATOR
Home
Environments
Learning
Community
Search Environments
base (root)
All
Channels
Name T Description
graphviz Open source gra
pydot Python interface
pydotplus Python interface
pygraphviz Python interface to graphviz
python-graphviz Simple python interface for graphviz

STRUCTURE DU PROJET

- static
- templates
- data.json
- endu_model.h5
- endura.py
- labels.pkl
- model_plot.png
- texts.pkl
- training.py


styles


bg.jpeg


endura.png


person-icon.png

Fichier de données data.json


```
{"intents":
    [
            {
             "tag":"greetings",
             "patterns": ["hello","hey","hi","good day","greetings","what's up?","how is it going"],
             "responses":["Hello","Hey There! What can i do for you today?","How has been your health lately?"]
            },
            {
             "tag":"goodbye",
             "patterns": ["goodbye","see you later","have a good day","bye","see ya"],
             "responses":["Have a nice day","Goodbye","Bye, take care."]
            },
            {
             "tag":"age",
             "patterns": ["How old are you?","What is your age","how old are you","age?"],
             "responses":["I get reborn after every compilation","My creator is almost 22 years!"]
            },
            {
             "tag":"name",
             "patterns": ["what is your name","what should i call you","what's your name?","who are you?","can you tell me your name"],
             "responses":["You can call me EnduraBot!","My name is Endura!","I am EnduraBot, your medical assistant :)"]
            },
            {
             "tag":"common cold symptoms",
             "patterns": ["Runny or stuffy nose",
             "Sore throat",
             "Cough",
             "Congestion",
             "Slight body aches or a mild headache",
             "Sneezing",
             "Low-grade fever",
             "Generally feeling unwell (malaise)"],
             "responses":["It seems that you are suffering from common cold"]
            },
            {
```

training.py

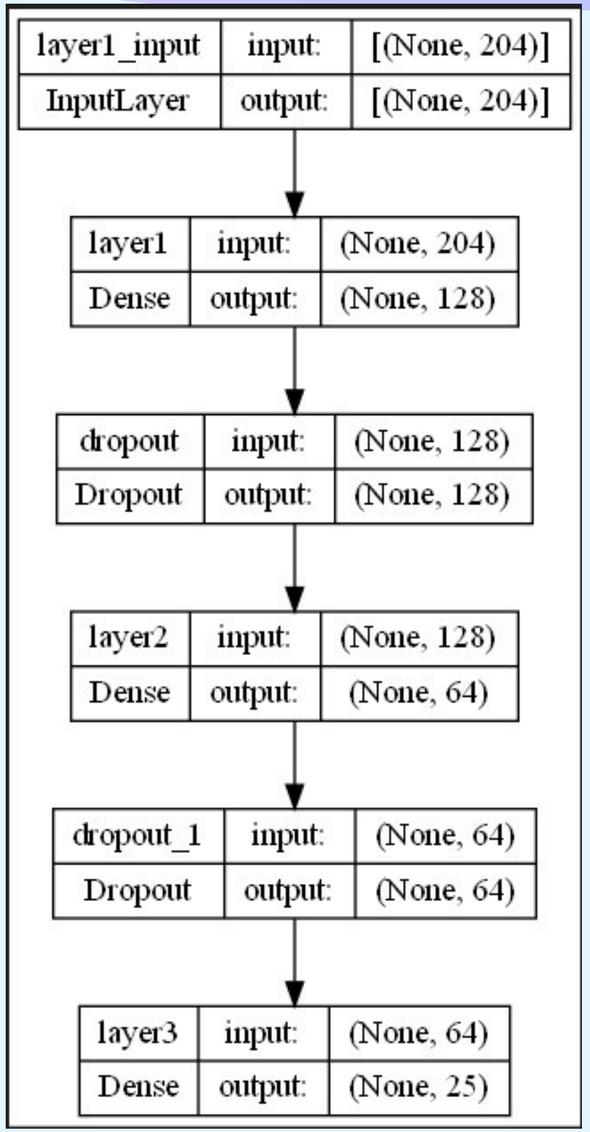
```python
import json

import nltk
import pickle
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

import numpy as np
import pandas as pd
import random
from keras.utils.vis_utils import plot_model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers.legacy import SGD


with open('data.json', 'r') as file:
    data_file = file.read()
    intents = json.loads(data_file)

words=[]
classes = []
documents = []
ignore_words = ['?', '!', ',', '.']
for intent in intents['intents']:
  for pattern in intent['patterns']:
    #tokenize each word
    w = nltk.word_tokenize(pattern)
    #add to our words
    words.extend(w)
    #add to documents
    documents.append((w,intent['tag']))
    #add to our classes list
    if intent['tag'] not in classes:
      classes.append(intent['tag'])

words =[lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]
words = sorted(set(words))
classes=sorted(set(classes))
# documents = combination between patterns and intents
print (len(documents), "documents")
# classes = intents
```

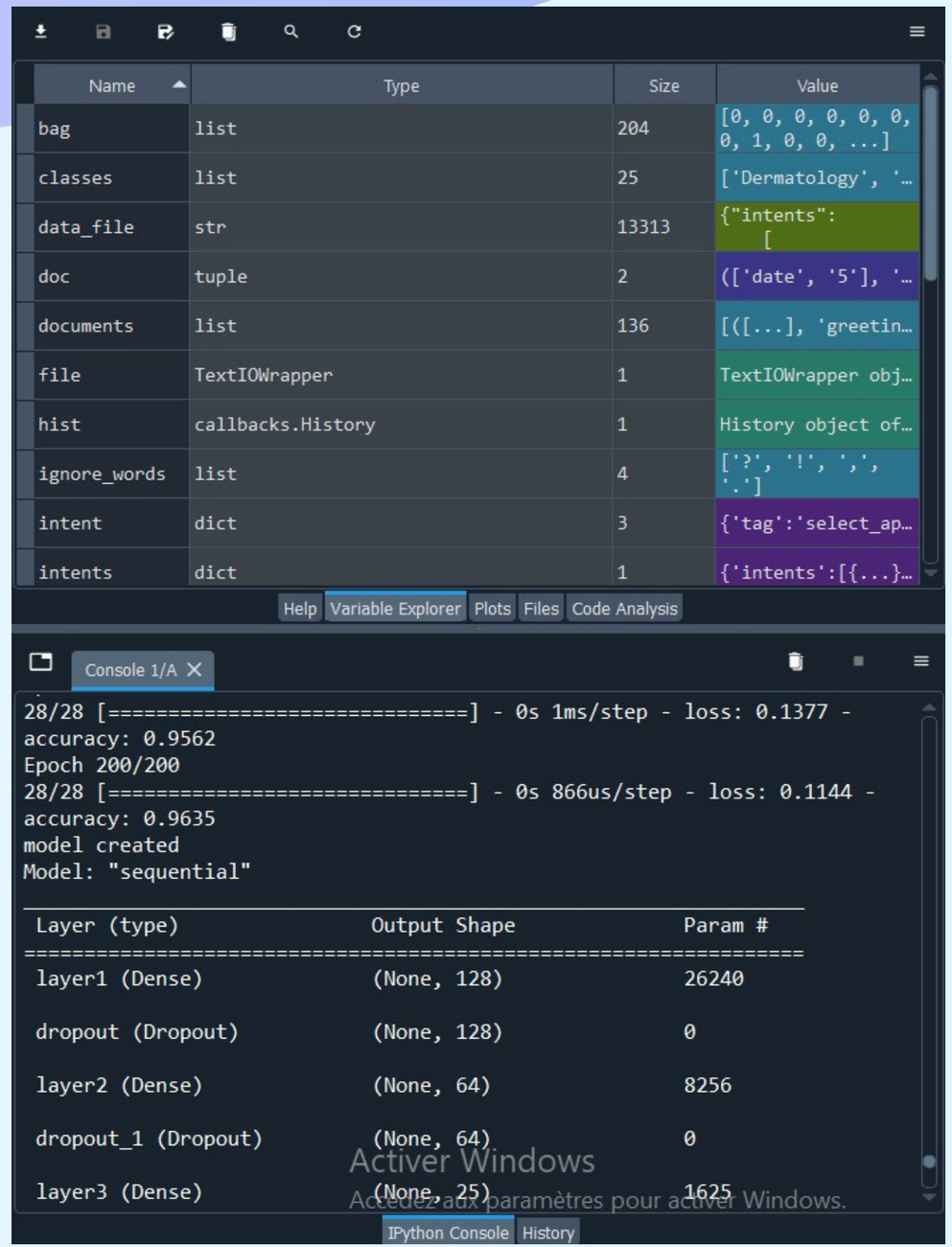
```python
print (len(words), "unique lemmatized words", words)
pickle.dump(words,open('texts.pkl','wb'))
pickle.dump(classes,open('labels.pkl','wb'))
#create my training data
training=[]
#create an empty array for the output
output_empty=[0]*len(classes)

#training set : bag of words for each sentence
for doc in documents:
  bag=[]
  #list of tokenized words for the pattern
  pattern_words = doc[0]
  #lemmatizing each word
  pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]
  #create my bag of words array with 1 if word match found in current position
  for word in words:
    bag.append(1) if word in pattern_words else bag.append(0)

  #output is 0 for each tag and 1 for current tag (for each pattern)
  output_row=list(output_empty)
  output_row[classes.index(doc[1])]=1

  training.append([bag,output_row])



training=np.array(training)
training = training.tolist()
training.append([bag, output_row])

# shuffle the features and turn them into numpy array
random.shuffle(training)
training = np.array(training)

# create train and test lists. X - patterns, Y - intents
train_x = list(training[:, 0])
train_y = list(training[:, 1])

print("Training data created")
```

training.py

```
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu', name="layer1"))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu', name="layer2"))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax', name="layer3"))

# Compile model. Stochastic gradient descent with Nesterov accelerated gradient gives good results for this model
sgd = SGD(lr=0.01, momentum=0.9, decay=1e-6, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

# fitting and saving the model
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1)

model.save('endu_model.h5', hist)
print("model created")
print(model.summary())
plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```

model_plot.png





endura.py

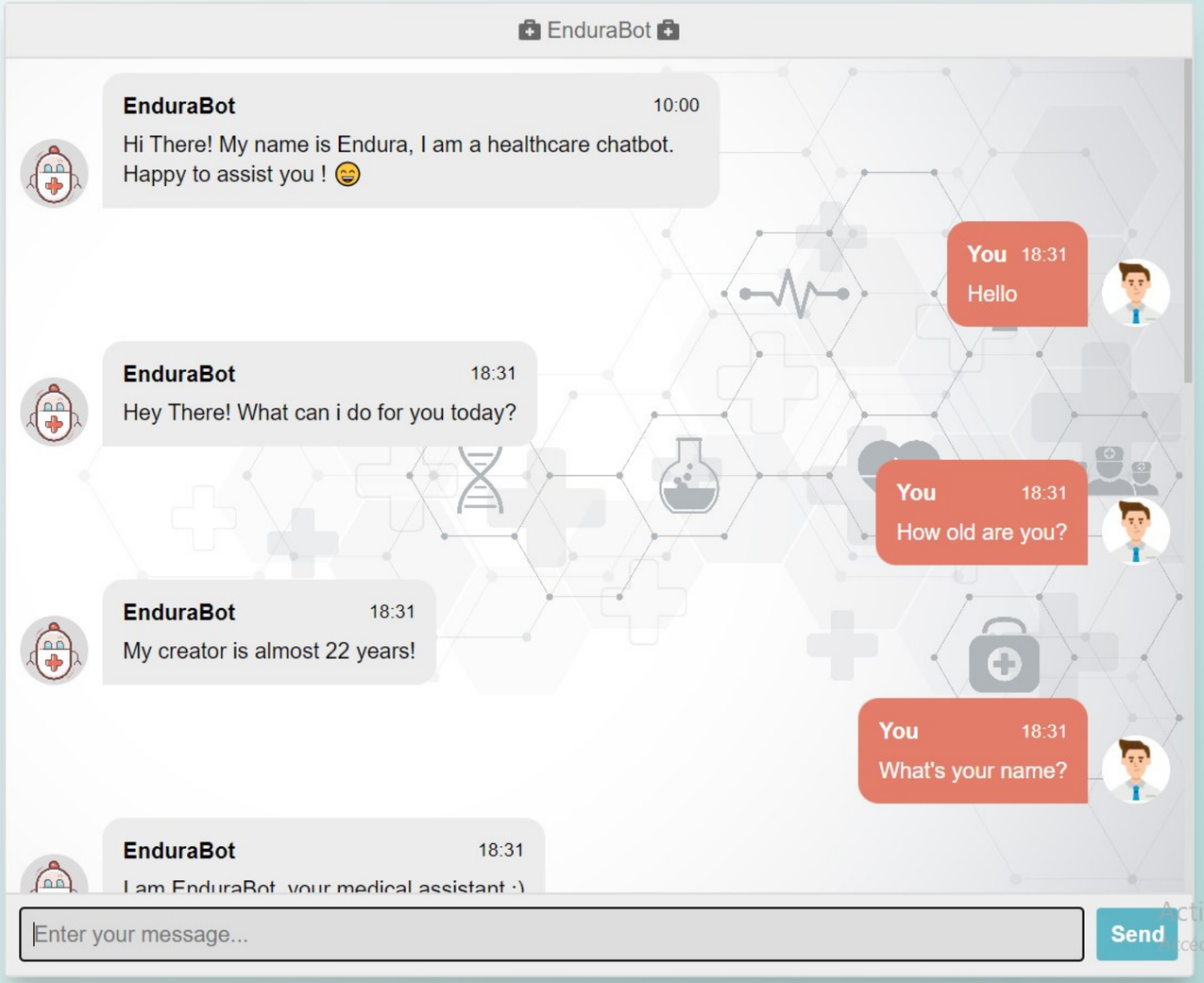
```python
model = load_model('endu_model.h5')

intents = json.loads(open('data.json').read())

words = pickle.load(open('texts.pkl','rb'))
classes = pickle.load(open('labels.pkl','rb'))

def clean_up_sentence(sentence):
    #tokenize the pattern
    sentence_words = nltk.word_tokenize(sentence)
    #lemmatize each word
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_wor
    return sentence_words

# return bag of words array: 0 or 1 for each word in the bag that exists in the s
def bow(sentence, words, show_details=True):
    # tokenize the pattern
    sentence_words = clean_up_sentence(sentence)
    # bag of words - matrix of N words, vocabulary matrix
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                # assign 1 if current word is in the vocabulary position
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)
    return np.array(bag)

def predict_class(sentence, model):
    #filter out predictions below a threshold
    b = bow(sentence, words,show_details=False)
    res = model.predict(np.array([b]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
    #sort by strength of probability
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append({"intent": classes[r[0]], "probability": str(r[1])})
    return return_list
```

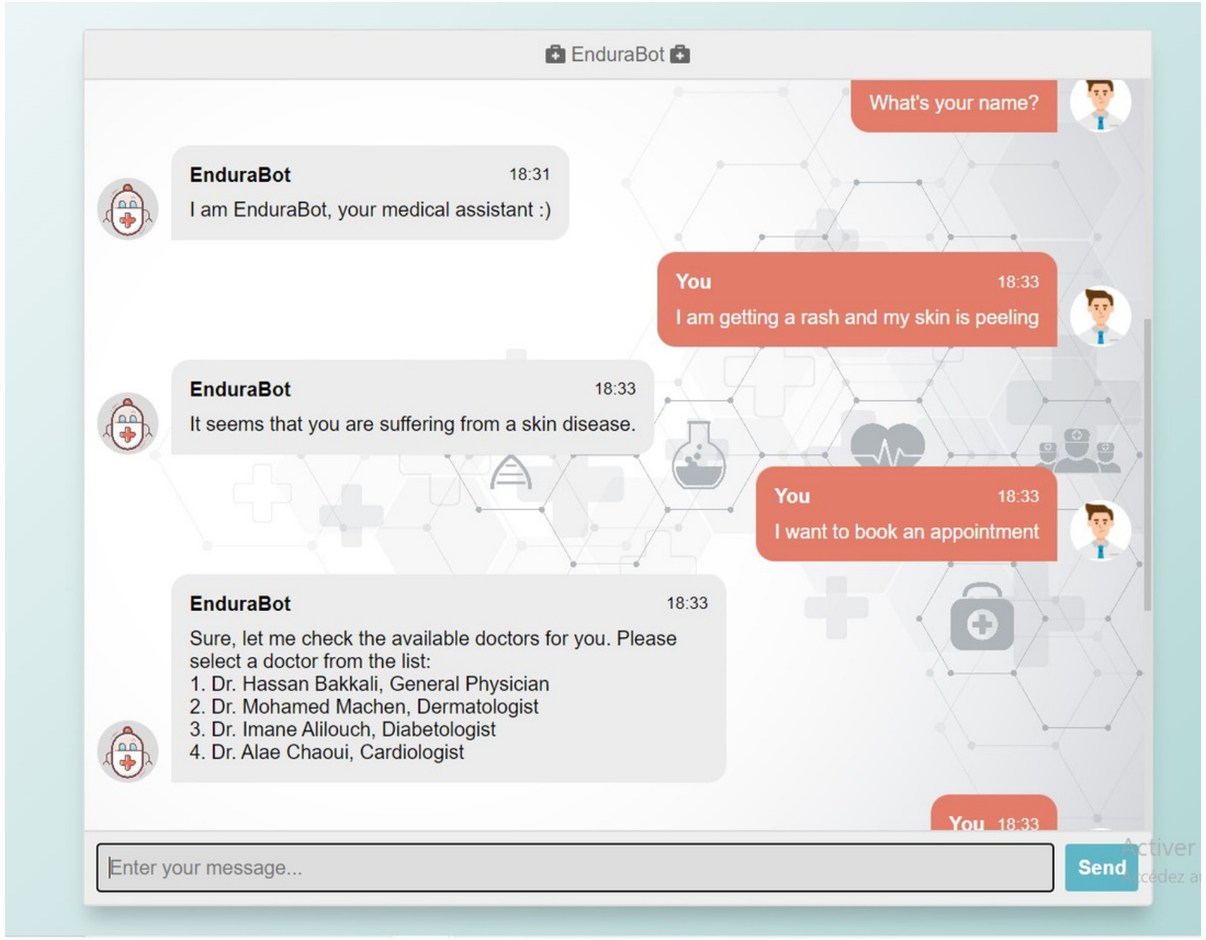
```python
    {"name": "Dr. Alae Chaoui", "specialty": "Cardiologist", "available_dates": ["2023-06-8 9:00", "2023-06-10 11:00",
]

# Global variables to store user state
user_state = {}

def handle_appointment_intent():
    response = "Sure, let me check the available doctors for you. Please select a doctor from the list:<br>"

    for i, doctor in enumerate(doctors):
        response += f"{i+1}. {doctor['name']}, {doctor['specialty']}<br>"

    user_state['current_intent'] = 'select_doctor'
    return response

def handle_select_doctor_intent(user_input):
    try:
        doctor_index = int(user_input) - 1
        if doctor_index < 0 or doctor_index >= len(doctors):
            return "Invalid doctor selection. Please try again."

        selected_doctor = doctors[doctor_index]
        user_state['selected_doctor'] = selected_doctor

        available_dates = selected_doctor['available_dates']
        response = "Great! Please select a date for your appointment from the available dates:<br>"
        for i, date in enumerate(available_dates):
            response += f"{i+1}. {date}<br>"

        user_state['current_intent'] = 'select_appointment_date'
        return response

    except ValueError:
        return "Invalid input. Please enter a valid number."


def handle_select_appointment_date_intent(user_input):
    try:
        selected_doctor = user_state.get('selected_doctor')

        if selected_doctor is None:
            return "No doctor selected. Please start over and select a doctor first."
```

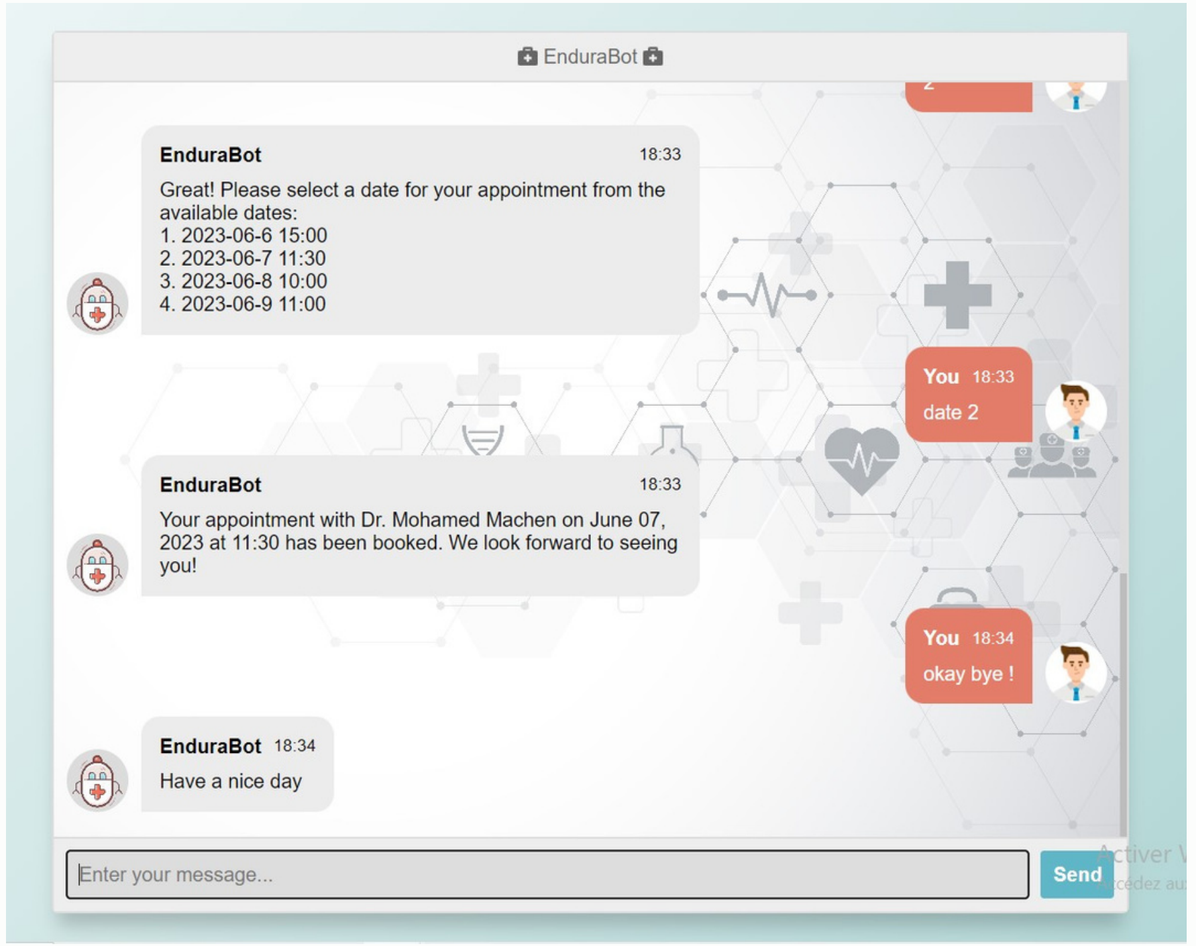
```python
def handle_select_appointment_date_intent(user_input):
    try:
        selected_doctor = user_state.get('selected_doctor')

        if selected_doctor is None:
            return "No doctor selected. Please start over and select a doctor first."

        available_dates = selected_doctor['available_dates']

        # Extract the number from the user input pattern (e.g., "date 1" -> 1)
        date_index = int(user_input.split()[1])

        if date_index < 1 or date_index > len(available_dates):
            return "Invalid date selection. Please try again."

        selected_date = available_dates[date_index - 1]
        appointment_date = datetime.datetime.strptime(selected_date, "%Y-%m-%d %H:%M").strftime("%B %d, %Y at %H:%M")

        response = f"Your appointment with {selected_doctor['name']} on {appointment_date} has been booked. We look forw
        return response

    except ValueError:
        return "Invalid input. Please enter a valid selection."


def chatbot_response(msg):
    ints = predict_class(msg, model)
    intent = ints[0]['intent']

    print("Detected intent:", intent)  # Print the detected intent for debugging purposes

    if intent == 'appointment':
        response = handle_appointment_intent()
        print("Response from handle_appointment_intent:", response)  # Print the response for debugging purposes
        if response:
            return response

    elif intent == 'select_doctor':
        response = handle_select_doctor_intent(msg)
        print("Response from handle_select_doctor_intent:", response)  # Print the response for debugging purposes
        if response:
            return response
```

```python
app = Flask(__name__)

app.static_folder = 'static'

@app.route("/")
def home():
    return render_template("index.html")
@app.route("/get")

def get_bot_response():
    userText = request.args.get('msg')
    return chatbot_response(userText)
if __name__ == "__main__":
    app.run()
```

INTERFACE




EnduraBot
What's your name?
EnduraBot 18:31
I am EnduraBot, your medical assistant :)
You 18:33
I am getting a rash and my skin is peeling
EnduraBot 18:33
It seems that you are suffering from a skin disease.
You 18:33
I want to book an appointment
EnduraBot 18:33
Sure, let me check the available doctors for you. Please select a doctor from the list:
1. Dr. Hassan Bakkali, General Physician
2. Dr. Mohamed Machen, Dermatologist
3. Dr. Imane Alilouch, Diabetologist
4. Dr. Alae Chaoui, Cardiologist
You 18:33
Enter your message...
Send


EnduraBot
EnduraBot 18:33
Great! Please select a date for your appointment from the available dates:
1. 2023-06-6 15:00
2. 2023-06-7 11:30
3. 2023-06-8 10:00
4. 2023-06-9 11:00
You 18:33
date 2
EnduraBot 18:33
Your appointment with Dr. Mohamed Machen on June 07, 2023 at 11:30 has been booked. We look forward to seeing you!
You 18:34
okay bye !
EnduraBot 18:34
Have a nice day
Enter your message...
Send

DEMO

CONCLUSION

PROJET INTÉRESSANT

Appliquer les notions et les techniques de base du NLP
Utilisation des frameworks puissants

AMÉLIORATIONS

Ce projet peut être amélioré en intégrant une base de connaissances plus étendue et peut être en utilisant un modèle de NLP plus complexe.

MERCI DE VOTRE ATTENTION

ENDURABOT