# sleep&go

## Роботизированное одеяло для прекрасного сна и отличного настроения

Описание проекта

**Исследования**

В начале нашей работы мы провели социологическое исследование путем Онлайн-опроса друзей и родственников.

Респондентам задавались следующие вопросы:

1. Возраст?
2. Может ли робот быть твоим другом?
3. Что твой робот-друг может для тебя делать?

Результаты исследования:


Всего в опросе приняло 60 человек. Основной возраст респондентов сосредоточен в области от 10 до 23 лет.

41 человек ответил положительно на вопрос «Может ли робот быть твоим другом?»


Основные положительные ответы на вопрос «Что твой робот-друг может для тебя делать?» можно разделить на следующие группы:

1. Робот-друг должен со мной общаться и оказывать психологическую поддержку
2. Робот-друг должен убираться или поддерживать порядок в комнате

**Идея**

В рамках проведенного социологического исследования одним из интересных ответов был «Робот-друг должен быть одеялом». Эта идея робота-одеяла легла в основу нашего стартапа «Sleep & Go».

Сон представляет собой пассивный отдых организма и является важной составляющей здоровья человека. Здоровый сон считается залогом успеха в карьере, учебе и в жизни.

Сон способствует переработке и хранению информации, облегчает закрепление изученного материала и реализует подсознательные модели ожидаемых событий.

Кроме того, во время сна вырабатывается ряд важных гормонов, идёт регенерация тканей, восполняются физические силы.

**1/3 часть жизни человек проводит в кровати… с одеялом!**

Основные причины нарушения сна:

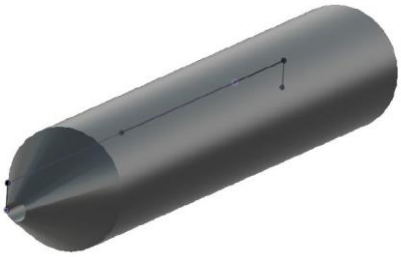
- стресс
- физическое состояние
- внешние раздражители (свет, шум)
- неудобное место для сна

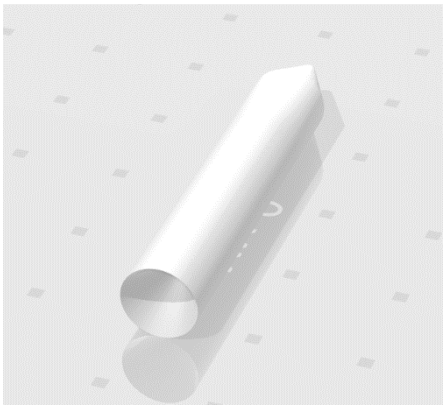
Мы проанализировали потребности человека перед сном и в процессе сна. В результате нами были определены следующие функциональные задачи:

1. Объект разработки должен представлять собой роботизированное одеяло
2. Робот-одеяло должен иметь возможность интерактивного общения с пользователем путем голосовой и видео коммуникации
3. Робот-одеяло должен иметь возможность лечебного физического воздействия на организм (массаж, разогрев и т.д.)
4. Робот-одеяло должен собирать статистику физических параметров пользователя и предоставлять интерактивные отчеты
5. Робот-одеяло должен уметь автоматически расправляться и принимать аккуратный вид

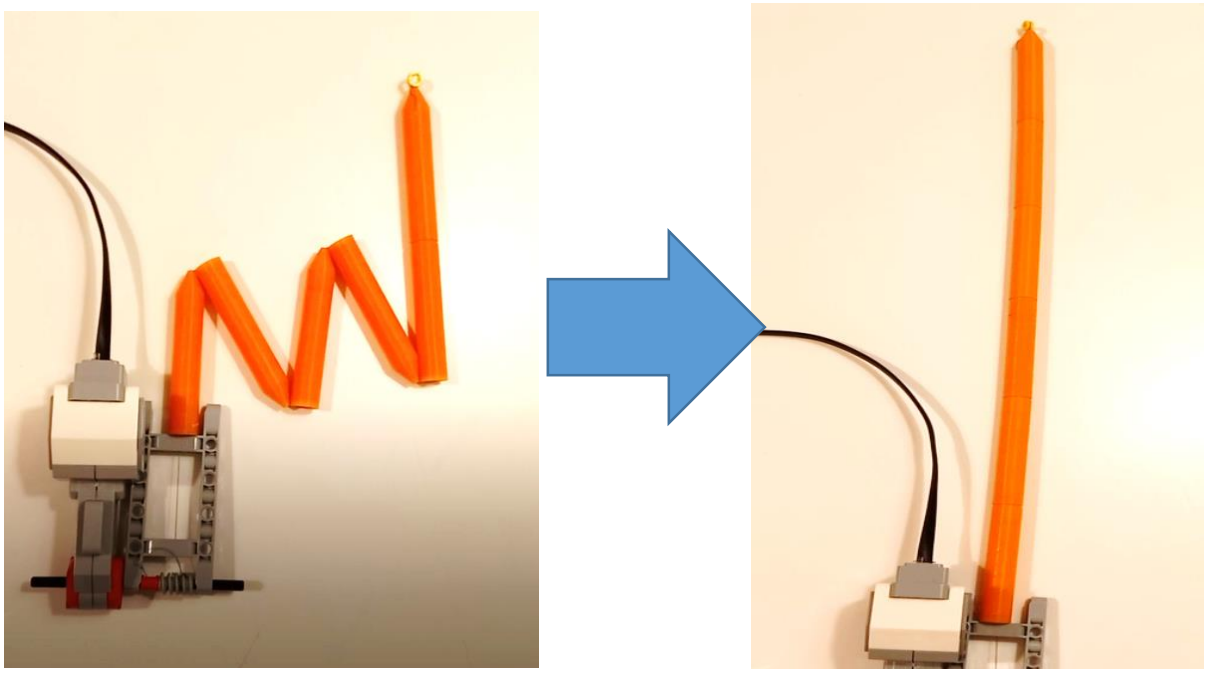
**Роботизированное решение**

В первую очередь мы озадачились каким образом одеяло может расправляться и делать массаж. В процессе долгих экспериментов мы остановились на каркасе состоящим из конических элементов.





Внутри элемента сделано отверстие, в которое протягивается шнур и крепится одним концом к двигателю, а другой конец фиксируется на вершине конечного элемента. За счет натяжения шнура конус входит в конический паз, и конструкция выпрямляется.



При ослаблении натяжения шнура каркас расслабляется. В результате чего одеяло становится снова мягким и может свободно деформироваться.

## Распознавание эмоций

С 2018 года в своих проектах по робототехнике (Vegetarium WRO 2019, Sinhronium WRO 2020, WindHunter PPO 2021) мы использовали машинное зрение на базе библиотеки Python Open-CV.

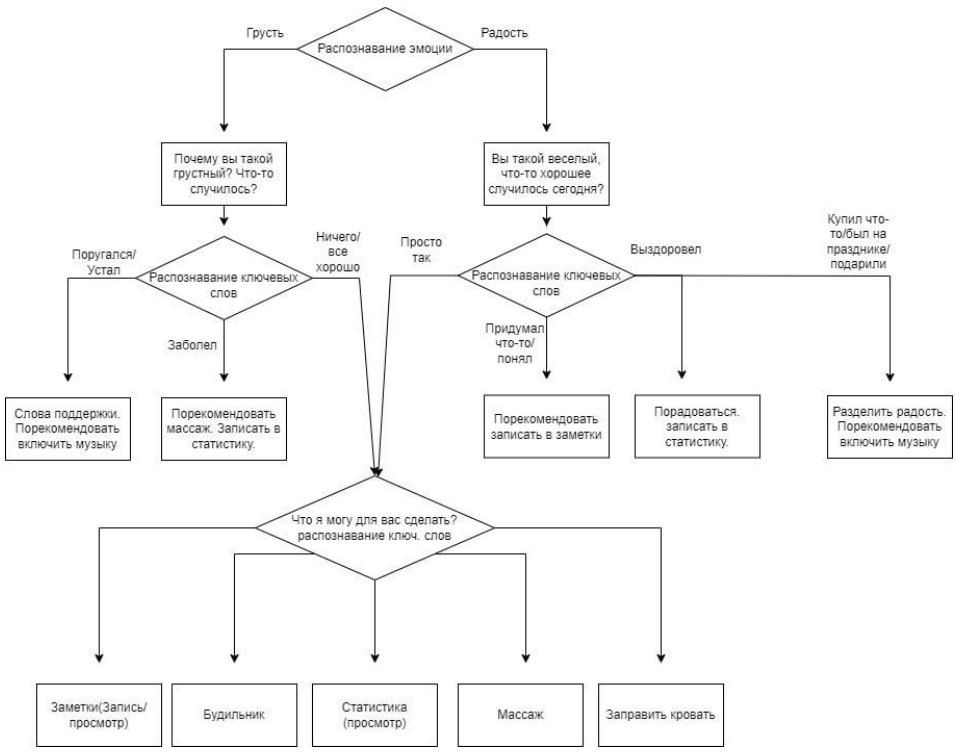
Для реализации процесса идентификации эмоций в настоящем проекте мы так же использовали библиотеку Open-CV и применили инструменты для обучения нейронных сетей - популярного фреймворка Tensorflow и библиотеку глубокого обучения Keras. В качестве основной сборки нейронной сети применялась свободно распространяемая база данных FER2013.

## Распознавание и воспроизведение речи

Для реализации речевой коммуникации с роботом-одеялом мы изучили опыт известных систем голосовых помощников от Apple Siri и Яндекс Алиса.

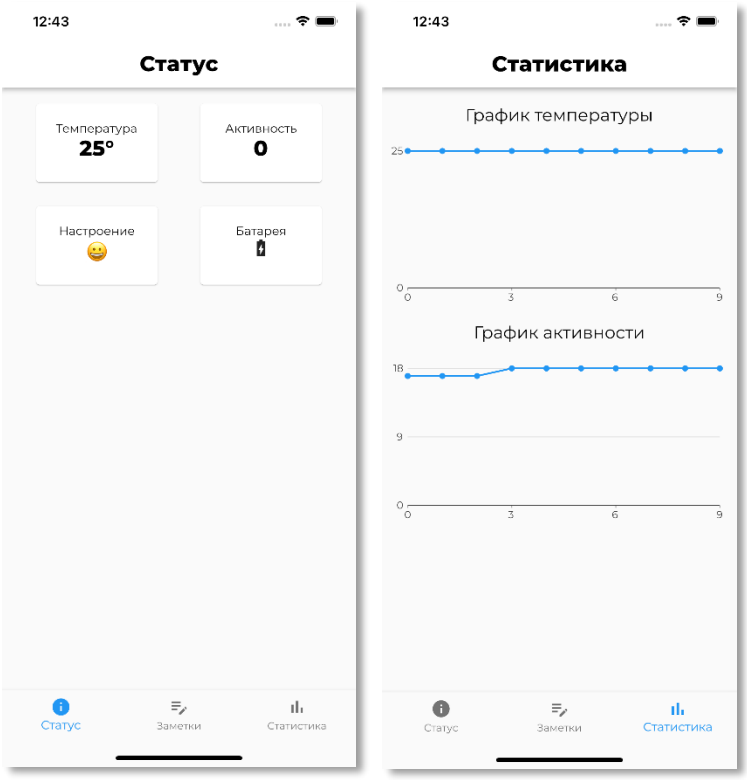
В нашей системе мы использовали библиотеку Python Vosk для распознавания речи и библиотеки gTTS, playsound для преобразования текста в речь.

## Алгоритм интерактивной работы голосового помощника и системы распознавания эмоций (самая первая версия:)



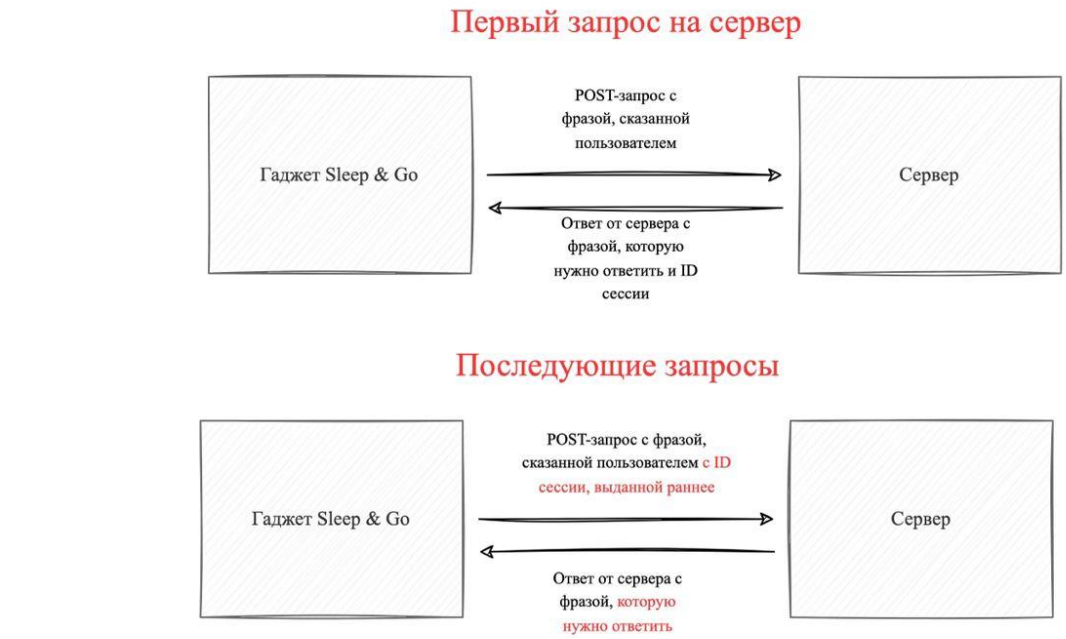
## Сенсоры и статистика

В нашем роботе-одеяле используются два типа сенсоров – датчик температуры и акселерометр. Данные с датчиков поступают в базу данных на серверной части ПО проекта. Обработанные данные можно посмотреть в виде графиков в специальном ПО для ОС Android.




## Hardware архитектура проекта

## Software архитектура проекта





## Социальное взаимодействие и инновации

Современное общество активно внедряет технологии цифровизации и робототехники. Так, к примеру, в условиях пандемии COVID-19 сформировалась потребность в телемедицинских устройствах дистанционной диагностики.

Робот-одеяло «Sleep & Go» - это инновационный шаг в развитии персональных медицинских гаджетов.

В условиях современного темпа жизни, уплотнения информационного потока, стрессов и болезней робот-одеяло значительно улучшит процессы сна.

**ВАЖНО:**

Робот одеяло может успешно применяться для малоподвижных больных и людей с ограниченными возможностями. С помощью него можно диагностировать состояние пациента, проводить массаж, помогать пациенту в вызове медицинского персонала и т.д.

**Стартап «Sleep & Go»**

Нам предстоит довести наш робот-гаджет «Sleep &Go» до промышленного образца, организовать производство и провести сертификацию.

Планируемый срок выхода нашего стартапа на продажи – конец 2023 года.

Планируемый рынок: Российская федерация, объем реализации: **1 год – 1000 единиц, 2 год до 10 тысяч единиц**, из них до 70% для нужд министерства здравоохранения.

Целевой потребитель: 30% - частные покупатели, **70% - корпоративные покупатели (Минздрав, Частные реабилитационные центры, Гостиницы)**

Предварительная оценка стоимости робо-гаджета «Sleep&Go» - **20 000 рублей.**

Ключевые затраты: опытно-конструкторские работы, организация производства, сертификация.

**Расчетный срок окупаемости проекта – 2 года.**

Планируется получение патентов на полезные модели и товарный знак.

**Команда проекта**

Проскуряков Иван

Архитектор проекта

ПО сервера обработки, протоколы взаимодействия, ПО ОС Android, аудиодетектирование

Мазуркевич Андрей

Робототехник

Робототехническое решение, машинное зрение и детектирование, алгоритмы коммуникации

# Текст ПО для интерфейса визуальной и голосовой коммуникации (колонка с камерой)

```
from tensorflow.keras import Sequential
from tensorflow.keras.models import load_model
import cv2
import json, pyaudio
import numpy as np
from tensorflow.keras.preprocessing.image import img_to_array
from threading import Thread
from http_client import Client
from vosk import Model, KaldiRecognizer
import time
from playsound import playsound
import os
import requests
from gtts import gTTS


client = Client(debug=False)

session_id = str()
found_in_cache = False
old_message = -1
message = -1


# загрузка моделей
model = Sequential() # модель для распознования эмоций
classifier = load_model('ferjj.h5')
class_labels = {0: 'Angry', 1: 'Fear', 2: 'Happy', 3: 'Neutral', 4: 'Sad', 5: 'Surprise'}
classes = list(class_labels.values())
face_classifier = cv2.CascadeClassifier('./Haarcascades/haarcascade_frontalface_default.xml')

speak_model = Model('model') # модель для распознования голоса
rec = KaldiRecognizer(speak_model, 16000)
p = pyaudio.PyAudio()
stream = p.open(format = pyaudio.paInt16, channels = 1, rate = 16000, input = True, frames_per_buffer = 8000)
stream.start_stream()
print("Говорите")

def speak(text, found_in_cache): # функция воспроизведения голоса
    name = text.replace('.', ',').replace(':', ',').replace('?', ',')
    for root, dirs, files in os.walk("./cache"):
        for filename in files:
            if filename.split('.')[0] == name:
                found_in_cache = True
    if found_in_cache == True:
        print("found in cache")
        playsound("./cache/" + name + ".mp3")
    else:
        print("not found in cache")
        voice = gTTS(text, lang="ru")
        voice.save("./cache/" + name + ".mp3")
        playsound("./cache/" + name + ".mp3")

def listen(): # функция преобразования речи в текст
    while True:
        data = stream.read(4000, exception_on_overflow = False)
        if (rec.AcceptWaveform(data)) and(len(data) > 0):
            answer = json.loads(rec.Result())
            if answer['text']:
                print("Пользователь:",answer['text'])
                return answer['text']


# функция для задания внешнего вида надписи, отображающей распознанную эмоцию
def text_on_detected_boxes(text,text_x,text_y,image,font_scale = 1,
                           font = cv2.FONT_HERSHEY_SIMPLEX,
                           FONT_COLOR = (0, 0, 0),
```

```
                    FONT_THICKNESS = 2,
                    rectangle_bgr = (0, 255, 0)):
    # определение ширины и высоты текстового поля
    (text_width, text_height) = cv2.getTextSize(text, font, fontScale=font_scale, thickness=2)[0]
    box_coords = ((text_x-10, text_y+4), (text_x + text_width+10, text_y - text_height-5))
    cv2.rectangle(image, box_coords[0], box_coords[1], rectangle_bgr, cv2.FILLED)
    cv2.putText(image, text, (text_x, text_y), font, fontScale=font_scale, color=FONT_COLOR,thickness=FONT_THICKNESS)

# распознавание эмоций на лице
def face_detector_video(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray, 1.3, 5)
    if faces is ():
        return (0, 0, 0, 0), np.zeros((48, 48), np.uint8), img
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), thickness=2)
        roi_gray = gray[y:y + h, x:x + w]
    roi_gray = cv2.resize(roi_gray, (48, 48), interpolation=cv2.INTER_AREA)
    return (x, w, y, h), roi_gray, img

def emotionVideo(cap, old_message, message): # функция для распознования эмоций в видеопотоке и отравки их на сервер
    global session_id
    flag = True
    while True:
        ret, frame = cap.read()
        rect, face, image = face_detector_video(frame)
        if np.sum([face]) != 0.0:
            roi = face.astype("float") / 255.0
            roi = img_to_array(roi)
            roi = np.expand_dims(roi, axis=0)
            # определение области, существенной для анализа (ROI), затем осуществляем поиск класса
            preds = classifier.predict(roi)[0]
            label = class_labels[preds.argmax()]
            label_position = (rect[0] + rect[1]//50, rect[2] + rect[3]//50)
            text_on_detected_boxes(label, label_position[0], label_position[1], image)
            fps = cap.get(cv2.CAP_PROP_FPS)
            cv2.putText(image, str(fps),(5, 40), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

            # if preds.argmax() == 3:
            #     message = 0
            if preds.argmax() == 2 and flag == True:
                message = 1
                flag = False
            # if preds.argmax() == 0:
            #     message = 2
            # if preds.argmax() == 5:
            #     message = 3
            if preds.argmax() == 4 and flag == True:
                message = 3
                flag = False
            if message != old_message: # отправка сообщения на сервер
                resp = client.emotion(message)
                session_id = str(resp.get('session_id'))
                print('session_id_video', session_id)
                speak(str(resp.get('result')), False)
                old_message = message
                print(message)

        else:
            cv2.putText(image, "No Face Found", (5, 40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)
        cv2.imshow('All', image)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
if __name__ == '__main__': # выполняется только если файл запущен напрямую
    camera = cv2.VideoCapture(1)
    th1 = Thread(target=emotionVideo, args=(camera, old_message, message)) # запуск видеопотока
    th1.start()
    print("run")

    time.sleep(1)

    while True: # часть кода отвечающая за разговор с пользователем
```

```
    text = listen()
    resp = client.phrase(text, session_id=session_id)

    if not session_id:
        session_id = resp.get('session_id')

    if int(resp.get('code')) > 1: # Команда не распознана
        print(f'Команда не распознана {resp.get("code")}')

    else:
        speak(str(resp.get('result')), False)
        print("Одеяло:", str(resp.get('result')))
        if int(resp.get('command_code')) == 1:
            playsound('./music.mp3')
        if int(resp.get('command_code')) == 2:
            playsound('./budilnik.wav')
        if int(resp.get('command_code')) == 3:
            playsound('./chto_ti_umeesh.mp3'
```

# Текст ПО обработчика эмоций  и фраз (сервер)

```
@csrf_exempt
def phrase_renderer(request):
    '''
    code 1 - phrase_text is not exists
    code 2 - no keyword

    command_code 1 - включить музыку
    '''

    # Получение параметров POST-запроса

    session_id = request.POST.get('session')
    phrase_text = request.POST.get('phrase')

    print('session_id', session_id)
    print('phrase_text', phrase_text)

    # Проверка параметров POST-запроса
    if not phrase_text:
        return JsonResponse({
            'success': False,
            'code': 1
        }, status=400)

    # Проверка на ключевое слово "одеяло"
    if not 'одеял' in phrase_text or 'одеяло' == phrase_text:
        return JsonResponse({
            'success': False,
            'code': 3
        }, status=400)

    # Проверка, есть ли сессия

    if not session_id:
        # Сессии нет, создаем новую сессию и фразу
        session = Session.objects.create()
        Phrase.objects.create(session=session, text=phrase_text.split(' ', 1)[1])

    else:
        # Сессия есть, находим ее
        sessions = Session.objects.filter(session_id=session_id)

        # Проверка, является ли сессия инвалидной
        if not sessions:
            return JsonResponse({
                'success': False,
                'code': 4
            }, status=400)

        # Получаем объект сессии
```

```
        session = Session.objects.get(session_id=session_id)

        # Создаем фразу
        Phrase.objects.create(session=session, text=phrase_text)

    # Получаем ответ на фразу из самописной функции dialog_menu
    result_answer = dialog_menu(session=session)
    command_code = 0

    print('answer:', result_answer)

    # Проверка, есть ли результат
    if not result_answer:
        return JsonResponse({
            'success': False,
            'code': 5,
            'command_code': command_code,
            'result': result_answer,
            'session_id': session.session_id
        }, status=400)

    # Проверка, есть ли command_code
    if type(result_answer) is tuple:
        command_code, result_answer = result_answer

    else:
        command_code = 0

    return JsonResponse({
        'success': True,
        'code': 0,
        'command_code': command_code,
        'result': result_answer,
        'session_id': session.session_id
    })

@csrf_exempt
def emotion_renderer(request):
    '''
    code 1 - emotion is not exists
    '''

    # Получение аргументов POST-запроса
    session_id = request.POST.get('session')
    emotion = request.POST.get('emotion')

    print('session_id', session_id)
    print('emotion', emotion)

    # Проверки (схожие с проверками в phrase_renderer)

    if not emotion:
        return HttpResponseBadRequest(1)

    if not session_id:
        session = Session.objects.create()

    else:
        sessions = Session.objects.filter(session_id=session_id)

        if not sessions:
            return HttpResponseBadRequest(2)

        session = Session.objects.get(session_id=session_id)

    # Создание новой эмоции, получение ответа

    Emotion.objects.create(session=session, code=int(emotion))

    result_answer = emotion_dialog_menu(session=session)

    print(result_answer)

    return JsonResponse({
```

```
    'result': result_answer,
    'session_id': session.session_id
})
```