Министерство науки и высшего образования РФ

ФГАОУ ВО «Уральский федеральный университет

имени первого Президента России Б. Н. Ельцина»

Институт радиоэлектроники и информационных технологий

Кафедра учебно-научный центр “Информационная безопасность”

Пояснительная записка к курсовому проекту

Тема: «Преобразование текста в речь»

по дисциплине «Языки и методы программирования»

Студенты:

гр. РИ-311055 Кабирова Е. А., Левашов В. А. Нохрина Д. Д.

Преподаватель: Мирвода С. Г.

Екатеринбург

2024

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc156050821)

[Заявление видения 3](#_Toc156050822)

[Видение продукта 3](#_Toc156050823)

[РУКОВОДИТЕЛЬ 4](#_Toc156050824)

[Функциональная диаграмма 4](#_Toc156050825)

[ПРОЕКТИРОВЩИК 5](#_Toc156050826)

[Архитектура продукта 5](#_Toc156050827)

[Требования к системному ПО 5](#_Toc156050828)

[Архитектурная диаграмма 5](#_Toc156050829)

[Функциональная диаграмма 6](#_Toc156050830)

[Диаграмма последовательности 7](#_Toc156050831)

[Концептуальная модель БД 8](#_Toc156050832)

[РАЗРАБОТЧИК 9](#_Toc156050833)

[Выбранные языки программирования, обоснование выбора 9](#_Toc156050834)

[Диаграмма пакетов 9](#_Toc156050835)

[Диаграмма физической модели 9](#_Toc156050836)

[ЗАКЛЮЧЕНИЕ 10](#_Toc156050837)

[СПИСОК ЛИТЕРАТУРЫ 11](#_Toc156050838)

[ПРИЛОЖЕНИЕ А 12](#_Toc156050839)

[Код 12](#_Toc156050840)

# ВВЕДЕНИЕ

## Заявление видения

«Голос твоего текста — твой выбор!»

## Видение продукта

Цель: предоставить людям возможность персонализировать опыт чтения и прослушивания контента, давая им возможность выбирать голос, который лучше всего подходит для их потребностей: анонимности, улучшения восприятия голоса, помощи людям с ограниченными возможностями, развлечения и т.д.

Концепция: создание продукта основывается на идее свободы и креативности. Мы стремимся создать платформу, где каждый человек может найти подходящий голос для своих текстов и уникально выразить свои мысли и идеи. Для людей с ограниченными возможностями платформа позволит лучше передавать информацию собеседнику. Озвучивание текста голосом может помочь людям, предпочитающим слушать вместо чтения, а также желающим сохранить анонимность в общении, не раскрывая свою личность.

Мотив: расширение возможности самовыражения и улучшение взаимодействия людей с технологией.

План действий:

1. Разработка прототипа и концепции приложения, выбор технологии озвучивания.

2. Создание базы данных записей и личного кабинета для каждого пользователя.

3. Разработка кода, который будет преобразовывать написанный пользователем текст в озвучку выбранного голоса.

4. Создание удобного интерфейса, который позволит пользователям выбирать голоса, подходящие для озвучивания своих текстов.

5. Тестирование приложения и исправление возможных ошибок.

6. Запуск продукта.

7. Развитие и постоянное улучшение продукта, добавление новых функций.

# РУКОВОДИТЕЛЬ

## Функциональная диаграмма

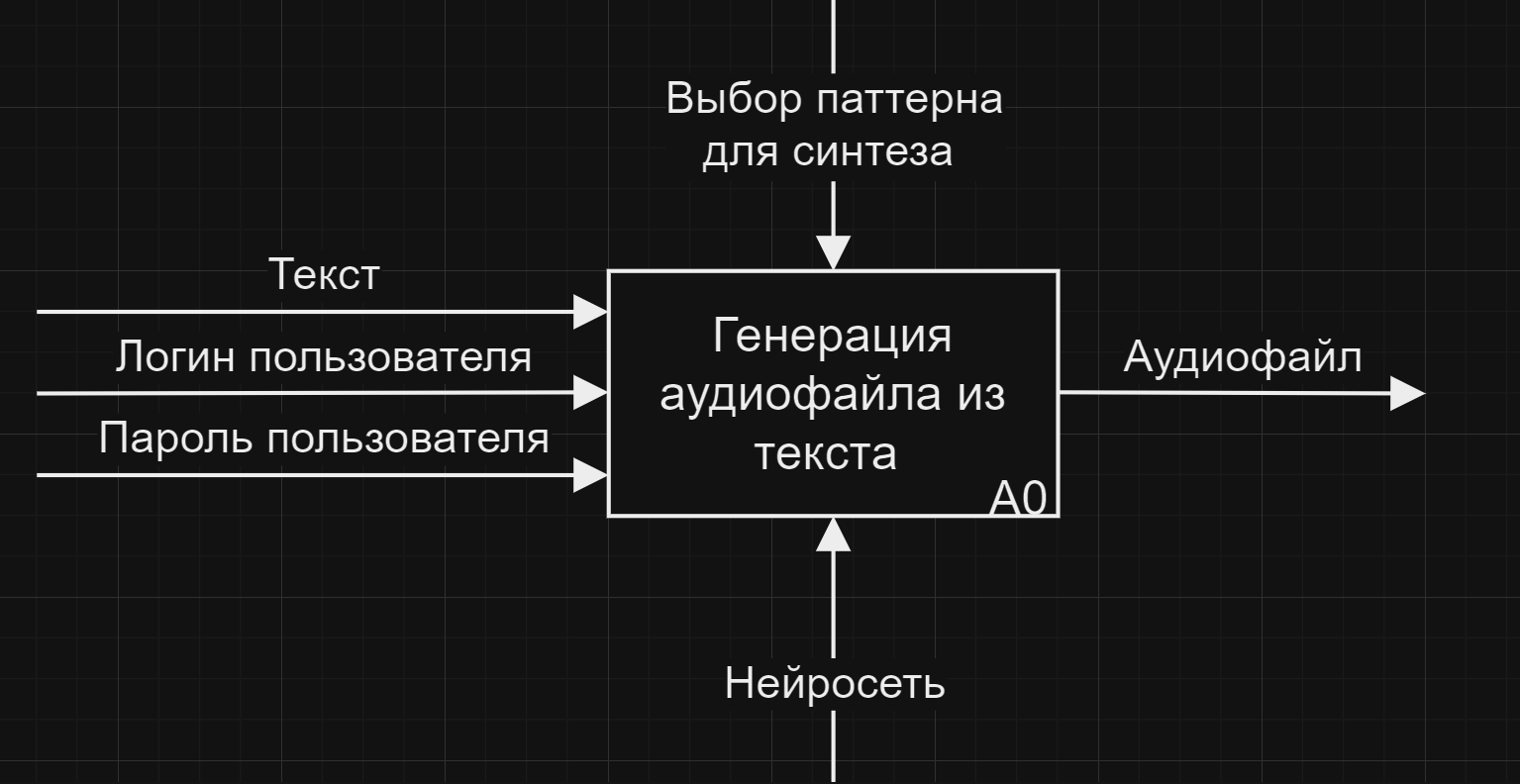
****

Рисунок 1 – Исходная функциональная диаграмма

# ПРОЕКТИРОВЩИК

## Архитектура продукта

1. Тело программы: включает в себя модули, отвечающие за взаимодействие с нейросетью, обучение моделей, преобразование массивов данных, связь с БД, выстраивание графической среды (построение GUI)

2. Хранилище данных: база данных для хранения голосовых и пользовательских данных

3. Пользовательский интерфейс (UI): часть приложения, с которой пользователь взаимодействует для ввода текста и выбора голоса

4. Аутентификация и управление учетными записями: компонент, предоставляющий безопасное управление учетными записями пользователей, аутентификацию и регистрацию.

## Требования к системному ПО

Для обеспечения эффективной работы приложения важно запускать его на операционной системе Windows, так как разработка и тестирование осуществлялись на этой платформе, а также система управления базами данных PostgreSQL 15. Особое внимание следует уделить ознакомлению с руководством по развертыванию нейросети на персональном компьютере, которое поможет правильно настроить объекты и предоставить необходимые ресурсы для работы приложения. Необходимый файл может быть найден и загружен на ресурсе GitHub, где также доступна подробная информация и инструкции по развертыванию и настройке.

## Архитектурная диаграмма

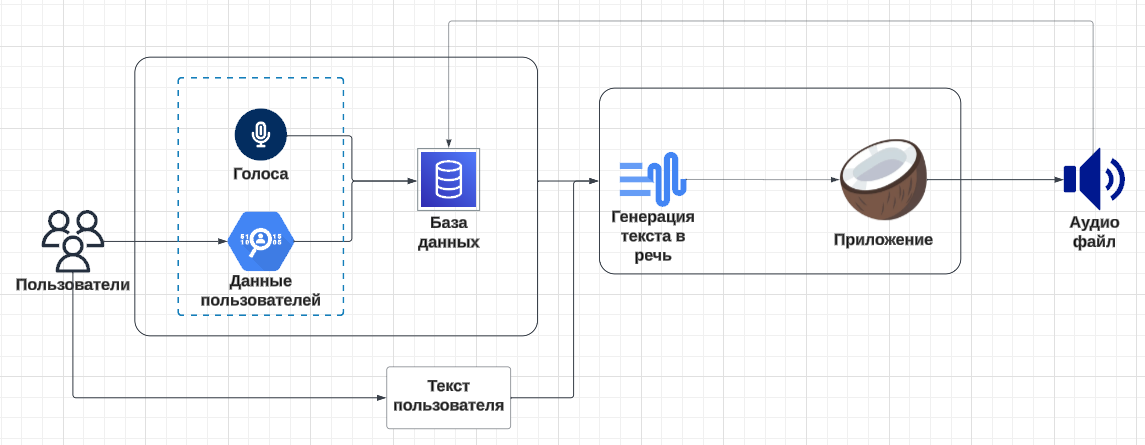
****

Рисунок 2 – Архитектурная диаграмма

## Функциональная диаграмма

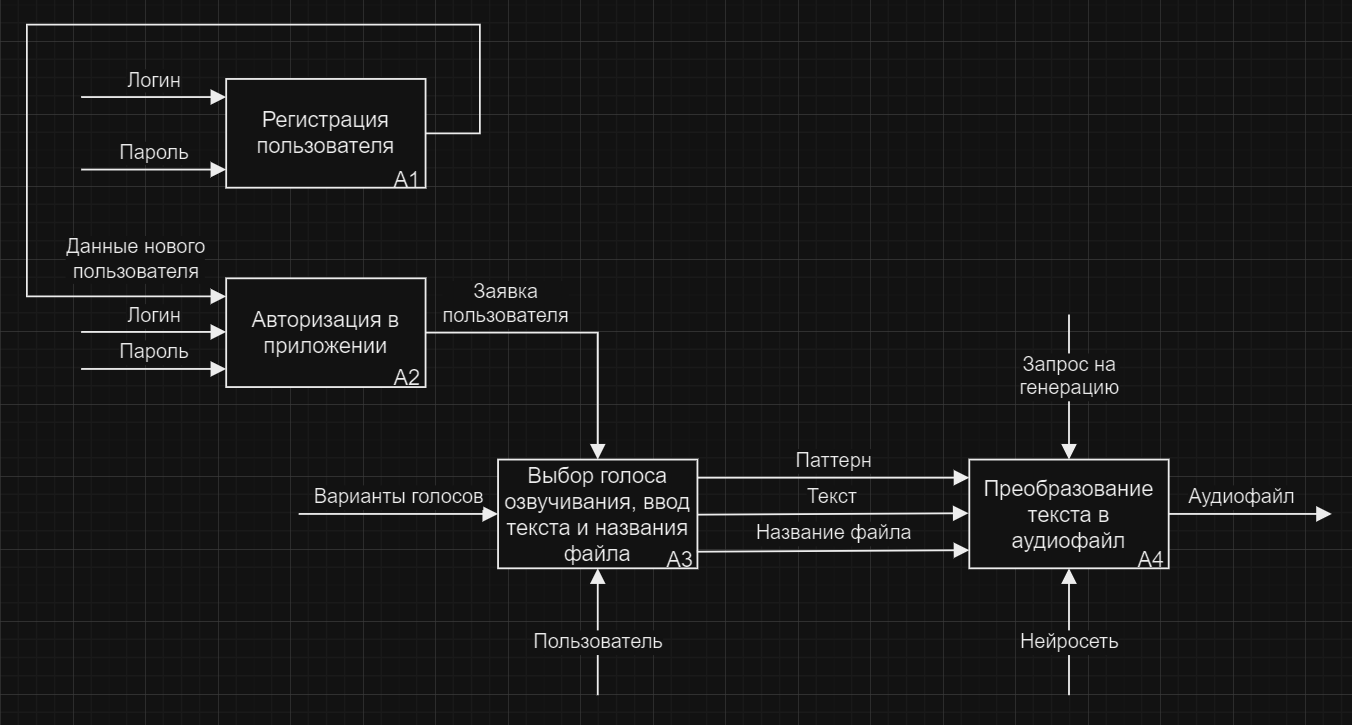


Рисунок 3 – Фактическая детализированная функциональная диаграмма

## Диаграмма последовательности

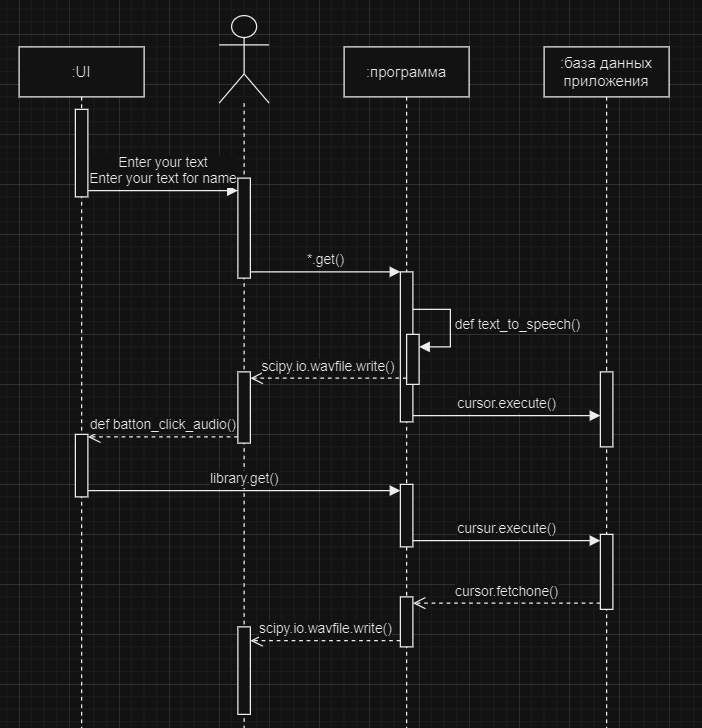


Рисунок 4 – Регистрация в приложении

## Концептуальная модель БД

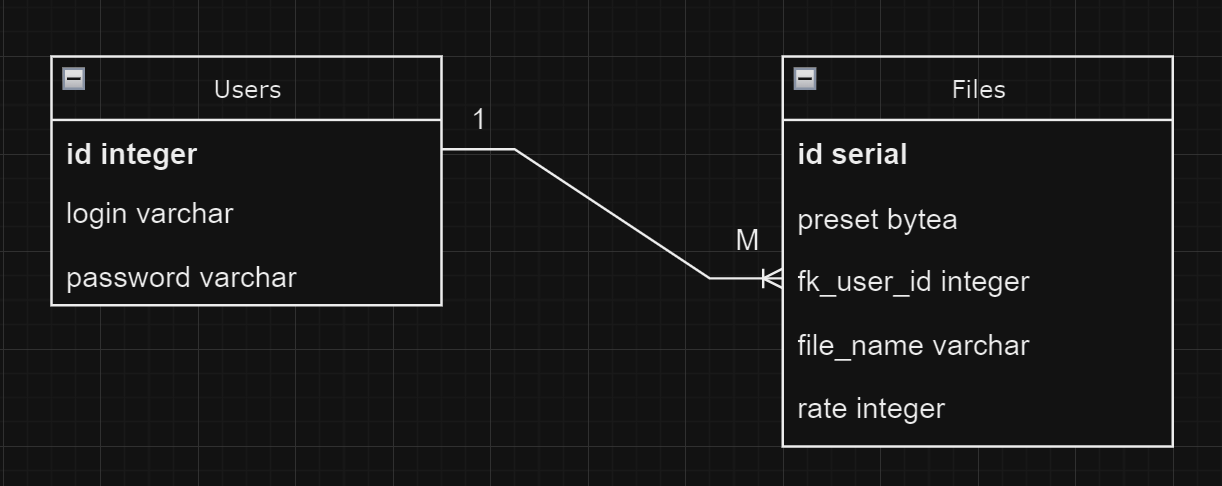
****

Рисунок 5 – Концептуальная модель БД

# РАЗРАБОТЧИК

## Выбранные языки программирования, обоснование выбора

Язык программирования Python был выбран в связи с его превосходной поддержкой нейронных сетей и обширным набором библиотек, что необходимо для реализации нашей идеи.

Мы приняли решение в пользу PostgreSQL, поскольку мы уже знакомы с ним благодаря предыдущим курсам, и работа с ним кажется нам более интуитивно понятной. Это важно для понимания структуры баз данных и способов работы с различными типами файлов, которые могут быть в ней сохранены.

## Диаграмма пакетов

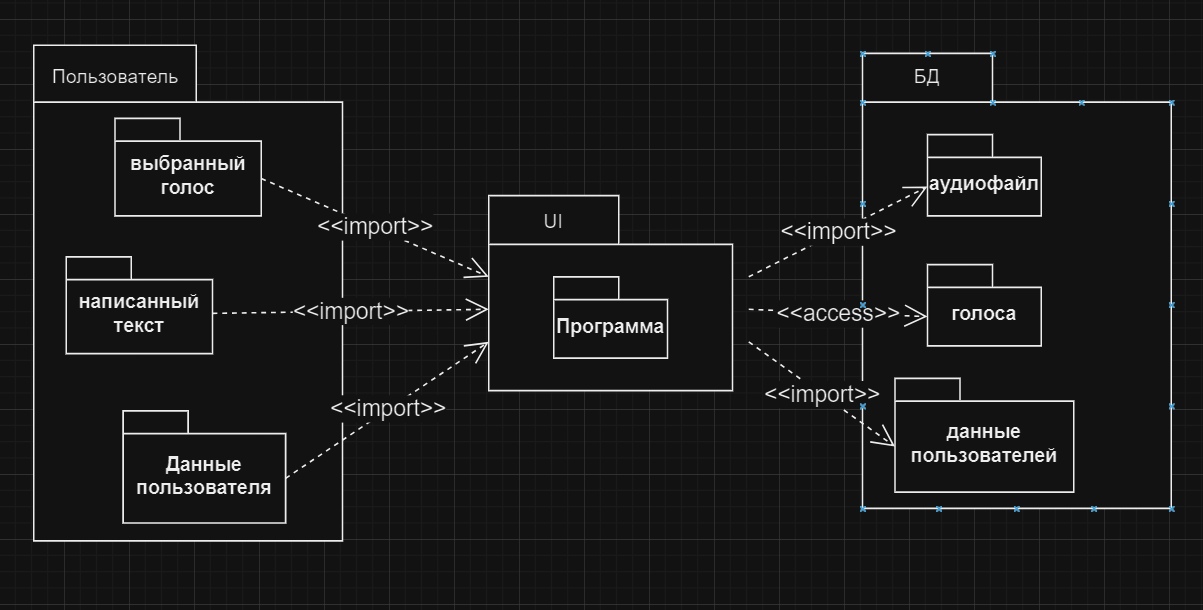


Рисунок 6 – Диаграмма пакетов

## Диаграмма физической модели

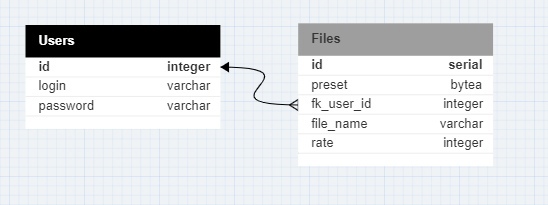


Рисунок 7 – Диаграмма физической модели

# ЗАКЛЮЧЕНИЕ

Наше приложение «Text To Speech» представляет собой эффективную программу для генерации текста в речь. В использовании легко выбирать паттерн для синтеза, а также вводить желаемый текст для озвучивания. В целом, это удобное приложение для создания аудиофайлов, наполнение и озвучивание которых Вы выбираете сами.

# СПИСОК ЛИТЕРАТУРЫ

1. Что такое построение архитектурных диаграмм? [Электронный ресурс]. - URL: <https://aws.amazon.com/ru/what-is/architecture-diagramming/>
2. What is Product Vision? [Электронный ресурс]. - URL: <https://www.tryexponent.com/blog/what-is-product-vision>
3. Минимально Жизнеспособный Продукт (Minimum Viable Product, MVP) [Электронный ресурс]. - URL: <https://scrumtrek.ru/blog/agile-scrum/scrum-glossary/3793/minimum-viable-product/>
4. Иванова Г.С. Технология программирования: Учебник для вузов. - М.: Изд-во МГТУ им. Н.Э. Баумана, 2002. – С. 78-81.
5. Фаулер M. UML. Основы, 3-е издание – Пер. с англ. – СПб: Символ-Плюс, 2004. – С. 80-120.

# ПРИЛОЖЕНИЕ А

## Код

from tkinter import \*

from tkinter import messagebox

from tkinter import ttk

import psycopg2

import numpy as np

from transformers import BarkModel, AutoProcessor

import torch

import scipy

def add\_window():

window = Tk()

window['bg'] = '#fafafa'

window.title('text-to-speech')

window.geometry('300x500')

window.resizable(width=FALSE,height=FALSE)

return(window)

def button\_click():

login = loginInput.get()

passwd = passField.get()

try:

# connect to exist database

connection = psycopg2.connect(

host="127.0.0.1",

user="postgres",

password="123",

database="postgres"

)

connection.autocommit = True

with connection.cursor() as cursor:

sql = "select True from Users where (login = %s)and(password = %s)"

cursor.execute(sql, (login, passwd))

output = cursor.fetchone()

print(f" {output}")

except Exception as \_ex:

print("[INFO] Error while working with PostgreSQL", \_ex)

finally:

if connection:

# cursor.close()

connection.close()

print("[INFO] PostgreSQL connection closed")

if output==None:

messagebox.showerror(title='autentic error',message='you write something wrong or please reg')

elif login != '' and passwd != '' and output[0]==True:

def button\_click\_audio():

choice = your\_bible.get()

if choice == '':

return(messagebox.showerror(title='Warning',message='fild shoud not be empty'))

str\_name = 'r'+str(choice)

try:

# connect to exist database

connection = psycopg2.connect(

host="127.0.0.1",

user="postgres",

password="123",

database="postgres"

)

connection.autocommit = True

with connection.cursor() as cursor:

sql = "SELECT preset,rate FROM Files WHERE file\_name = %s"

cursor.execute(sql, (choice, ))

row = cursor.fetchone()

print(f" {row}")

except Exception as \_ex:

print("[INFO] Error while working with PostgreSQL", \_ex)

finally:

if connection:

# cursor.close()

connection.close()

print("[INFO] PostgreSQL connection closed")

audio\_data = row[0]

print(audio\_data)

audio\_array = np.frombuffer(audio\_data, dtype=np.int64)

audio\_array = audio\_array/100000000000000

audio\_array = audio\_array.astype(np.float32)

print(audio\_array)

print(len(audio\_array))

print(type(audio\_array))

print(audio\_array.dtype)

scipy.io.wavfile.write(str\_name, rate=row[1], data=audio\_array)

def button\_click\_text\_to\_speech(temp\_fk\_id = login):

preset = your\_bibler\_presset.get()

text\_for\_using = str(textField.get())

text\_for\_name = str(textField1.get())+'.wav'

text\_for\_name\_temp = str(textField1.get())

if (preset == '') or (text\_for\_name\_temp == '') or (text\_for\_using == ''):

return(messagebox.showerror(title='Warning',message='fild shoud not be empty'))

fk\_id = temp\_fk\_id

print(fk\_id)

model = BarkModel.from\_pretrained('suno/bark')

device = 'cuda' if torch.cuda.is\_available() else 'cpu'

model = model.to(device)

processor = AutoProcessor.from\_pretrained('suno/bark')

inputs = processor(text\_for\_using, voice\_preset = preset).to(device)

audio\_array2 = model.generate(\*\*inputs)

audio\_array2 = audio\_array2.cpu().numpy().squeeze()

print(audio\_array2)

print(len(audio\_array2))

sample\_rate = model.generation\_config.sample\_rate

scipy.io.wavfile.write(text\_for\_name,rate = sample\_rate,data=audio\_array2)

print('ffffffffffffffffffffffffffff')

rate, audio\_array = scipy.io.wavfile.read(text\_for\_name)

print(type(audio\_array))

print(audio\_array.dtype)

print(len(audio\_array))

# Преобразование аудиофайла в тип bytea

audio\_data = audio\_array\*100000000000000

print('file base:', audio\_data)

print('file not base:', audio\_data.astype(np.int64))

audio\_data = audio\_data.astype(np.int64).tobytes()

print(len(audio\_data))

'''audio\_array = np.frombuffer(audio\_data, dtype=np.int64)

audio\_array = audio\_array/100000000000000

audio\_array = audio\_array.astype(np.float32)

print(audio\_array)

print(len(audio\_array))

print(type(audio\_array))

print(audio\_array.dtype)

scipy.io.wavfile.write('восстановленный\_файл1.wav', rate=rate, data=audio\_array)'''

try:

# connect to exist database

connection = psycopg2.connect(

host="127.0.0.1",

user="postgres",

password="123",

database="postgres"

)

connection.autocommit = True

with connection.cursor() as cursor:

insert\_query1 = "select id from Users where login = %s"

cursor.execute(insert\_query1, (fk\_id,))

output = cursor.fetchone()

insert\_query = "INSERT INTO files (file\_name, preset, fk\_user\_id,rate) VALUES (%s, %s, %s, %s)"

cursor.execute(insert\_query, (text\_for\_name, audio\_data, output[0], rate))

output = cursor.fetchone()

print(f" {output}")

except Exception as \_ex:

print("[INFO] Error while working with PostgreSQL", \_ex)

finally:

if connection:

# cursor.close()

connection.close()

print("[INFO] PostgreSQL connection closed")

window1.destroy()

new\_window = add\_window()

new\_canvas=Canvas(new\_window, height=300, width=250)

new\_canvas.pack

new\_frame=Frame(new\_window, bg='red')

new\_frame.place(relx=0.15, rely=0.15, relwidth=0.7,relheight=0.7)

try:

# connect to exist database

connection = psycopg2.connect(

host="127.0.0.1",

user="postgres",

password="123",

database="postgres"

)

connection.autocommit = True

with connection.cursor() as cursor:

insert\_query1 = "select id from Users where login = %s"

cursor.execute(insert\_query1, (login,))

output = cursor.fetchone()

sql = "select \* from Files where fk\_user\_id = %s"

cursor.execute(sql, (output[0],))

masss = cursor.fetchall()

except Exception as \_ex:

print("[INFO] Error while working with PostgreSQL", \_ex)

finally:

if connection:

# cursor.close()

connection.close()

print("[INFO] PostgreSQL connection closed")

your\_presset=[]

for i in range(len(masss)):

your\_presset.append(masss[i][3])

def button\_click\_refresh():

try:

# connect to exist database

connection = psycopg2.connect(

host="127.0.0.1",

user="postgres",

password="123",

database="postgres"

)

connection.autocommit = True

with connection.cursor() as cursor:

insert\_query1 = "select id from Users where login = %s"

cursor.execute(insert\_query1, (login,))

output = cursor.fetchone()

sql = "select \* from Files where fk\_user\_id = %s"

cursor.execute(sql, (output[0],))

masss = cursor.fetchall()

except Exception as \_ex:

print("[INFO] Error while working with PostgreSQL", \_ex)

finally:

if connection:

# cursor.close()

connection.close()

print("[INFO] PostgreSQL connection closed")

your\_presset=[]

for i in range(len(masss)):

your\_presset.append(masss[i][3])

your\_bible['values']=your\_presset

selected\_option = StringVar()

your\_bible = ttk.Combobox(new\_frame, textvariable=selected\_option, values=your\_presset, state='readonly')

your\_bible.pack(pady=(20,0))

press\_me\_new = Button(new\_frame, text = 'select', bg='grey',command=button\_click\_audio)

press\_me\_new.pack(pady=5)

press\_me\_new = Button(new\_frame, text = 'refresh', bg='grey',command=button\_click\_refresh)

press\_me\_new.pack()

choice\_your\_voice = ['v2/ru\_speaker\_1','v2/ru\_speaker\_2','v2/ru\_speaker\_3','v2/ru\_speaker\_5','v2/ru\_speaker\_6']

selected\_option2 = StringVar()

your\_bibler\_presset = ttk.Combobox(new\_frame, textvariable=selected\_option2, values=choice\_your\_voice, state='readonly')

your\_bibler\_presset.pack(pady=(10,0))

new\_title = Label(new\_frame, text='Enter your text',bg='grey',font=30)

new\_title.pack(pady=(5,5))

textField = Entry(new\_frame, bg='white')

textField.pack(pady=(0,5))

new\_title1 = Label(new\_frame, text='Enter your text for name',bg='grey',font=30)

new\_title1.pack(pady=(5,5))

textField1 = Entry(new\_frame, bg='white')

textField1.pack(pady=(0,5))

press\_me\_new1 = Button(new\_frame, text = 'text to speech', bg='grey',command=button\_click\_text\_to\_speech)

press\_me\_new1.pack()

new\_window.mainloop()

else:

messagebox.showerror(title='warning',message='This login is not available')

def regist\_click():

def button\_click\_regist():

temp\_output = [False,True]

login = loginInput1.get()

passwd = passField1.get()

if (login == '') or (passwd == ''):

return(messagebox.showerror(title='Warning',message='filds shoud not be empty'))

try:

# connect to exist database

connection = psycopg2.connect(

host="127.0.0.1",

user="postgres",

password="123",

database="postgres"

)

connection.autocommit = True

with connection.cursor() as cursor:

sql = "SELECT True FROM Users where login = %s"

cursor.execute(sql, (login, ))

output1 = cursor.fetchone()

if output1 == None:

output1 = temp\_output

sql = "INSERT INTO Users(login, password) VALUES (%s, %s); SELECT True FROM Users where login = %s"

cursor.execute(sql, (login, passwd, login))

output = cursor.fetchone()

print(f" {output}")

except Exception as \_ex:

print("[INFO] Error while working with PostgreSQL", \_ex)

finally:

if connection:

# cursor.close()

connection.close()

print("[INFO] PostgreSQL connection closed")

print(output1)

if output1[0]==True:

messagebox.showerror(title='Warning',message='This login is not available')

elif output[0]==True:

def button\_click\_audior():

choice = your\_bibler.get()

if choice!='':

str\_name = 'r'+str(choice)

try:

# connect to exist database

connection = psycopg2.connect(

host="127.0.0.1",

user="postgres",

password="123",

database="postgres"

)

connection.autocommit = True

with connection.cursor() as cursor:

sql = "SELECT preset,rate FROM Files WHERE file\_name = %s"

cursor.execute(sql, (choice, ))

row = cursor.fetchone()

print(f" {row}")

except Exception as \_ex:

print("[INFO] Error while working with PostgreSQL", \_ex)

finally:

if connection:

# cursor.close()

connection.close()

print("[INFO] PostgreSQL connection closed")

audio\_data = row[0]

print(audio\_data)

audio\_array = np.frombuffer(audio\_data, dtype=np.int64)

audio\_array = audio\_array/100000000000000

audio\_array = audio\_array.astype(np.float32)

print(audio\_array)

print(len(audio\_array))

print(type(audio\_array))

print(audio\_array.dtype)

scipy.io.wavfile.write(str\_name, rate=row[1], data=audio\_array)

else:

messagebox.showerror(title='Warning',message='filds shoud not be empty')

def button\_click\_text\_to\_speechr(temp\_fk\_id = login):

text\_for\_using = str(textFieldr.get())

text\_for\_name = str(textFieldr1.get())+'.wav'

text\_for\_name\_temp = str(textFieldr1.get())

preset = your\_bibler\_presset.get()

if (text\_for\_using != '') and (text\_for\_name\_temp != '') and (preset != ''):

fk\_id = temp\_fk\_id

print(fk\_id)

model = BarkModel.from\_pretrained('suno/bark')

device = 'cuda' if torch.cuda.is\_available() else 'cpu'

model = model.to(device)

processor = AutoProcessor.from\_pretrained('suno/bark')

inputs = processor(text\_for\_using, voice\_preset = preset).to(device)

audio\_array2 = model.generate(\*\*inputs)

audio\_array2 = audio\_array2.cpu().numpy().squeeze()

print(audio\_array2)

print(len(audio\_array2))

sample\_rate = model.generation\_config.sample\_rate

scipy.io.wavfile.write(text\_for\_name,rate = sample\_rate,data=audio\_array2)

print('ffffffffffffffffffffffffffff')

rate, audio\_array = scipy.io.wavfile.read(text\_for\_name)

print(type(audio\_array))

print(audio\_array.dtype)

print(len(audio\_array))

# Преобразование аудиофайла в тип bytea

audio\_data = audio\_array\*100000000000000

print('file base:', audio\_data)

print('file not base:', audio\_data.astype(np.int64))

audio\_data = audio\_data.astype(np.int64).tobytes()

print(len(audio\_data))

try:

# connect to exist database

connection = psycopg2.connect(

host="127.0.0.1",

user="postgres",

password="123",

database="postgres"

)

connection.autocommit = True

with connection.cursor() as cursor:

insert\_query1 = "select id from Users where login = %s"

cursor.execute(insert\_query1, (fk\_id,))

output = cursor.fetchone()

insert\_query = "INSERT INTO files (file\_name, preset, fk\_user\_id,rate) VALUES (%s, %s, %s, %s)"

cursor.execute(insert\_query, (text\_for\_name, audio\_data, output[0], rate))

output = cursor.fetchone()

print(f" {output}")

except Exception as \_ex:

print("[INFO] Error while working with PostgreSQL", \_ex)

finally:

if connection:

# cursor.close()

connection.close()

print("[INFO] PostgreSQL connection closed")

else:

messagebox.showerror(title='Warning',message='filds shoud not be empty')

try:

# connect to exist database

connection = psycopg2.connect(

host="127.0.0.1",

user="postgres",

password="123",

database="postgres"

)

connection.autocommit = True

with connection.cursor() as cursor:

insert\_query1 = "select id from Users where login = %s"

cursor.execute(insert\_query1, (login,))

output = cursor.fetchone()

sql = "select \* from Files where fk\_user\_id = %s"

cursor.execute(sql, (output[0],))

masss = cursor.fetchall()

except Exception as \_ex:

print("[INFO] Error while working with PostgreSQL", \_ex)

finally:

if connection:

# cursor.close()

connection.close()

print("[INFO] PostgreSQL connection closed")

your\_presset=[]

for i in range(len(masss)):

your\_presset.append(masss[i][3])

def button\_click\_refreshr():

try:

# connect to exist database

connection = psycopg2.connect(

host="127.0.0.1",

user="postgres",

password="123",

database="postgres"

)

connection.autocommit = True

with connection.cursor() as cursor:

insert\_query1 = "select id from Users where login = %s"

cursor.execute(insert\_query1, (login,))

output = cursor.fetchone()

sql = "select \* from Files where fk\_user\_id = %s"

cursor.execute(sql, (output[0],))

masss = cursor.fetchall()

except Exception as \_ex:

print("[INFO] Error while working with PostgreSQL", \_ex)

finally:

if connection:

# cursor.close()

connection.close()

print("[INFO] PostgreSQL connection closed")

your\_presset=[]

for i in range(len(masss)):

your\_presset.append(masss[i][3])

your\_bibler['values']=your\_presset

new\_window1.destroy()

new\_windowr = add\_window()

new\_canvasr=Canvas(new\_windowr, height=300, width=250)

new\_canvasr.pack

new\_framer=Frame(new\_windowr, bg='red')

new\_framer.place(relx=0.15, rely=0.15, relwidth=0.7,relheight=0.7)

your\_voice = your\_presset

selected\_option = StringVar()

your\_bibler = ttk.Combobox(new\_framer, textvariable=selected\_option, values=your\_voice, state='readonly')

your\_bibler.pack(pady=(20,0))

press\_me\_newr = Button(new\_framer, text = 'select', bg='grey',command=button\_click\_audior)

press\_me\_newr.pack(pady=5)

press\_me\_new = Button(new\_framer, text = 'refresh', bg='grey',command=button\_click\_refreshr)

press\_me\_new.pack()

choice\_your\_voice = ['v2/ru\_speaker\_1','v2/ru\_speaker\_2','v2/ru\_speaker\_3','v2/ru\_speaker\_5','v2/ru\_speaker\_6']

selected\_option1 = StringVar()

your\_bibler\_presset = ttk.Combobox(new\_framer, textvariable=selected\_option1, values=choice\_your\_voice, state='readonly')

your\_bibler\_presset.pack(pady=(10,0))

new\_titler = Label(new\_framer, text='Enter your text',bg='grey',font=30)

new\_titler.pack(pady=(5,5))

textFieldr = Entry(new\_framer, bg='white')

textFieldr.pack(pady=(0,5))

new\_titler1 = Label(new\_framer, text='Enter your text for name',bg='grey',font=30)

new\_titler1.pack(pady=(5,5))

textFieldr1 = Entry(new\_framer, bg='white')

textFieldr1.pack(pady=(0,5))

press\_me\_new1r = Button(new\_framer, text = 'text to speech', bg='grey',command=button\_click\_text\_to\_speechr)

press\_me\_new1r.pack()

new\_windowr.mainloop()

window1.destroy()

new\_window1 = add\_window()

new\_canvas1=Canvas(new\_window1, height=300, width=250)

new\_canvas1.pack

new\_frame1=Frame(new\_window1, bg='red')

new\_frame1.place(relx=0.15, rely=0.15, relwidth=0.7,relheight=0.7)

title1 = Label(new\_frame1, text='Enter your login',bg='grey',font=40)

title1.pack(pady=(30,5))

loginInput1 = Entry(new\_frame1,bg = 'white')

loginInput1.pack()

title1 = Label(new\_frame1, text='Enter your password',bg='grey',font=40)

title1.pack(pady=(10,5))

passField1 = Entry(new\_frame1, bg='white', show='\*')

passField1.pack(pady=(0,2))

press\_me1 = Button(new\_frame1, text = 'confirm', bg='grey',command=button\_click\_regist)

press\_me1.pack()

window1 = add\_window()

canvas = Canvas(window1, height=300, width=250)

canvas.pack()

frame = Frame(window1,bg='red')

frame.place(relx=0.15, rely=0.15, relwidth=0.7,relheight=0.7)

title = Label(frame, text='Enter your login',bg='grey',font=40)

title.pack(pady=(30,5))

loginInput = Entry(frame,bg = 'white')

loginInput.pack()

title = Label(frame, text='Enter your password',bg='grey',font=40)

title.pack(pady=(10,5))

passField = Entry(frame, bg='white', show='\*')

passField.pack(pady=(0,2))

press\_me = Button(frame, text = 'confirm', bg='grey',command=button\_click)

press\_me.pack(side=LEFT,padx=10)

press\_me = Button(frame, text = 'registration', bg='grey',command=regist\_click)

press\_me.pack(side=LEFT,padx=10)

window1.mainloop()