Министерство образования и высшего образования РФ
Санкт-Петербургский государственный университет Петра Великого
Институт компьютерных наук и кибербезопасности
Высшая школа киберфизических систем и управления

УТВЕРЖДАЮ

_____

_____

_____

«_____» _____ _____ г.

ОТЧЕТ

о прохождении учебной (ознакомительной) практики

Веб-приложение для управления аудио-оборудованием с Raspberry Pi 4

Выполнил:

студент гр. 5132704/20001 _____ А. А. Ларионенко

подпись, дата

Проверил:

старший преподаватель ВШКФСиУ _____ А. В. Милицын

подпись, дата

Санкт-Петербург 2023 г.

Реферат

Отчет 32 с., 9 рис., 3 источника.

МИКРОКОМПЬЮТЕР, ПРИЛОЖЕНИЕ, ПРОГРАММИРОВАНИЕ, ВЕБ-ИНТЕРФЕЙС, ВИРТУАЛЬНЫЙ СЕРВЕР, PYTHON, HTTP, WEBSOCKETS

Цель работы – создать веб-приложение (веб-интерфейс), который позволит управлять аудиосистемой удалённо (включать, выключать, переключать музыку, регулировать громкость и т.д.) с устройства на любой операционной системе (Windows, MacOS, Linux, Android, iOS)

В процессе работы проводилось написание кода самого приложения, создание визуального интерфейса на стороне клиента и тестирование (отладка) всех функций.

Содержание

Введение

В различных центрах и летних лагерях существуют летние сцены или другие подобные места, где может быть размещено разнообразное оборудование для проигрывания музыки (активные или пассивные колонки, усилители, микшеры и т.п.). Часто, когда на данных сценах не проводится никакое мероприятие, включается фоновая музыка с компьютера, который находится непосредственно рядом с оборудованием и который подключен к аудио-оборудованию посредством кабеля (USB или TRS).

Данный метод управления музыкой является недостаточно удобным, так как в случае, когда требуется переключить музыку, выбрать другое расположение папки с музыкой или изменить выходную громкость, необходимо находится непосредственно за компьютером, с которого передаётся сигнал на аудио-оборудование.

Веб-приложение, о создании которого рассказывается в данном реферате, сильно упрощает взаимодействие с подобным аудио-оборудованием, позволяя управлять музыкой, находясь в любом месте, где имеется доступ к интернету.

1 Подготовка к разработке

1.1 Требуемые компоненты

Для реализации данного проекта понадобятся следующие компоненты:

1. Микроконтроллер с TRS-разъёмом и возможностью подключения к интернету по LAN/WLAN протоколу, который будет находиться непосредственно рядом с аудио-оборудованием и который будет подключен к нему. В данном случае будет использоваться Raspberry Pi. В контексте текущего проекта более оптимальным будет LAN подключение (проводное), так ка WLAN (беспроводное) имеет меньшую отказоустойчивость и низкую стабильность подключения на непрямой видимости с беспроводным маршрутизатором.
2. Устройство, с которого будет осуществляться подключение к веб-приложению. В данном случае в роли этого устройства будет смартфон и персональный компьютер.

1.2 Выбор фреймворка

Конечный продукт должен иметь UI/UX дизайн, понятный любому пользователю и в том числе тому, у кого нет опыта взаимодействия с подобными технологиями. Поэтому для написания данного проекта мною был выбран фреймворк Flet.

Данный фреймворк наследует практически все компоненты из фреймворка Flutter (Flutter – фреймворк для написания нативных приложений под Android-устройства) для разработки кроссплатформенных приложений на языке Python. Отличие Flet от остальных схожих фреймворков заключается в том, что он не использует сторонние SDK и имеет встроенный веб-сервер, который позволяет отображать весь пользовательский интерфейс. Также Flet способен автоматически оптимизировать интерфейс на любом устройстве, поэтому благодаря этому фреймворку не требуется писать и оптимизировать код для разных устройств – на всех он будет выглядеть одинаково.

Помимо прочего данный фреймворк имеет очень простую в использовании систему отладки, которая позволяет максимально быстро запустить интерфейс на любом устройстве внутри текущей локальной сети (данная способность описана подробнее в п. 2.5)

1.3 Выбор среды разработки

В качестве среды разработки мною была выбрана JetBrains PyCharm 2023.1. Данная IDE (интегрированная среда разработки) отличается от остальных большим функционалом в следующих аспектах:

- непосредственная работа с написанием кода (подсказки при вводе, подсветка синтаксиса и т.д.);
- отладка (проверка) кода (устранение лишних блоков кода, которые утяжеляют его чтение, поиск и подсказки в исправлении багов и т.д.);
- способность работать с системой контроля версий (VCS, англ.: “Version Control System”) для того, чтобы иметь возможность восстановить код до предыдущей версии на случай, если текущая версия работает нестабильно;
- простота в работе с плагинами и модулями (загрузка всей необходимой документации и её представление в понятном виде).

Перед началом создания проекта VCS была синхронизирована с аккаунтом на GitHub, чтобы можно было получить доступ к работе над проектом не только с основного компьютера.

2 Процесс разработки

2.1 Описание требуемого функционала

Перед началом написания кода выделим функционал, который потребуется реализовать в готовом приложении. Функционал в нижеприведённом списке расположен в порядке своей значимости для конечного использования:

1. Управление текущим треком (воспроизведение, пауза, изменение громкости);
2. Синхронизация изменений, если приложение открыто на нескольких устройствах одновременно (к примеру, если на устройстве A изменяют громкость, то на устройстве B интерфейс должен обновиться и показывать новое значение громкости);
3. Автоматический переход к следующему треку после окончания предыдущего;
4. Выбор директории (папки), в который содержится список файлов для воспроизведения;
5. Ручной переход к следующему треку в любой момент времени;
6. Таймеры для автоматического начала и остановки воспроизведения в определённое время (например, включение музыки утром после подъёма или отключение музыки после отбоя);
7. Контроль доступа к интерфейсу управления (в случае несанкционированного получения доступа к приложению должен быть затребован пароль для верификации текущего пользователя).

2.2 Подбор модулей для функционирования проекта

Помимо самого фреймворка Flet понадобится некоторые другие модули, требуемые для работы с аудио, файловой системой и т.д.:

- asyncio – модуль, предоставляющий инфраструктуру для написания асинхронного кода, использующего сопрограммы.
- fnmatch – модуль, предоставляющий функции для сравнения строк с использованием шаблонов, аналогичных тем, что используются в командах оболочки Unix;
- json – модуль для работы с JSON (JavaScript Object Notation – текстовый формат обмена данными, основанный на JavaScript), который позволяет кодировать и декодировать данные в формате JSON;
- eyed3 – библиотека для работы с метаданными аудиофайлов (в данном случае для mp3). Позволяет извлекать информацию о треках, альбомах и т.д.;
- sounddevice – модуль для работы с звуковыми устройствами. Предоставляет функции для воспроизведения и записи звука.
- os – модуль для работы с операционной системой. Предоставляет функции для взаимодействия с файловой системой, выполнения команд в командной строке и т.д.;
- mixer из библиотеки pygame – модуль для управления аудио в играх и мультимедийных приложениях. pygame предоставляет функционал для работы с изображениями, звуками и событиями;
- schedule – модуль для удобного планирования выполнения задач в определенное время или с интервалами. Позволяет создавать периодические задачи.
- re – модуль для работы со строками, их делением и обработкой.

2.3 Общая структура кода

Код состоит из 3 частей: импортирование модулей и иных зависимостей, основная функция main() и параметры запуска приложения.

Все остальные функции содержатся внутри функции main() для большего понимания структуры кода и места, в котором требуется та или иная функция.

В самой функции main() созданы все объекты интерфейса, с которыми взаимодействует пользователь (текстовые надписи, кнопки, слайдеры и т.д.). Каждое из них имеет лаконичное название, позволяющее только по нему понять, что это за элемент и за что он отвечает.

Главным элементом, в который добавляются все остальные, является элемент page – это страница, с которой взаимодействует пользователь. В неё при запуске приложения добавляются все остальные дочерние элементы.

Так как было необходимо реализовать немалый функционал в приложении, разместить все элементы управления на одном экране было невозможно. Поэтому в проекте было использовано виртуализация нескольких страниц. Когда пользователь переходит с основного экрана на экран выбор папки или на экран управления таймерами, для него это выглядит как переход на другую страницу, но на самом деле лишь отключается видимость элементов «главного» экрана и включается видимость требуемого экрана. За это отвечает функция “.visible”, которая может принимать значения «True» или «False», то есть видимость включена или выключена соответственно.

2.4 Описание функций

- check_cable_connection() – уведомляет пользователя о том, чтобы он не забыл подключить кабель, если таковой не подключен к компьютеру.
- start_main_screen(e) – инициализирует и запускает главный экран.

- update_view(msg) – обновляет элементы всех экранов после получения обновления от другой активной копии приложения.
- explorer_changing(e) – позволяет перемещаться по каталогам во время выбора папки с файлами
- show_folders(e) – показывает обновления каталогов, вызывается предыдущей функцией.
- find_mp3_files(folder) – ищет в выбранном каталоге все файлы с расширением «mp3».
- after_folder_picked(e) – запускает предыдущую функцию и обновляет элементы.
- go_up_explorer(e) – позволяет переходить по дереву каталогов вверх.
- open_explorer_screen(e) – инициализирует и открывает экран выбора папки.
- continue_playing(e) – продолжает воспроизведение.
- pause_playing(e) – останавливает воспроизведение.
- next_track(e) – переключает очередь на следующий аудиофайл.
- change_volume(e) – меняет текущую громкость воспроизведения.
- previous_track(e) – переключает очередь на предыдущий аудиофайл.
- get_update(msg) –
- send_data() – отправляет обновления элементов на все активные копии приложения
- update_data() – обновляет конфигурационный файл «current_params.json»
- open_main_screen(e) – возвращает на главный экран с любого другого.
- open_timers_screen(e) – инициализирует и открывает экран работы с таймерами автоматического управления воспроизведением.

2.5 Запуск в режиме отладки

Фреймворк Flet позволяет запускать текущую версию приложения в режиме отладки. Это можно осуществить двумя вариантами:

1. Отладка в режиме оконного приложения. Достигается путём введения в терминал команды “flet main.py -r”, где main.py – запускаемый файл, а “-r” – аргумент терминала, указывающий на запуск приложения в режиме отладки. После данной команды открывается стандартное окно Windows с проектом.
2. Отладка в режиме сервиса, запущенного на определённом порту локального хоста. Данный вариант отличается от предыдущего тем, что открыть приложение можно на любом устройстве в локальной сети, зная IP-адрес компьютера и порт, на котором запускается приложение. Изначально этот вариант недоступен, так как брандмауэр Windows закрывает все порты кроме основных, но после некоторых манипуляций появляется возможность отключить блокировку изменения доступных порт и разрешить нужный.

3 Запуск проекта

Запуск финальной версии проекта будет проводиться на одноплатном компьютере Raspberry Pi со следующими тактико-техническими характеристиками:

- Система на чипе – Broadcom BCM2711 SoC;
- Центральный процессор – ARMv8 Cortex-A72 x64 (1,5 ГГц);
- Графический сопроцессор – VideoCored VI;
- Оперативная память – LPDDR4 SDRAM 8GB;
- Порты: Gigabit Ethernet, 2x USB 3.0, 2x USB 2.0, 2x micro HDMI;
- Модуль WI-FI – 2.4 ГГц и 5 ГГц IEEE 802.11.b/g/n/асж
- Модуль Bluetooth – Bluetooth 5.0 Low Energy (BLE).

Питается данный компьютер от USB-C кабеля, который подключен к источнику тока с выходным напряжением 5В и током от 2,5А до 3А.

3.1 Подготовка Raspberry Pi

Изначально плата Raspberry Pi не содержит внутри себя операционной системы, которая позволит загрузить и запустить проект. Поэтому для начала требуется установить операционную систему на внешний носитель, а после чего подключить его к Raspberry Pi.

В данном проекте Raspberry Pi функционирует на ОС Ubuntu Server 22.04.3 64-bit (Focal Fossa). Ubuntu Server – это основанная на ядре Linux операционная система для работы на серверах. Установка операционной системы осуществляется с помощью ПО Raspberry Pi Imager версии 1.7.5

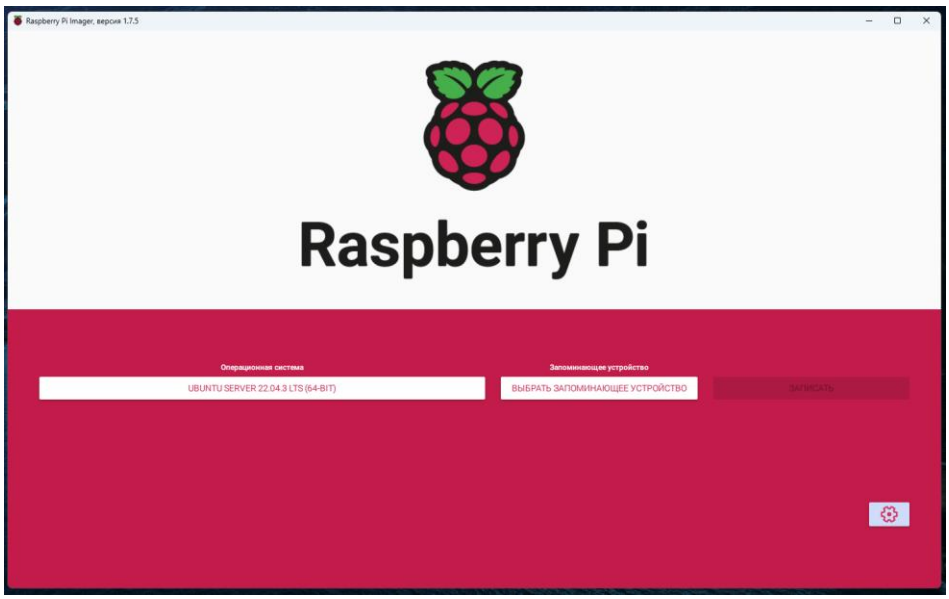

Рис. 3.1.1 – интерфейс главного окна Raspberry Pi Imager

После загрузки ОС на внешний носитель и его подключения к Raspberry Pi, последняя подключается в WI-FI, параметры которого были указаны при загрузке ОС (рис. 3.1.2).

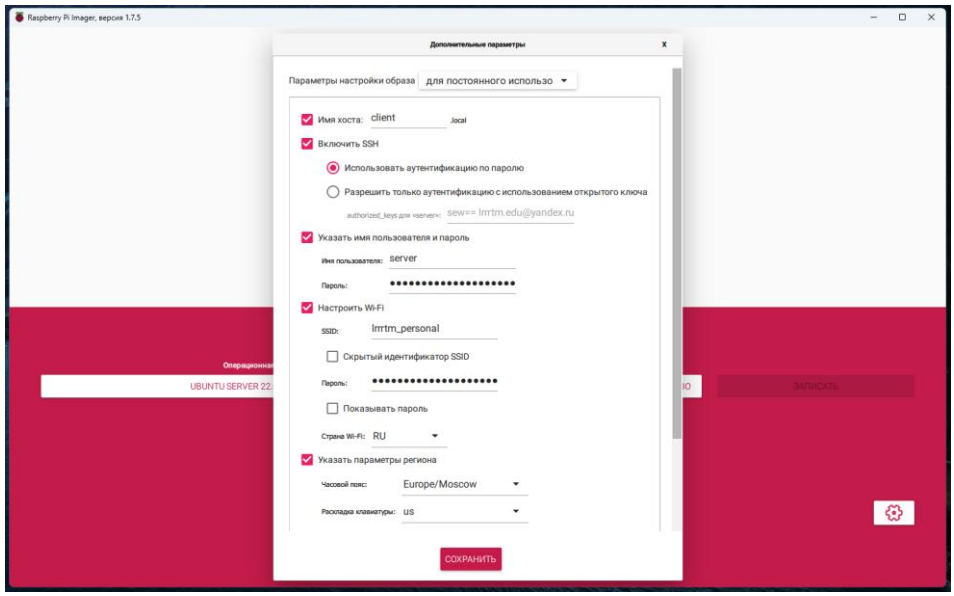

Рис. 3.1.2 – установка начальных параметров для установки ОС

Подключиться к компьютеру можно по протоколу SSH (защищенный сетевой протокол для удалённого управления сервером через глобальную или локальную сеть) с помощью, например, Windows PowerShell (рисунок 3.1.3).

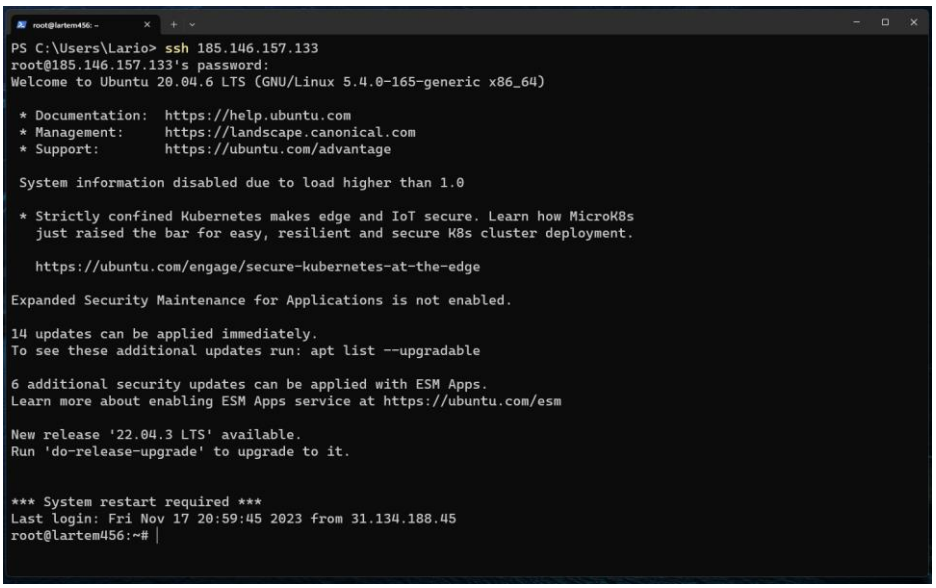

Рис. 3.1.3 – успешная авторизация на Raspberry Pi через Windows PowerShell

После запуска ОС необходимо загрузить, установить и настроить все модули, необходимые для корректной работы проекта. Были выполнены следующие манипуляции:

- Создана директория /home/server/projects/dcaudio/source для исходных файлов проекта, а также …/dcaudio/music для размещения каталогов с файлами mp3;
- Написаны скрипты “ngrok.service” и “flet.service” для автоматического запуска (см. п. 3.3) и перезагрузки проекта и ПО ngrok (об ngrok подробнее в п. 3.2) в случае достижения ошибки во время выполнения или перезагрузки компьютера. Содержание файлов описано в приложении №1;
- Установлены все модули Python, описанные в п. 2.2

3.2 Настройка туннеля ngrok

Чтобы иметь доступ к приложению не только из локальной, но и из глобальной сети, необходимо пробросить адрес компьютера в локальной сети и порт, на котором запущено приложение из локальной сети в глобальную. Данное действие было сделано с помощью ПО ngrok. Это

кроссплатформенное приложение, которое может создать URL-адрес туннелирования или пересылки, чтобы запросы к Интернету могли обращаться к локальному компьютеру.

После регистрации аккаунта и установки модуля ngrok на Raspberry Pi необходимо подключить свой аккаунт, введя команду “ngrok config add-authtoken <TOKEN>”, где <TOKEN> - уникальный код для каждого пользователя данного сервиса. После чего можно запускать туннель. Это делается с помощью команды “ngrok http <порт> --domain <домен> --basicauth <логин:пароль>”, в которой указывается порт приложения, домен и связка логин-пароль для авторизации (полный код для запуска находится в приложении №1)

3.3 Автоматизация запуска

Для повышения отказоустойчивости приложения в случае внеплановой остановки скрипта или перезагрузки компьютера требовалось автоматизировать процесс перезагрузки/перезапуска нужных систем. Операционная система Ubuntu, установленная на компьютере позволяет сделать автоматизацию.

Для этого требуется написать специальный сервисный файл с расширением .service, в котором будут содержаться параметры и ситуации, в которых требуется запустить или перезагрузить какой-либо сервис. Всё это работает благодаря двум утилитам, встроенным в ОС: systemctl (утилита в операционной системе Linux, которая используется для управления службами или демонами, рис. 3.3.1) и systemd (инструмент для управления процессами и службами).

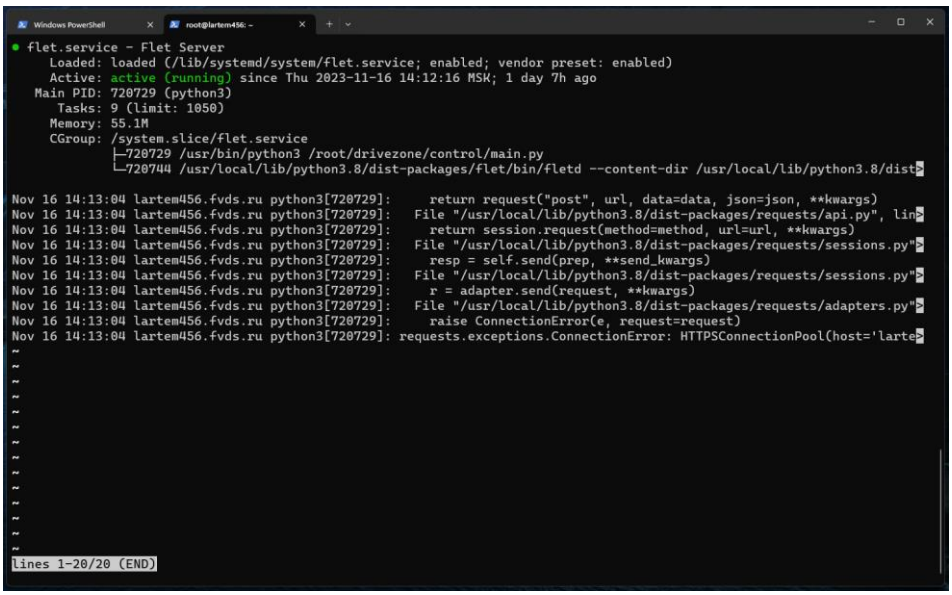

Рис. 3.3.1 – интерфейс утилиты systemctl с информацией о процессе flet.service

3.4 Структура проекта

Основная часть проекта находится в каталоге dcaudio, внутри него имеется два подкаталога: source – каталог, в котором размещены основной исполняемый файл main.py, а также два json-файла, отвечающих за хранение основной информации о текущем состоянии системы и о состоянии таймеров автоматического управления воспроизведением. На рисунке 3.4.1 показано структура проекта в виде «дерева».

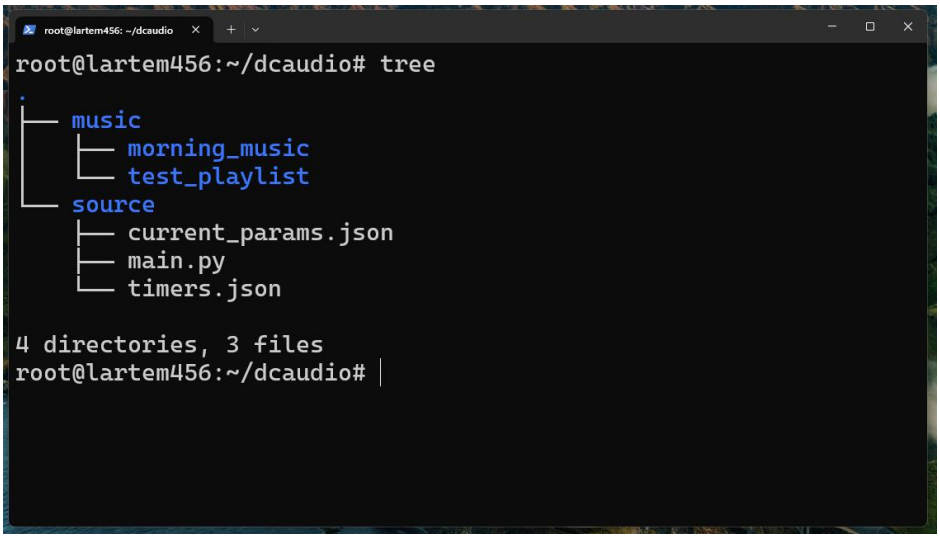

Рис. 3.4.1 – структура проекта

4 Заключение

В результате работы было создано веб-приложение «DCAudio» (скриншоты интерфейса показаны в приложении 2), отвечающее всем целям, которые были выставлены в начале работы над проектом.

Веб-приложение тестировалось на протяжении длительного времени, и никаких багов и ошибок замечено не было, все элементы удалённого управления аудиопотоком работали штатно и выдавали ожидаемый результат. Также был проведён опрос о взаимодействии конечного пользователя с интерфейсом приложения среди моего круга общения. Никаких «неудобных» решений в интерфейсе приложения по результатам обработки данного опроса обнаружено не было, отмечается интуитивно понятный дизайн и расположение элементов управления (кнопки, слайдеры, переключатели и т.д.)

Данное веб-приложение также планируется использовать и в реальных условиях на базе Центра Развития Одарённых Детей в калининградской области.

Список использованных источников

1. Flet Tutorials [Электронный ресурс]. - Appveyor Systems Inc., 2023. - Режим доступа: https://flet.dev/docs/. (Дата обращения: 12.11.2023)
2. OS – Miscellaneous operating system interfaces [Электронный ресурс]. - Python Software Foundation, 2023. - Режим доступа: https://docs.python.org/3/library/os.html. (Дата обращения: 12.11.2023)
3. Holzer, Raphel. Pygame tutorial Documentation / Raphel Holzer. - Release, 2021. - 120 с. (Дата обращения: 12.11.2023)

Приложение 1

Исходные коды

Сервисный файл “flet.service”

```
[Unit]
Description=ngrok
After=network.target
After=syslog.target
[Service]
Type=simple
ExecStart=python3 main.py
WorkingDirectory=/home/server/projects/dcaudio/source
Restart=always
RestartSec=1
User=server
Group=server
StartLimitBurst=99999
StartLimitInterval=999999
[Install]
WantedBy=multi-user.target
```

Сервисный файл “ngrok.service”

```
[Unit]
Description=ngrok
After=network.target
After=syslog.target
[Service]
Type=simple
ExecStart=/snap/bin/ngrok http 8502 --basic-auth
"user:password" --domain evolving-hedgehog-wholly.ngrok-
free.app
Restart=always
RestartSec=1
User=server
Group=server
StartLimitBurst=99999
StartLimitInterval=999999
[Install]
WantedBy=multi-user.target
```

Файл “main.py”

```
# Проект DCAudio - веб-приложение для удалённого управления
воспроизведением аудио-файлов
# Артём Ларионенко (t.me/lrrrtm)
# 5132704/20001

import asyncio
import fnmatch
import json
import eyed3
import flet as ft
import sounddevice as sd
import os
from pygame import mixer
import schedule
import re


async def main(page: ft.Page):
# ----------------------- ГЛОБАЛЬНЫЕ ПАРАМЕТРЫ ---------------------
-

    global folder_now
    global CUR_PLAYLIST
    global CUR_TRACK_ID

    CUR_PLAYLIST = []
    CUR_TRACK_ID = -1
    folder_now = "C:/Users/Lario/OneDrive/Рабочий стол/music_open"
    page.title = "CROD Audio"
    mixer.init()

# ----------------------------- ФУНКЦИИ ----------------------------
-

    async def check_cable_connection():
        devices = sd.query_devices()
        for device in devices:
            if "analog" in device["name"].lower() and "output" in
device["name"].lower():
                return True
        return False

    async def start_main_screen(e):
        try:
            with open("current_params.json", "r") as file:
                data = file.readline()
                data = json.loads(data)
                await update_view(data)
        except Exception:
```

```
            pass
        current_playlist_col.visible = True
        track_name_artist_col.visible = True
        control_buttons_row.visible = True
        main_screen_btns_row.visible = True
        check_connection_col.visible = False
        volume_control_row.visible = True
        timer_btn.visible = True
        to_main_screen_btn.visible = False
        page.appbar.title = ft.Text("Основной экран")
        await page.update_async()

    async def update_view(msg):
        global CUR_PLAYLIST, CUR_TRACK_ID, folder_now
        if msg['status'] != "":
            volume_slider.value = msg['current_volume']
            folder_now = msg['current_folder']
            CUR_TRACK_ID = msg['current_track_id']
            CUR_PLAYLIST = msg['current_playlist']
            track_name_artist_text.value = msg['track_name']
            current_status_btn.text = msg['status']
            pick_playlist_btn.text = re.split(r'//|/|\\',
folder_now)[-1]
            current_playlist_col.controls.clear()

            for cur_track in CUR_PLAYLIST:
                audio = eyed3.load(cur_track)
                try:
                   track_name = ft.Text(f"{audio.tag.title} -
{audio.tag.artist}")
                    if track_name is None:
                        track_name = ft.Text(re.split(r'//|/|\\',
cur_track)[-1])
                except AttributeError:
                    track_name = ft.Text(re.split(r'//|/|\\',
cur_track)[-1])
                current_playlist_col.controls.append(track_name)
        page.snack_bar = ft.SnackBar(
            content=ft.Text(f"Подключение выполнено успешно!"))
        page.snack_bar.open = True

    async def explorer_changing(e):
        global folder_now
        try:
            os.listdir(folder_now + f"/{e.control.value}")
            folder_now += f"/{e.control.value}"
            source_text.value = f"Путь: {folder_now}".replace("//",
"/")
            await page.update_async()
            await show_folders("e")
        except PermissionError:
```

```
            page.snack_bar = ft.SnackBar(
                content=ft.Text("К данной директории нет доступа,
выберите другую директорию"))
            page.snack_bar.open = True
            await page.update_async()

    async def show_folders(e):
        all_list = os.listdir(folder_now)
        folders_list = [folder for folder in all_list if
os.path.isdir(os.path.join(folder_now, folder))]
        col_list = ft.Column(height=400)

        for folder in folders_list:
            col_list.controls.append(
                ft.Radio(value=f"/{folder}",
                         label=folder)
            )
        folders_explorer.content = col_list
        await page.update_async()

    async def find_mp3_files(folder):
        mp3_files = []
        for root, dirs, files in os.walk(folder):
            for file in fnmatch.filter(files, '*.mp3'):
                mp3_files.append(os.path.join(root, file))
        return mp3_files

    async def after_folder_picked(e):
        global CUR_PLAYLIST
        CUR_PLAYLIST = await find_mp3_files(folder_now)
        if len(CUR_PLAYLIST) > 0:
            await open_main_screen("e")
            pick_playlist_btn.text = re.split(r'//|/|\\',
folder_now)[-1]
            await send_data()
            await update_data()
            page.snack_bar = ft.SnackBar(
                content=ft.Text(f"Обнаружено треков:
{len(CUR_PLAYLIST)}")
            )
            page.snack_bar.open = True
            track_name_artist_text.value = f"Нажмите на клавишу
переключения, чтобы запустить проигрывание"
            current_playlist_col.controls.clear()

            for cur_track in CUR_PLAYLIST:
                audio = eyed3.load(cur_track)
                try:
                    track_name = ft.Text(f"{audio.tag.title} -
{audio.tag.artist}")
                    if track_name is None:
```

```
                        track_name = ft.Text(re.split(r'//|/|\\',
cur_track)[-1])
                except AttributeError:
                    track_name = ft.Text(re.split(r'//|/|\\',
cur_track)[-1])
                current_playlist_col.controls.append(track_name)
        else:
            page.snack_bar = ft.SnackBar(
                content=ft.Text("В данной директории нет файлов mp3,
выберите другую директорию"))
            page.snack_bar.open = True
        await page.update_async()

    async def go_up_explorer(e):
        global folder_now
        folder_now = "/".join(folder_now.split("/")[:-2])
        source_text.value = f"Путь: {folder_now}".replace("//", "/")
        await page.update_async()
        await show_folders("e")

    async def open_explorer_screen(e):
        volume_control_row.visible = False
        current_playlist_col.visible = False
        track_name_artist_col.visible = False
        control_buttons_row.visible = False
        main_screen_btns_row.visible = False
        folders_explorer_col.visible = True
        pick_folder_text.visible = True
        source_text.visible = True
        explorer_btns_row.visible = True
        timer_btn.visible = False
        page.appbar.leading = to_main_screen_btn
        to_main_screen_btn.visible = True
        page.appbar.title = ft.Text("Выбор папки")
        await show_folders("e")
        await page.update_async()

    async def continue_playing(e):
        mixer.music.unpause()
        current_status_btn.text = "Играет"
        current_status_btn.bgcolor = ft.colors.GREEN
        await send_data()
        await page.update_async()
        await update_data()

    async def pause_playing(e):
        mixer.music.pause()
        current_status_btn.text = "Пауза"
        current_status_btn.bgcolor = ft.colors.YELLOW
        await send_data()
        await page.update_async()
```

```
        await update_data()

    async def next_track(e):
        global CUR_PLAYLIST, CUR_TRACK_ID
        if CUR_TRACK_ID == len(CUR_PLAYLIST) - 1:
            CUR_TRACK_ID = -1
        mixer.music.load(CUR_PLAYLIST[CUR_TRACK_ID + 1])
        mixer.music.set_volume(volume_slider.value / 100)
        CUR_TRACK_ID += 1
        audio_info = eyed3.load(CUR_PLAYLIST[CUR_TRACK_ID])
        try:
            track_name_artist_text.value = f"{audio_info.tag.title} -
{audio_info.tag.artist}"
        except AttributeError:
            track_name_artist_text.value =
f"{CUR_PLAYLIST[CUR_TRACK_ID].split('/')[-1]}"
        mixer.music.play(fade_ms=1000)
        current_status_btn.text = "Играет"
        current_status_btn.bgcolor = ft.colors.GREEN

        await update_data()
        await send_data()
        await page.update_async()

    async def change_volume(e):
        volume = volume_slider.value / 100
        print(f"VOLUME CHANGED TO {volume}")
        mixer.music.set_volume(volume)
        await update_data()
        await send_data()

    async def previous_track(e):
        global CUR_PLAYLIST, CUR_TRACK_ID
        if CUR_TRACK_ID == 0:
            CUR_TRACK_ID = len(CUR_PLAYLIST)
        mixer.music.load(CUR_PLAYLIST[CUR_TRACK_ID - 1])
        CUR_TRACK_ID -= 1
        audio_info = eyed3.load(CUR_PLAYLIST[CUR_TRACK_ID])
        track_name_artist_text.value = f"{audio_info.tag.title} -
{audio_info.tag.artist}"
        mixer.music.play(fade_ms=1000)
        current_status_btn.text = "Играет"
        current_status_btn.bgcolor = ft.colors.GREEN

        await send_data()
        await update_data()
        await page.update_async()

    async def get_update(msg):
        print("RECIEVED UPDATE")
        global CUR_PLAYLIST, CUR_TRACK_ID, folder_now
```

```
        volume_slider.value = msg['current_volume']
        folder_now = msg['current_folder']
        CUR_TRACK_ID = msg['current_track_id']
        CUR_PLAYLIST = msg['current_playlist']
        track_name_artist_text.value = msg['track_name']
        player_status = msg['status']

        current_status_btn.text = player_status
        pick_playlist_btn.text = folder_now.split("/")[-1]
        current_playlist_col.controls.clear()
        for cur_track in CUR_PLAYLIST:
            audio = eyed3.load(cur_track)
            try:
                track_name = ft.Text(f"{audio.tag.title} -
{audio.tag.artist}")
                if track_name is None:
                    track_name = ft.Text(re.split(r'//|/|\\',
cur_track)[-1])
            except AttributeError:
                track_name = ft.Text(re.split(r'//|/|\\', cur_track)[-
1])
            #track_name = ft.Text(f"{audio.tag.title} -
{audio.tag.artist}")
            current_playlist_col.controls.append(track_name)
        await page.update_async()
    await page.pubsub.subscribe_async(get_update)

    async def send_data():
        print("UPDATE SENT")
        data = {
            "current_folder": folder_now,
            "current_track_id": CUR_TRACK_ID,
            "current_playlist": CUR_PLAYLIST,
            "current_volume": volume_slider.value,
            "track_name": track_name_artist_text.value,
            "status": current_status_btn.text

        }
        if data['current_track_id'] != -1:
            await page.pubsub.send_others_async(data)

    async def update_data():
        data = {
            "current_folder": folder_now,
            "current_track_id": CUR_TRACK_ID,
            "current_playlist": CUR_PLAYLIST,
            "current_volume": volume_slider.value,
            "track_name": track_name_artist_text.value,
            "status": current_status_btn.text
```

```
    }
    jsonStr = json.dumps(data)
    with open("current_params.json", "w") as file:
        file.write(jsonStr)

async def open_main_screen(e):
    volume_control_row.visible = True
    current_playlist_col.visible = True
    track_name_artist_col.visible = True
    control_buttons_row.visible = True
    main_screen_btns_row.visible = True
    folders_explorer_col.visible = False
    pick_folder_text.visible = False
    source_text.visible = False
    explorer_btns_row.visible = False
    timer_btn.visible = True
    to_main_screen_btn.visible = False
    add_timer_btn.visible = False
    timers_list_col.visible = False
    page.appbar.title = ft.Text("Основной экран")

    await page.update_async()

def stop_music_by_schedule():
    mixer.music.pause()

def start_music_by_schedule():
    mixer.music.play(fade_ms=1000)

async def add_timer(time, task):
    schedule.every().day.at(time).do(task)

async def remove_timer(task):
    schedule.cancel_job(task)

async def open_timers_screen(e):
    page.appbar.title = ft.Text("Таймеры")
    timer_btn.visible = False
    volume_control_row.visible = False
    current_playlist_col.visible = False
    track_name_artist_col.visible = False
    control_buttons_row.visible = False
    main_screen_btns_row.visible = False
    timer_btn.visible = False
    add_timer_btn.visible = True
    timers_list_col.visible = True
    to_main_screen_btn.visible = True
    page.appbar.leading = to_main_screen_btn

    with open("timers.json", "r") as file:
        timers = file.readline()
```

```
            timers = json.loads(timers)['data']
            timers_list_col.controls.clear()
            for timer in timers:
                timers_list_col.controls.append(
                    ft.Card(
                        content=ft.Row(
                            [
                                ft.Container(
                                    ft.Column(
                                        [
                                            ft.Text(timer['time'],
size=24),
                                            ft.Text(f"Действие:
{timer['task']}"),
                                        ],
                                        alignment="center",
                                    ),
                                    margin=ft.margin.only(right=30)
                                ),

ft.IconButton(icon=ft.icons.EDIT_ROUNDED),

ft.IconButton(icon=ft.icons.DELETE_ROUNDED),
                                ft.Switch(value=bool(timer['status']))
                            ],
                            alignment="center"
                        ),
                        width=400,
                        height=130
                    )
                )

        await page.update_async()

# ------------------ СОЗДАНИЕ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ ------------------

    to_main_screen_btn = ft.IconButton(
        icon=ft.icons.ARROW_BACK_ROUNDED,
        visible=False,
        on_click=open_main_screen
    )
    timer_btn = ft.FilledTonalButton(
        icon=ft.icons.TIMER_ROUNDED,
        visible=False,
        on_click=open_timers_screen,
        text="Таймеры"
    )
    add_timer_btn = ft.FilledTonalButton(
        icon=ft.icons.ADD_ALARM_ROUNDED,
        visible=False,
        text="Создать"
```

```
    )
    page.appbar = ft.AppBar(
        title=ft.Text("Инциализация"),
        bgcolor=ft.colors.SURFACE_VARIANT,
        actions=[timer_btn, add_timer_btn]
    )

    track_name_artist_text = ft.Text("Папка с музыкой не выбрана,
воспроизведение невозможно")

    check_connection_text = ft.Text("Для инициализации нажмите на
кнопку")

    check_connection_btn = ft.OutlinedButton(
        text="Инициализироваться",
        icon=ft.icons.CABLE_ROUNDED,
        on_click=start_main_screen,
        height=50,
        width=300
    )
    timers_list_col = ft.Column(
        visible=False,
        alignment="center",
        scroll=ft.ScrollMode.ADAPTIVE,
        expand=True
    )
    check_connection_col = ft.Column(
        controls=[
            check_connection_text,
            check_connection_btn
        ],
        visible=False,
        alignment="center",
    )
    track_name_artist_col = ft.Column(
        controls=[
            track_name_artist_text
        ],
        alignment="center",
        visible=False
    )
    control_buttons_row = ft.Row(
        controls=[
            ft.IconButton(icon=ft.icons.SKIP_PREVIOUS_ROUNDED,
                          icon_size=50,
           on_click=previous_track
                          ),
            ft.IconButton(icon=ft.icons.PLAY_CIRCLE_FILL_ROUNDED,
                          icon_size=62,
                          on_click=continue_playing
                          ),
```

```
            ft.IconButton(icon=ft.icons.PAUSE_CIRCLE_FILLED_ROUNDED,
                          icon_size=62,
                          on_click=pause_playing
                          ),
            ft.IconButton(icon=ft.icons.SKIP_NEXT_ROUNDED,
                          icon_size=50,
                          on_click=next_track
                          ),

        ],
        alignment="center",
        visible=False
    )
    pick_playlist_btn = ft.FilledButton(
        text="Выбор папки",
        visible=True,
        on_click=open_explorer_screen,
        height=50, width=150, icon=ft.icons.FOLDER_OPEN_ROUNDED
    )
    current_status_btn = ft.FilledButton(
        text="Не запущено",
        disabled=True,
        height=50,
        width=150
    )
    main_screen_btns_row = ft.Row(
        [
            pick_playlist_btn,
            current_status_btn
        ],
        visible=False,
        alignment="center"
    )
    pick_folder_text = ft.Text("Выберите папку, в которой находится
плейлист",
                           visible=False
                           )
    source_text = ft.Text(f"Путь: {folder_now}",
                          visible=False
                          )

    folders_explorer = ft.RadioGroup(
        on_change=explorer_changing,
    )
    pick_explorer_btn = ft.FilledButton(text="Выбрать",
                                on_click=after_folder_picked,
                                icon=ft.icons.FOLDER_OPEN_ROUNDED,
                                height=50,
                                width=200
                                )
    pick_explorer_btn.bgcolor = ft.colors.GREEN_ACCENT
```

```
up_explorer_btn = ft.IconButton(on_click=go_up_explorer,
                                icon=ft.icons.ARROW_UPWARD_ROUNDED,
                                height=50,
                                bgcolor=ft.colors.SURFACE_VARIANT)
explorer_btns_row = ft.Row(
    [
        up_explorer_btn,
        pick_explorer_btn
    ],
    visible=False,
    alignment="center"
)

folders_explorer_col = ft.Column(
    [
        folders_explorer,
    ],
    alignment="center",
    visible=False,
    scroll=ft.ScrollMode.ADAPTIVE,
    expand=True
)

current_playlist_col = ft.Column(
    visible=False,
    expand=True,
    spacing=20,
    scroll=ft.ScrollMode.ADAPTIVE
)

volume_slider = ft.Slider(
    min=0,
    max=100,
    divisions=100,
    label="{value}%",
    on_change=change_volume,
    value=50,
    expand=True
)

volume_control_row = ft.Row(
    [
        ft.IconButton(
            icon=ft.icons.VOLUME_UP_ROUNDED,
            icon_size=24,
            disabled=True
        ),
        volume_slider
    ],
    alignment="center",
```

```
        visible=False,
    )

    if await check_cable_connection():
        track_name_artist_col.visible = True
        control_buttons_row.visible = True
        volume_control_row.visible = True
        main_screen_btns_row.visible = True
        await page.update_async()
    else:
        check_connection_col.visible = True
        await page.update_async()

    await page.add_async(
        current_playlist_col,
        pick_folder_text,
        source_text,
        track_name_artist_col,
        control_buttons_row,
        main_screen_btns_row,
        check_connection_col,
        folders_explorer_col,
        explorer_btns_row,
        volume_control_row,
        timers_list_col
    )

    page.vertical_alignment = "center"
    page.horizontal_alignment = "center"

    await page.update_async()

    while True:
        schedule.run_pending()
        await asyncio.sleep(1)

# ------------------------ ПАРАМЕТРЫ ЗАПУСКА ------------------------

DEFAULT_FLET_PATH = ''
DEFAULT_FLET_PORT = 8502

if __name__ == "__main__":
    flet_path = os.getenv("FLET_PATH", DEFAULT_FLET_PATH)
    flet_port = int(os.getenv("FLET_PORT", DEFAULT_FLET_PORT))
    ft.app(name=flet_path, target=main, view=None, port=flet_port)
```

Ознакомиться с проектом также можно на GitHub: https://clck.ru/36fRL8

Приложение 2

Скриншоты интерфейса (светлая и тёмная темы)

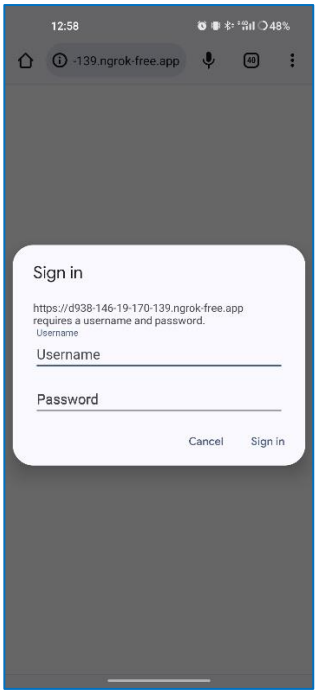

Рис. 1 – экран аутентификации

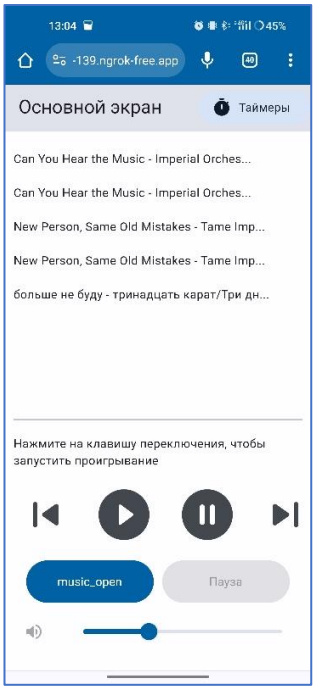

Рис. 2 – основной экран

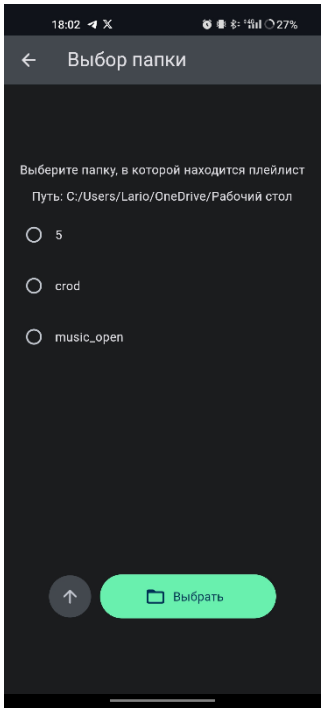

Рис. 3 – экран выбора папки

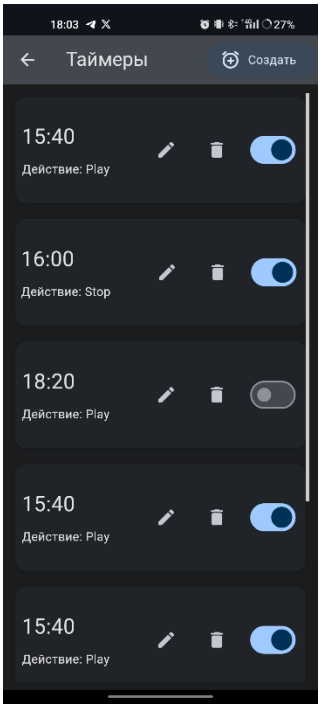

Рис. 4 – экран управления таймерами