APEM Projekt - Dokumentacja

Rafał Kutnik **325296**, Jakub Walasek **325330**

Maj 2025

Projekt dostępny jest w repozytorium pod adresem: github.com/Elmonus/APEM-Projekt

Spis treści

1 Cel i role

1.1 Opis projektu

Celem niniejszego projektu jest zaprojektowanie oraz zaimplementowanie wieloplatformowej aplikacji do edycji multimediów, ze szczególnym uwzględnieniem operacji na plikach audio. Główna funkcjonalność aplikacji koncentruje się na przycinaniu oraz konwersji formatów dźwiękowych, co umożliwia użytkownikom szybkie i intuicyjne przetwarzanie nagrań dźwiękowych. Aplikacja ma nazwę *"Audiowerter"*.

Projekt zakłada stworzenie trzech wersji aplikacji:

- **wersji webowej** (przeglądarkowej),
- **wersji desktopowej** (dla systemów Windows i macOS),
- **wersji mobilnej** (dla systemów iOS oraz Android).

Proces implementacji rozpoczął się od opracowania wersji webowej, która stanowiła bazę do dalszego rozwoju pozostałych platform. Na jej podstawie powstała wersja desktopowa, zrealizowana przy użyciu frameworka Electron, oraz wersja mobilna, opracowana w oparciu o Ionic Framework, rozwiązanie umożliwiające tworzenie aplikacji mobilnych z wykorzystaniem technologii webowych.

1.2 Podział zadań

W ramach zespołu projektowego:

- **Jakub Walasek** odpowiadał za implementację wersji webowej i desktopowej aplikacji.
- **Rafał Kutnik** był odpowiedzialny za przygotowanie mobilnej i desktopowej wersji aplikacji.

2 Aplikacja Webowa

2.1 Założenia

Aplikacja webowa została zaprojektowana jako narzędzie do podstawowej edycji plików dźwiękowych, czyli przycinania nagrań oraz konwersji między różnymi formatami audio. Zrealizowana została w technologii Python (Flask) z użyciem HTML jako warstwy frontendowej.

Główne założenia funkcjonalne i techniczne:

- **Lokalne przetwarzanie plików dźwiękowych** – aplikacja umożliwia wgrywanie plików z dysku użytkownika i ich obróbkę.
- **Obsługa wielu popularnych formatów audio** – system wspiera import i eksport formatów takich jak na przykład: `MP3`, `WAV`, `FLAC` lub `OGG`.
- **Przycinanie plików audio** – aplikacja pozwala na zdefiniowanie zakresu czasowego (start i koniec) w sekundach, umożliwiając precyzyjne wycięcie fragmentu nagrania.
- **Interfejs HTML** – interfejs użytkownika ładowany jest z lokalnego pliku HTML, co umożliwia jego łatwą edycję i integrację z systemem Flask jako backendem.
- **Weryfikacja danych wejściowych** – wbudowana walidacja plików uwzględnia:
  - dozwolone rozszerzenia plików,
  - maksymalny rozmiar pliku (50 MB).
- **Wieloplatformowa kompatybilność** – dzięki wykorzystaniu HTML oraz Pythona (Flask + Pydub), aplikacja może działać lokalnie w systemach przez przeglądarkę.
- **Monitorowanie stanu aplikacji** – wbudowany endpoint `/health` pozwala na sprawdzenie, czy aplikacja działa poprawnie, co wspiera jej stabilność i możliwość automatycznego monitorowania.

2.2 Użyte narzędzia

Projekt aplikacji webowej opiera się na zestawie technologii open-source. Przedstawiono narzędzia wykorzystane na różnych etapach tworzenia i uruchamiania aplikacji.

- **Python 3** – główny język programowania backendu. Pozwala na szybkie tworzenie aplikacji serwerowych, przetwarzanie danych oraz integrację z zewnętrznymi bibliotekami.
- **Flask** – mikroframework webowy dla Pythona, odpowiedzialny za obsługę zapytań HTTP, routing, oraz uruchamianie lokalnego serwera aplikacji. Umożliwia integrację z szablonami HTML.
- **Flask-CORS** – rozszerzenie frameworka Flask, które umożliwia obsługę zapytań typu Cross-Origin Resource Sharing (CORS). Jest to niezbędne przy komunikacji między frontendem a backendem.
- **Pydub** – biblioteka do przetwarzania plików audio w Pythonie. Umożliwia m.in. konwersję między formatami, cięcie ścieżek dźwiękowych oraz eksport do różnych typów plików. Wymaga zewnętrznego narzędzia `ffmpeg` do działania.
- **ffmpeg** – zewnętrzne narzędzie wykorzystywane przez bibliotekę Pydub do obsługi konwersji i operacji na plikach audio. Musi być zainstalowane w systemie i dostępne w ścieżce systemowej.
- **HTML5** – język znaczników używany do budowy interfejsu użytkownika aplikacji webowej. Plik `index.html` zawiera strukturę strony oraz formularze do wgrywania i konfigurowania operacji na plikach audio.

2.3 Wymagania

Aby uruchomić aplikację webową lokalnie, użytkownik musi zadbać o odpowiednie przygotowanie środowiska. Poniżej przedstawiono wszystkie wymagane komponenty.

- **Komputer z systemem operacyjnym**
  - Windows 10/11,
  - MacOs 10.15+,
  - dowolna dystrybucja Linux z obsługą Pythona i ffmpeg.
- **Python 3.9 lub nowszy**
- **biblioteki Python** do obsługi serwerów i plików audio:
  - flask,
  - flask-cors,
  - pydub.
- **ffmpeg** () - zewnętrzne narzędzie do konwersji plików audio
- **przeglądarka z obsługą HTML5**
- **mikrofon** do ewentualnych nagrań

2.4 Instalacja i Uruchamianie

Poniżej przedstawiono wszystkie potrzebne kroki do instalacji potrzebnych komponentów:

1. Utworzenie folderu z potrzebnymi komponentami. Wszystkie kolejne kroki powinny być wykonywane w tym folderze.
2. Pobranie i umieszczenie w nim *app.py* oraz *index.html* z repozytorium projektu.
3. Instalacja Pythona 3.9 lub nowszego. Służy jako środowisko uruchomieniowe aplikacji backendowej. Zaleca się instalację poprzez:
   - macOS: `brew install python3` w terminalu
   - Windows: instalator z `https://www.python.org/downloads/`
   - Linux: `sudo apt install python3` w terminalu
4. Instalacja potrzebnych bibliotek Pythona przez menedżer pakietów **pip** (instalowany domyślnie razem z Pythonem). Umożliwia instalację wymaganych bibliotek przez terminal/cmd:

```
pip install flask flask-cors pydub
```

5. Instalacja **ffmpeg** dla różnych systemów operacyjnych:
   - macOS: `brew install ffmpeg` w terminalu
   - Windows: instalator z `https://ffmpeg.org/download.html`, dodać ścieżkę do `ffmpeg.exe` do zmiennej środowiskowej `PATH` w kodzie *app.py*
   - Linux: `sudo apt install ffmpeg` w terminalu
6. Instalacja przeglądarki nie jest konieczna ponieważ wszystkie nowoczesne systemy operacyjne mają takową już domyślnie zainstalowaną
7. Uruchomienie samej aplikacji przez wpisanie w terminal/cmd:

   ```
   python app.py
   ```

8. Po poprawnym uruchomieniu, wpisać ten konkretny adres w przeglądarce:

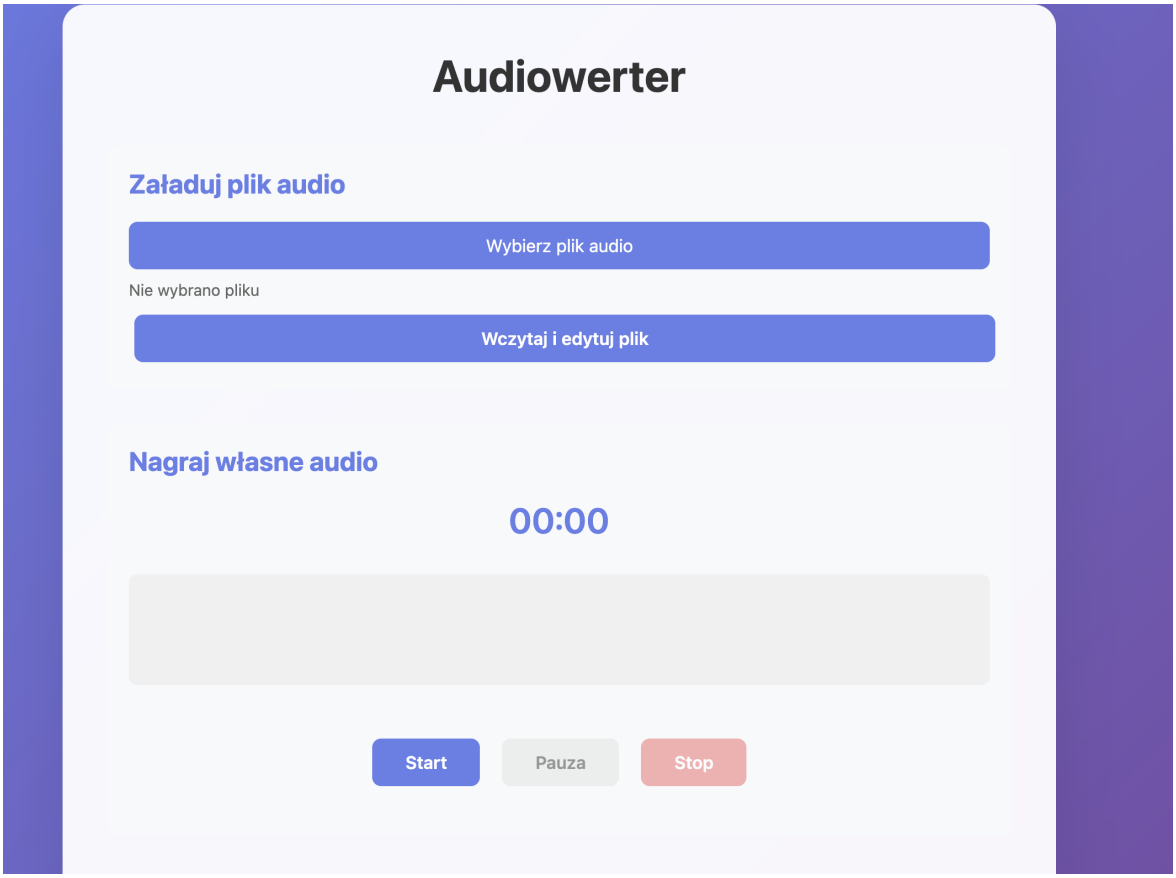
   ```
   http://127.0.0.1:5000
   ```

   lub

   ```
   http://localhost:5000
   ```

2.5 Użytkowanie

Po uruchomieniu aplikacji użytkownikowi zostaje wyświetlony poniższy interfejs graficzny umożliwiający przesyłanie oraz przetwarzanie plików audio.



Rysunek 1: Interfejs

Następnie prezentuje użytkownikowi opcje załadowania gotowego już pliku w formacie:

- mp3
- FLAC
- OGG
- WAV
- WEBM

Użytkownik może też nagrać własne audio, przy użyciu interfejsu umieszczonego pod opcją załadowania pliku audio. Jest możliwość zatrzymania nagrania i ponownego jego kontynuowania, przy użyciu odpowiednich przycisków.

Po załadowaniu pliku obojętną metodą, ukaże się opcja edycji audio, która wygląda następująco:

Rysunek 2: Interfejs edycji

Na górze pojawia się instrukcja, która szybko tłumaczy jak obsłużyć cały proces cięcia audio. Pod nim widać "waveform" pliku audio, który można edytować. Jest on niezbędny do łatwego cięcia.

Kolejno są opcje przybliżenia do dokładniejszych cięć oraz przycisk reset. Następnie znajdują się suwaki do właściwego cięcia, opcje odtwarzania danego fragmentu, informacje o starcie i zakończeniu fragmentu. Przycisk "Zastosuj" służy do zmiany początku i końca wybranej części.

Na samym końcu można wybrać format audio i ostateczne zapisanie.

2.6 Proces realizacji, problemy i ewentualne rozwiązania

Pierwsza wersja aplikacji pozwalała tylko nagrywania i wgrywania audio. Pliki audio były dodawane do folderu "Uploads", więc pliki były zapisywane lokalnie na komputerze i nie były usuwane.

W ostatecznej wersji plik html został mocno rozbudowany przez: dodanie przycinania, estetyczny interfesj oraz możliwość konwersji audio. Problemem w konwersji audio były pliki które nie były w formacie "wav". Trzeba było skorzystać z zewnętrznego narzędzia "ffmpeg". Dodatkowo funkcja przybliżenia zawsze były przybliżane na początku dźwięku. Podczas nagrywania audio, przycisk stop nie zatrzymuje czasu tylko wycisza mikrofon co nie jest optymalne rozwiązanie. Czasami z nieokreślonych przyczyn podczas nagrywania, wizualizacja dźwięku wyświetla się jako gruba linia.

3 Aplikacja Desktopowa

3.1 Założenia

Wersja desktopowa została przygotowana za pomocą narzędzia Electron ()electronjs.org, która umożliwia proste realizacje natywne programy na komputery osobiste do różnych systemów operacyjnych. Jako frontend może być używany ten plik html. Funkcjonalność ma być taka sama jak przy rozwiązaniu webowych. Python teoretycznie może zostać użyty jako backend ale dla lepszej kompatybilności zostanie użyty JavaScript.

3.2 Użyte narzędzia

- **Electron** - framework umożliwiający tworzenie aplikacji desktopowych za pomocą technologii webowych (HTML, CSS, JavaScript), łącząc Chromium i Node.js w jednej platformie.
- **Node.js** - *runtime environment* umożliwiający działanie JavaScript po stronie serwerowej. Nasza aplikacja nie będzie komunikować się z zewnętrznym serwerem, przetwarzanie będzie dokonywane lokalnie.
- **npm** - menadżer pakietów dla Node.js.
- **React** - biblioteka do budowania interfejsów i tworzenia skryptów.

3.3 Wymagania

- **Komputer z systemem operacyjnym**:
  - Windows 10/11,
  - MacOs 10.15+,
  - dowolna dystrybucja Linux z obsługą Note.js
- **Mikrofon**

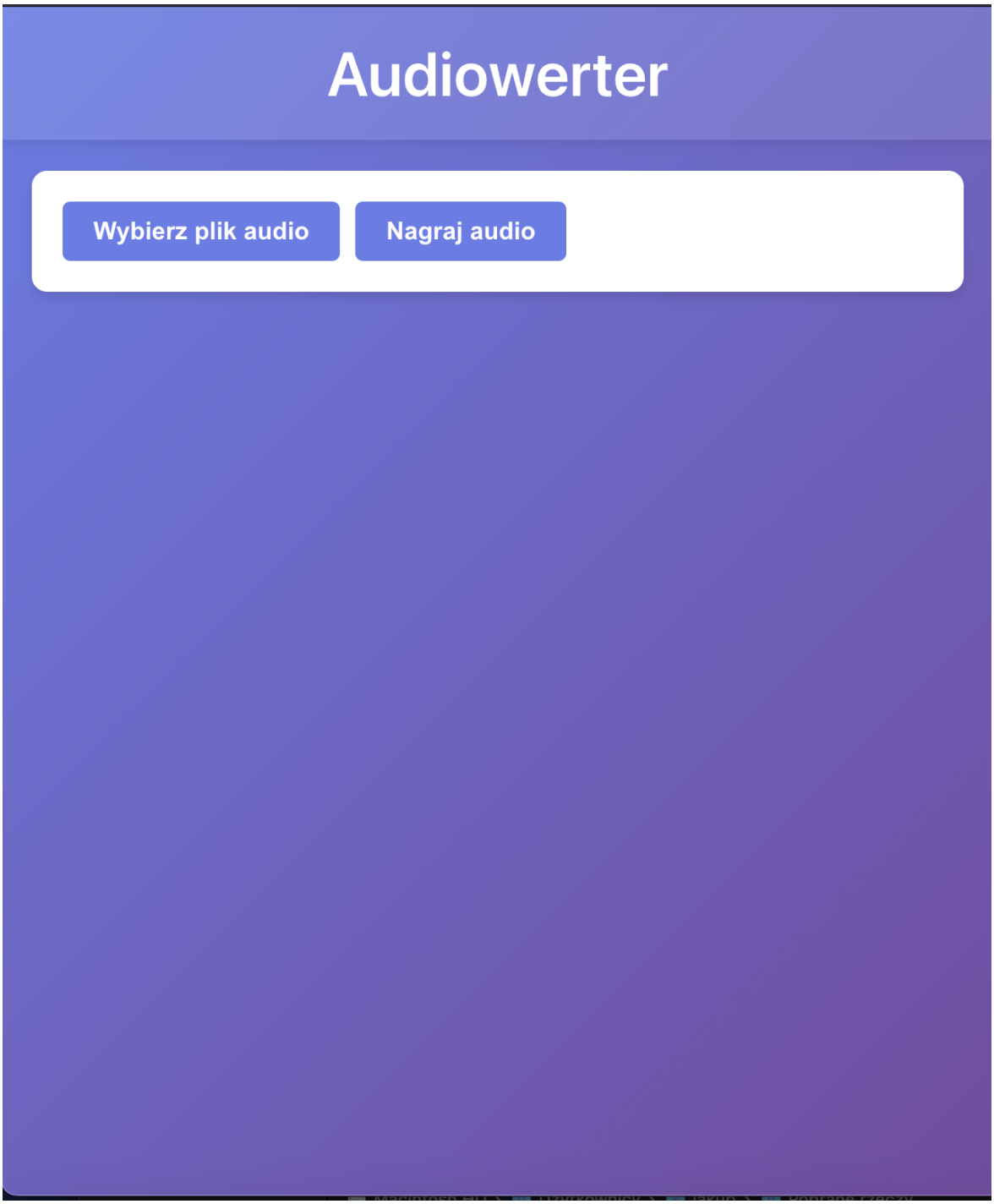
3.4 Instalacja i Uruchamianie

Poniżej przedstawiono wszystkie potrzebne kroki do instalacji potrzebnych komponentów:

1. Utworzenie folderu z potrzebnymi komponentami. Wszystkie kolejne kroki powinny być wykonywane w tym folderze.
2. Pobranie Node.js przez:
   - macOS: `brew install node` w terminalu
   - Windows: instalator z `https://nodejs.org/en/download`
   - Linux: `sudo apt install node` w terminalu
3. Wpisanie w cmd/terminal `npm install .`
4. Umieszczenie plików z folderu electron_react z repozytorium projektu.
5. Pobrać "file-saver" przez komendę: `npm install file-saver` a potem `npm install -save-dev @types/file-saver`
6. Pobrać wszystkie potrzebne komponenty .
7. Napisać w cmd/terminal `npm start .`
8. ewentualnie istnieje możliwość stworzenia pliku uruchamiającego przez wpisanie w cmd/terminal:`npm run dist`

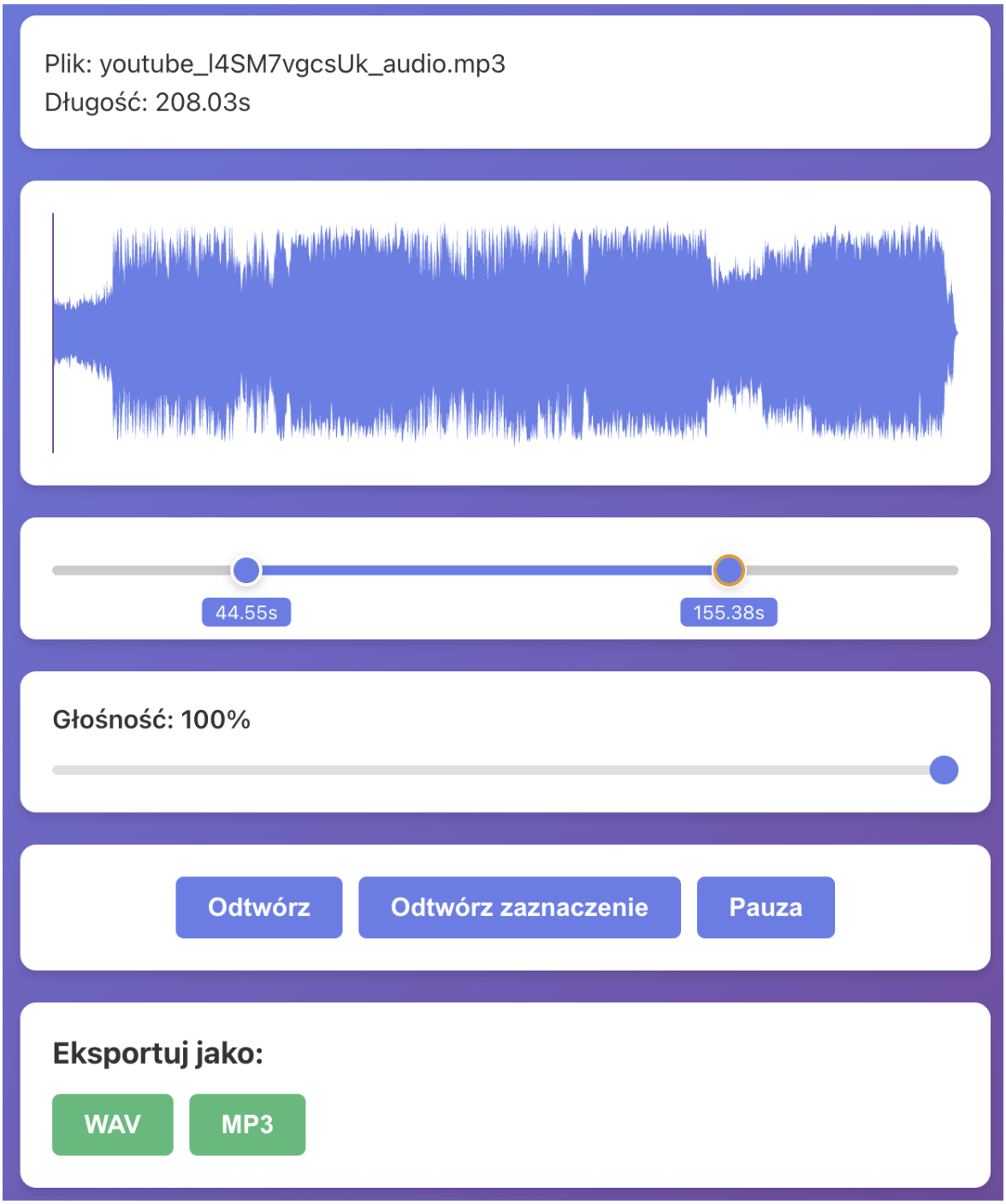
3.5 Użytkowanie

Po uruchomieniu aplikacji użytkownikowi zostaje wyświetlony poniższy interfejs graficzny umożliwiający przesyłanie oraz przetwarzanie plików audio.



Rysunek 3: Interfejs

Użytkownik ma dwie opcje: może wgrać plik audio oraz nagrać go od razu w programie. Po wczytaniu pliku zostaje przedstawiony użytkownikowi następujący interfejs:



Rysunek 4: Interfejs

Program określa (od góry):

- długość w sekundach
- waveform
- Suwaki do obcinania audio
- Suwak do określenia głośności
- Trzy przyciski do osłuchiwania wybranych fragmentów
- Eksport audio w formacie WAV lub mp3

3.6 Proces realizacji, problemy i ewentualne rozwiązania

Początkowo założono, że można użyć gotowego pliku *app.py* ponieważ jest to teoretycznie możliwe, jednak nie jest samowystarczalny jak rozwiązanie JavaScriptem. Dodatkowo wymaga zainstalowania wszystkich potrzebnych

bibliotek, ffmpega i samego Pythona tak jak w rozwiązaniu webowym. Zostaliśmy przy Pythonie bo liczyliśmy, że pisanie programu jeszcze raz w JavaScripcie zajmie niepotrzebnie sporo czasu. Udało się to w takiej konfiguracji program uruchomić, jednak nie działało wgrywanie audio, program pokazywał biały ekran. Dużo debugowania nie było, ponieważ uznaliśmy, że lepszym rozwiązaniem będzie jednak JavaScript. Wszystko co związane było z tymi próbami zostało w folderze "*electron_py*".

W folderze "*electron_js*" został napisany nowy kod z rozwiązaniem JavaScriptowym, jednak problem białego ekranu nie zniknął. Tym razem zaczęliśmy szukać problemu, szczególnie używając konsole w "DevTools", które oferuje Electron. Pomagało to w debugingu. Zostało umieszczone dużo poleceń w kodzie aby każdy proces po kolei był "logowany", aby lepiej można było zauważyć co zawiodło. Doszło nawet do tego, że program był pozbawiany funkcjonalności do momentu kiedy zacznie działać. Były przypuszczenia, że "AudioContext" czyli API używana do przetwarzania dźwięku, może być przyczyną problemów ponieważ po usunięciu program się nie zawieszał. Ostatecznie udało się wgrywać pliki audio ale nie można było ich modyfikować co mijało się z celem. Ta wersja programu nie jest w repozytorium.

Trzecią propozycją było przekierowanie uwagi na html, który raczej nie był przebudowywany tak jak backend. W naszej mobilnej wersji, która była gotowa dobrze sprawdził się React więc postanowiliśmy go tutaj zaimplementować. Z taką implementacją uzyskaliśmy końcowy efekt korzystając w dużej mierzę z rozwiązań zastosowanych już w wersji mobilnej. Ta wersja aplikacji znajduje się w folderze "electron_react".

4 Aplikacja mobilna

4.1 Założenia

Wersje mobilną aplikacji przygotowano za pomocą narzędzia Ionic ()ionicframework.com, z implementacją do aplikacji na potrzeby systemu Android. Założono przeniesienie funkcjonalności z wersji webowej, z potrzebnymi zmianami na rzecz współdziałania z systemem android, oraz specyfiką projektowania aplikacji na potrzeby mobilne:

- **Lokalne przetwarzanie plików dźwiękowych** - tak jak w wersji webowej użytkownik może przekazać plik z urządzenia do załadowania do aplikacji, gdzie przejść może przez dalszą obróbkę. Ze względu na systemem uprawnień dla aplikacji systemu Android, użytkownik musi wyrazić zgodę, aby aplikacja mogła mieć dostęp do systemu plików. Należało zapewnić obsługę prośby i przyznawania takiego uprawnienia.
- **Obsługa wielu formatów audio**
- **Interfejs dostosowany do obsługi na urządzeniu mobilnym** - wygoda obsługi dotykiem, oraz dostosowanie do obsługi ekranów o różnorakich rozmiarach i proporcjach.
- **Przekazywanie nagrania głosowego** - użytkownik może nagrać dźwięk z mikrofonu z poziomu aplikacji i przekazać go do obróbki. Tak jak w przypadku przekazywania plików, użytkownik musi wyrazić zgodę na dostęp do mikrofonu dla aplikacji, należy obsłużyć ten mechanizm.

4.2 Użyte narzędzia

- **Ionic Framework** w wersji 8 (najnowszej) - SDK pozwalające na tworzenie aplikacji hybrydowych i eksport dla urządzeń Android i iOS.
- **Capacitor** - natywne środowisko do uruchamiania aplikacji (ang *runtime environment*) stworzonych w Ionic. Zawiera zestaw bibliotek potrzebnych do stworzenia działającej aplikacji.
- biblioteka **React** - biblioteka do budowania interfejsów i tworzenia skryptów. Zintegrowany jest w Ionic Framework, (także dostępna jest integracja z biblioteką Angular lub Vue).
- **Node.js** - *runtime environment* umożliwiający działanie JavaScript po stronie serwerowej. Nasza aplikacja nie będzie komunikować się z zewnętrznym serwerem, przetwarzanie będzie dokonywane lokalnie.
- **npm** - menadżer pakietów dla Node.js.
- **Android Studio** - IDE służące do tworzenia aplikacji dla systemu Android. Umożliwia wirtualizacje smartphonów w celu testowania aplikacji lub połączenie z fizycznym urządzeniem. Użyte zostało na potrzeby testów oraz zbudowania gotowej aplikacji Android.
- **@breezystack/lamejs** - wersja kodera mp3 Lamejs, jest to fork oryginalnej paczki, z poprawionym błędem: *ReferenceError: MPEGMode is not defined*. Więcej o paczce na githubie oraz podstronie na stronie npm.

4.3 Wymagania

- **Smartphone** z **API 23**+ (Android 6 lub późniejsze)
- sprawny **mikrofon** do funkcji nagrywania dostępny w urządzeniu.

4.4 Instalacja i Uruchamianie

Instalacja i przygotowanie środowiska:

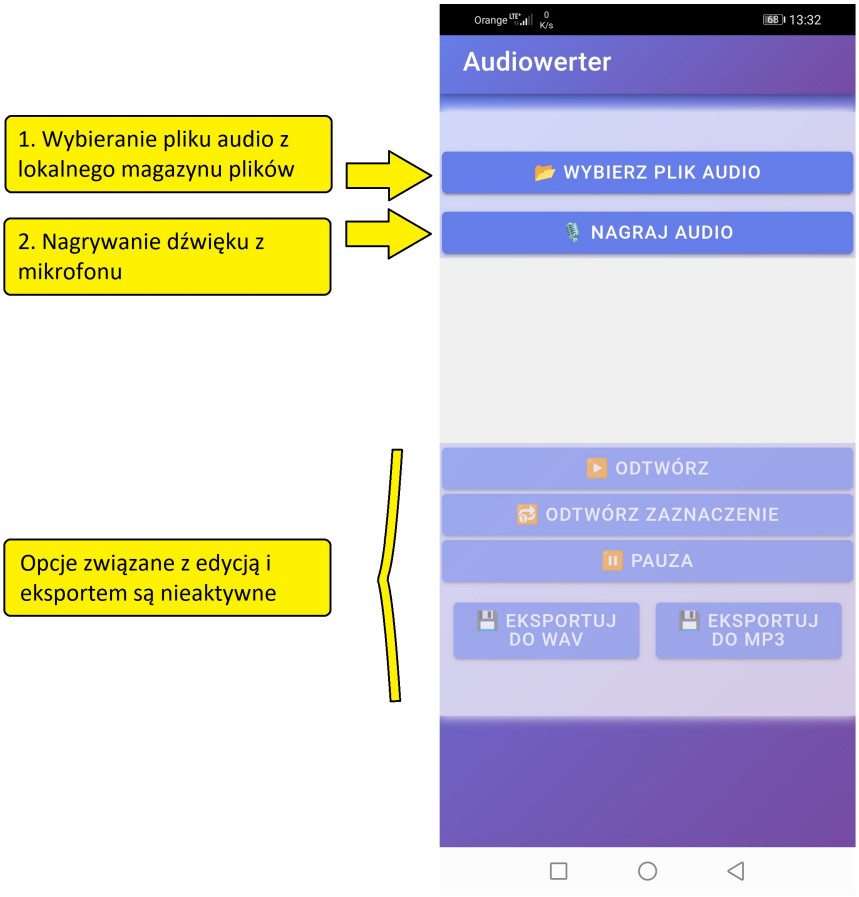
1. Instalacja najnowszej wersji Node.js nodejs.org/en/download/,
2. instalacja CLI Ionica: `npm install -g @ionic/cli`,
3. instalacja niezbędnych paczek przy pomocy npm:
   (a) `npm install @capacitor/android`,
   (b) `npm install @capawesome/capacitor-file-picker`,
   (c) `npm install @capacitor/filesystem`,
   (d) `npm install @capacitor/directory`,
   (e) `npm install @breezystack/lamejs` (zmodyfikowana wersja oryginalnego lamejs),
   (f) `npm install react-range`,
   (g) `npm install capacitor-voice-recorder --legacy-peer-deps` (w momencie tworzenia projektu nie istniała wersja paczki przygotowania do działania z najnowszą wersją Capacitor v7);
4. zbudowanie aplikacji przy pomocy polecenia: `npm run build`,
5. (a) synchronizacja z buildem androida: `npx cap sync android`,
   (b) otwarcie projektu w android studio: `npx cap open android`.

Uruchomienie projektu w Android Studio

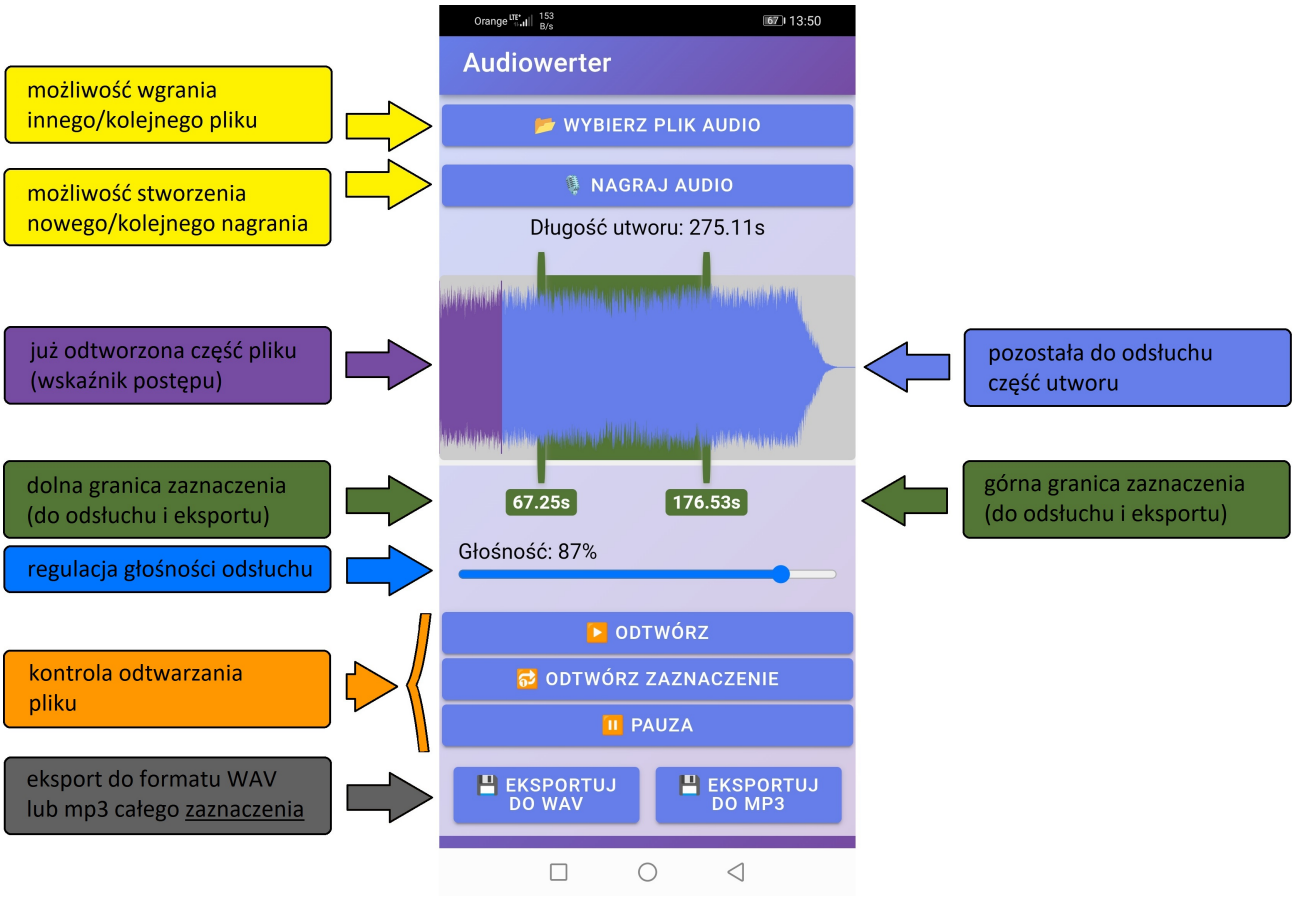
1. Uruchomienie urządzenia: przycisk „[+]" przy „Running Devices" - wybrać należy urządzenie z listy, jeżeli nie utworzono jeszcze urządzenia, należy utworzyć nowe za pomocą Device Menagera. Można także podłączyć smartphone z systemem Android za pomocą przewodu USB, uruchomić w nim opcje programistyczne → debugowanie przez USB;
2. uruchomienie aplikacji guzikiem „[Run]" (lub kombinacją klawiszy [Shift]+[F10]).

4.5 Użytkowanie

Rysunek 5: Po uruchomieniu aplikacji, przed 1. załadowaniem pliku lub 2. nagraniem audio z mikrofonu wszystkie przyciski związane z kontrolą odtwarzani i eksportem są nieaktywne. Nie widzimy też przebiegu nagrania w czasie (*wavesurf*). Po wciśnięciu przycisku [nagraj audio], jeżeli aplikacja jest uruchomiona po raz pierwszy należy nadać uprawnienia do korzystania z mikrofonu. Po rozpoczęciu nagrywania guzik zmienia się na [zatrzymaj nagrywanie], po wciśnięciu którego nagranie jest ładowane do edycji. Wciśnięcie guzika [wybierz plik audio] prosi o uprawnienia do dostępu do plików lokalnych, jeżeli nie przyznano ich wcześniej, po wybraniu pliku jest on załadowany do edycji.



Rysunek 6: Kiedy użytkownik wczyta lub nagra audio, może odsłuchać go za pomocą guzika odtwórz, wtedy odtwarzanie następuje z pozycji oznaczonej końcem część przebiegu zakolorowanej na fioletowo. Przesuwając zielone dymki z dopiskiem czasu zaznaczenia można wybrać część utworu, można go wtedy odtworzyć od początku zaznaczenia używając przycisku [odtwórz zaznaczenie]. Można także regulować głośność odsłuchu suwakiem. Na samym dole są przyciski służące do eksportu do formatu wav oraz mp3. Eksportowana jest ta część pliku, która jest zaznaczona.



4.6 Proces realizacji, problemy i ewentualne rozwiązania

Proces realizacji

Przetwarzanie plików oparto na *Capacitor Voice Recorder* do nagrywania, *Capacitor File Picker / Filesystem* – do wyboru plików z systemu i zapisu wyników. Cały interfejs oparto o komponentach Ionic (tj. *IonPage, IonHeader, IonToolbar, IonContent* itp.). Dodatkowo użyto WaveSurfer.js – do wizualizacji fali audio i zaznaczania regionów.

Początkowo planowano użyć *FFmpegKit* do konwersji na format mp3, jednak jest on przeznaczony na React Native, a nie wersje Ionic. W dalszej kolejności zdecydowano się na biblioteke Lamejs. Występuje jednak błąd w tej bibliotece, który przy próbie zapisu do pliku mp3 rzuca wyjątek: *ReferenceError: MPEGMode is not defined.* Użyto poprawionej wersji użytkownika github *breezystack*: **@breezystack/lamejs**, w której ten błąd nie występuje. Należało zadeklarować ją jako moduł, tak samo należało postąpić z *Wavesurf.*

Aby móc korzystać z mikrofonu oraz z systemu plików androida, należało w pliku *AndroidManifest* dodać następujące wpisy:

```
<uses-permission android:name="android.permission.ACCESS_MEDIA_LOCATION" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.MANAGE_EXTERNAL_STORAGE" />
```

Zauważone błędy i niedociągnięcia:

- wczytanie pliku nie resetuje pozycji odtwarzania,

- po zapauzowaniu nie można wznowić odtwarzania fragmentu, trzeba od nowa wysłuchać fragment,
- brak instrukcji z wersji webowej, potencjanie możnaby dodać osobny guzik z pomocą, albo rozbudować aplikacje o np. hamburgerowe menu, gdzie dodanoby zakładki z pomocą i sekcje *about*.