

AKADEMIA NAUK STOSOWANYCH W NOWYM SĄCZU

Wydział Nauk Inżynieryjnych
Katedra Informatyki

DOKUMENTACJA PROJEKTOWA PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

Raptor

Autor:
Mateusz Stanek
Dawid Szoldra
Filip Wachała

Prowadzący:
mgr inż. Dawid Kotlarski

Nowy Sącz 2024

Spis treści

1. Ogólne określenie wymagań projektu	3
1.1. Ogólny zarys wymagań	3
1.2. Wykorzystane czujniki	3
1.3. Zarys interfejsu	3
2. Określenie wymagań szczegółowych	7
2.1. Ogólny opis wymagań projektu	7
2.2. Ogólny zarys narzędzi użytych w projekcie	7
2.2.1. Android Studio	7
2.2.2. Kotlin	8
2.3. Wykorzystanie czujników	8
2.4. Zachowanie w niepożądanych sytuacjach	9
2.5. Dalszy rozwój	9
3. Projektowanie	10
4. Implementacja	11
5. Testowanie	12
6. Podręcznik użytkownika	13
Literatura	14
Spis rysunków	15
Spis tabel	16
Spis listingów	17

1. Ogólne określenie wymagań projektu

1.1. Ogólny zarys wymagań

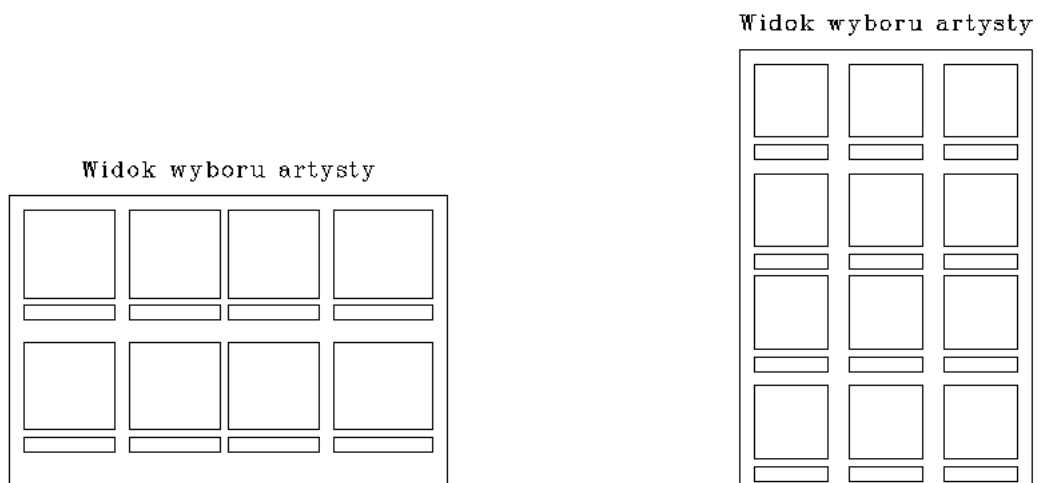
Celem programu jest pełnienie funkcji odtwarzacza muzyki oraz dodatkowo ma on pełnić rolę dyktafonu. Program będzie mógł skanować dany folder, a w nim tagi zawartych plików muzycznych i tworzyć na jego podstawie graficzną reprezentację biblioteki.

1.2. Wykorzystane czujniki

Program ma na celu wykorzystanie trzech czujników, z którymi użytkownik będzie wchodził w interakcję. Zostaną użyte następujące:

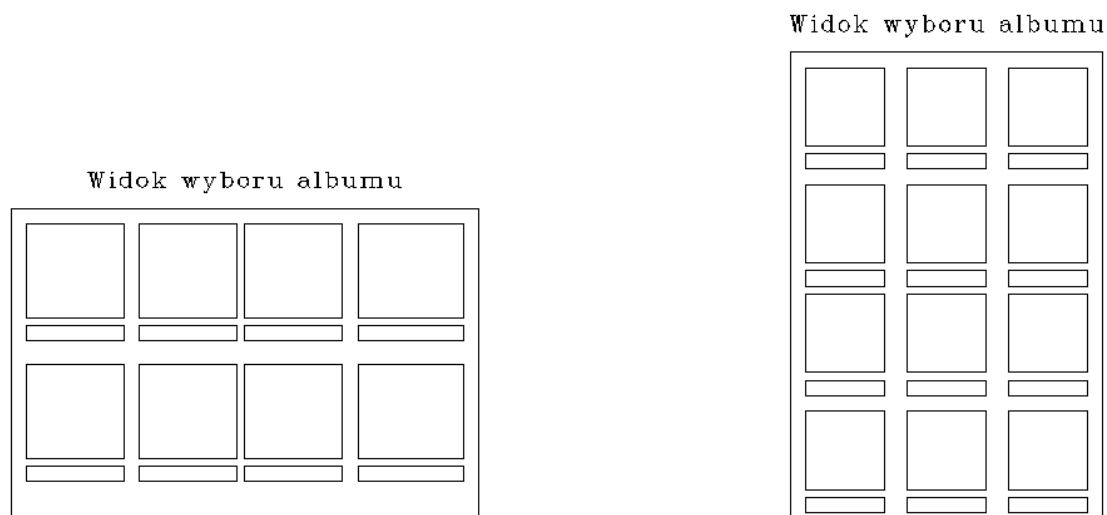
- Żyroskop - Interfejs programu będzie się zmieniał w zależności od orientacji urządzenia.
- Mikrofon - Program będzie posiadał funkcję nagrywania dźwięku. Nagrane pliki będzie można odtwarzać w odtwarzaczu
- Czujnik światła - Interfejs programu będzie mógł zmieniać swoje kolory w zależności od wykrytego poziomu światła na czujniku

1.3. Zarys interfejsu



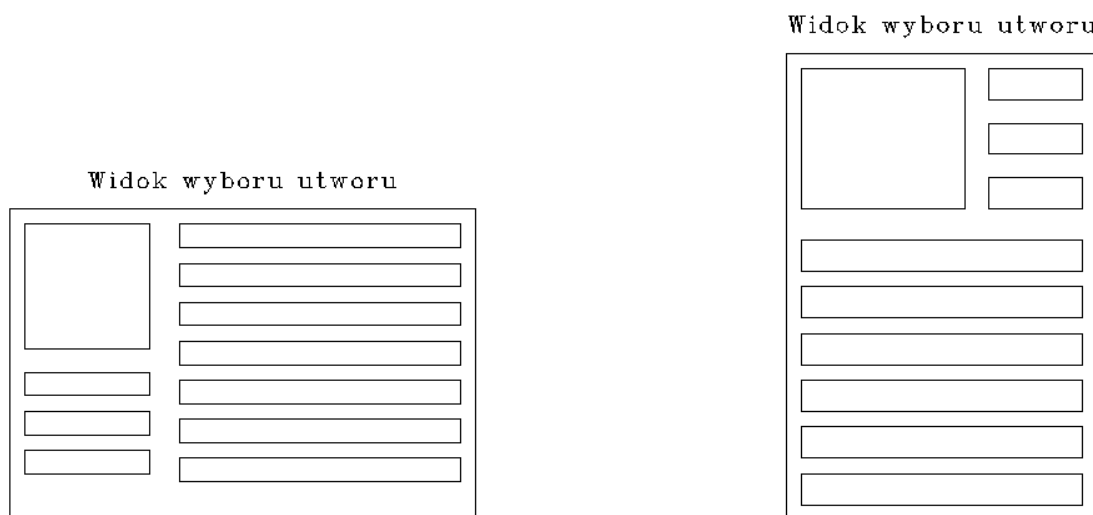
Rys. 1.1. Mockup widoku biblioteki - listing wykonawców

Widok wykonawców będzie ekranem startowym aplikacji. "Kafelki" będą zdjęciami wykonawców. Klikanie na jeden z nich przejdzie do widoku albumów danego wykonawcy



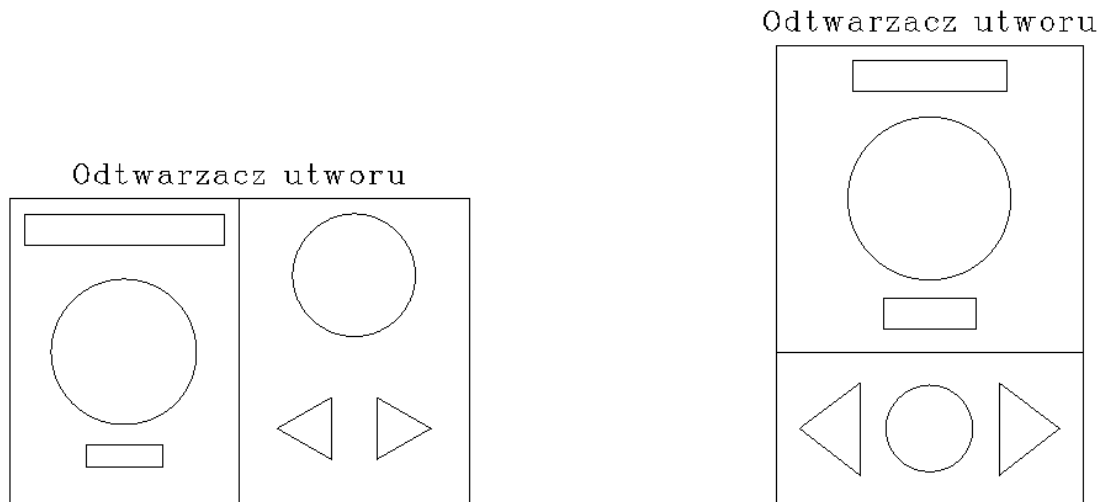
Rys. 1.2. Mockup widoku albumów danego wykonawcy

Widok albumów jest identyczny jak widok wykonawców. Jedyna różnica polega tym, że zdjęcia na kafelkach będą zdjęciami albumów.



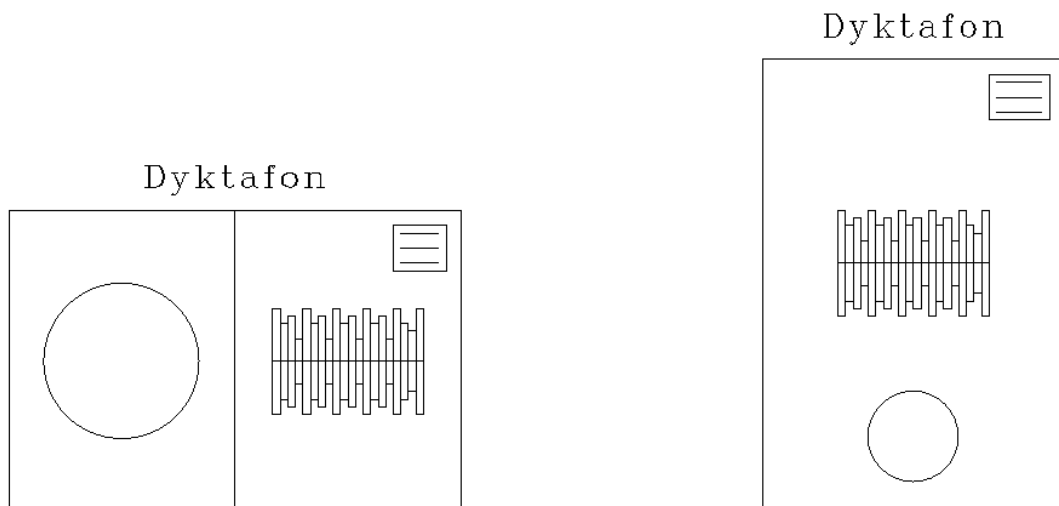
Rys. 1.3. Mockup widoku wyboru utworu

Po wejściu na jakiś album zaprezentowane zostaną zawarte w nim utwory. W lewym górnym jest zdjęcie danego albumu, a obok niego jest kilka informacji o albumie jak wykonawca, data, tytuł. Dłuższe paski to lista tytułów piosenek, które można kliknąć, aby daną piosenkę włączyć.



Rys. 1.4. Mockup odtwarzacza

Odtwarzacz będzie działał następująco: duże koło będzie stylizowane na płytę, gdzie wypełniona ona będzie obrazem albumu. Płyta ta będzie się kręcić w czasie gdy gra piosenka. Kąt płyty (od 0° , do 360°) będzie określał jak duża część piosenki została odtworzona. Kąt ten będzie określony jeszcze niezdefiniowanym efektem graficznym. Prostokąty wokół płyty to tytuł piosenki, a na dole czas grania. Kółko i wokół niego trójkąty to przyciski odtwarzania - graj/pauza, następny, poprzedni.



Rys. 1.5. Mockup dyktafonu

Do dyktafonu będzie można się dostać przesuając palcem w prawo na ekranie wykonawców. Dyktafon jest aktywowany wielkim okrągłym przyciskiem. Te kreski obok niego to wizualizacja dźwięku z mikrofonu. Menu w rogu będzie pozwalało na

m.in. skonfigurowanie folderu zapisu nagrań.

2. Określenie wymagań szczegółowych

2.1. Ogólny opis wymagań projektu

Aplikacja jest zaprojektowana w Android Studio w języku Kotlin. Całe UI aplikacji będzie zbudowane na podstawie Frameworka Jetpack Compose[1] Używając wbudowanych bibliotek w SDK Androida, będzie mogła odczytywać pliki ze wskazanego folderu. Odczytywanie tagów z plików odbędzie się za pomocą biblioteki Taglib[2], która posiada nieoficjalne bindingi do Kotlin. Wszelki processing audio np. na potrzeby wizualizacji może zostać wykonany za pomocą SDK i wbudowanego modułu AudioProcessor[3]. Odtwarzaniem pliku będzie się zajmował moduł MediaPlayer[4].

2.2. Ogólny zarys narzędzi użytych w projekcie

2.2.1. Android Studio

Android Studio jest IDE stworzonym przez Google, na bazie IntelliJ IDEA od JetBrains. Jest ono przystosowane, jak z nazwy wynika, do tworzenia aplikacji na Androida. Ku temu celu posiada wiele udogodnień, odróżniających program od typowego edytora jak np. wbudowany emulator Androida, integrujący się z całym środowiskiem, czy preview różnych elementów interfejsu - gdzie elementy te generowane są w kodzie, a nie w osobnym języku jak np. xml - bez potrzeby dekompilacji całej aplikacji.

Android Studio został użyty w projekcie, ponieważ:

- Sam program jest crossplatformowy - nasz zespół używa wielu systemów operacyjnych. Platformy takie jak MAUI, są zespoliczone z Visual Studio, czyli z Windowsem. Android Studio jest dostępny na wszystkie większe systemy operacyjne, co ułatwia nam pracę.
- Jest to program, zbudowany na podstawie IdeaJ, czyli zagłębiony jest w tym ekosystemie. Oznacza to dostęp do większej ilości pluginów niż np. Visual Studio, nie wspominając o ogólnej możliwości dostosowania ustawień.

Wady korzystania z Android Studio to m.in.

- Duże wykorzystanie zasobów - program lubi zżerać duże ilości RAMu. W tym momencie, mając otwarty mały projekt + emulator, program wykorzystuje ponad 9GB RAMu.

2.2.2. Kotlin

Kotlin został stworzony w 2010 roku przez firmę JetBrains oraz jest on przez nią rozwijany. Kotlin jest wieloplatformowym językiem typowanym statystycznie który został zaprojektowany aby współpracować z maszyną wirtualną Javy. Swoją nazwę zawdzięcza wyspie Kotlin która znajduje się w zatoce fińskiej.

Kotlin jest wykorzystywany w projekcie ze względów:

- Jest on wspierany przez Android Studio, razem z Javą i C++. Kotlin ponadto, ma dostęp do nowoczesnych frameworków jak Jetpack Compose
- Jest on *defakto* językiem do programowania na Androida - do niedawna Java mogła cieszyć się tym tytułem, ale od 2019 r. Google ogłosiło Kotlin jako rekomendowany język do tworzenia aplikacji na Android.

Składnia Kotliny wygląda następująco:

```
1 fun main() {  
2     printf("Czesc to ja, kotlin!")  
3 }
```

Listing 1. kotlin001 - Funkcje

Definicja funkcji wykonywana jest za pomocą "fun".

Zmienne w Kotlinie deklarowane są za pomocą **val** i **var**. Różnica polega na tym, że zmienne oznaczone **val** mogą zostać modyfikowane natomiast zmienne oznaczone **val** już nie.

```
1 fun main() {  
2     var nazwa = "Projekt Android"  
3     val liczba = "777"  
4 }
```

Listing 2. kotlin002 - Zmienne

Kompilator Kotliny posiada funkcję autodedukacji typów, więc w wielu wypadkach typu zmiennej nie trzeba adnotować.

2.3. Wykorzystanie czujników

- Żyroskop - Z racji, że każdy element interfejsu w Jetpack jest generowany kodem, można, przynajmniej na początku, ustawić każdą wersję interfejsu jako osobną funkcję. Następnie, w zależności od wykrytej orientacji, przy użyciu API sensorów[5], można wywoływać odpowiednią funkcję.

- Mikrofon - Funkcja dyktafonu najprawdopodobniej będzie całkiem oddzielnym Activity. Funkcjonalność ta, z natury, jest dosyć oddzielna od reszty aplikacji. Nagrania dyktafonem powinny być zapisywane do osobnego folderu. Można by zintegrować nagrania z resztą aplikacji jako osobnego wykonawcę w widoku biblioteki. Mikrofon będzie nagrywany poprzez moduł MediaRecorder[6]
- Czujnik światła - Android Studio oferuje możliwość definiowania własnych klas zajmujących się kolorystyką. Oznacza to że można używać różnych obiektów w zależności od warunków. Wykrywanie światła będzie się odbywało używając API sensorów[5]

2.4. Zachowanie w niepożądanych sytuacjach

Głównym wyjątkiem, na który może napotkać się aplikacja jest błąd odczytu albo plików, albo tagów z pliku. Kotlin, na szczęście, pozwala na łatwe sprawdzanie wartości null danych zmiennych operatorem ?. W odpowiednich fragmentach kodu dotyczących ładowania plików, będzie sprawdzana poprawność danych i najprawdopodobniej pojawi się pop-up po stronie użytkownika, że wystąpił błąd, a po stronie dewelopera błąd zostanie logowany.

2.5. Dalszy rozwój

Jeżeli praca nad aplikacją będzie się odbywała w przyszłości, należy skupić uwagę na lepszym zarządzaniu biblioteką (auto tagowanie, pobieranie miniatur z internetu, itp.). Ponadto, należy szukać błędów, które nadal zostały w aplikacji.

3. Projektowanie

4. Implementacja

5. Testowanie

6. Podręcznik użytkownika

Bibliografia

- [1] *Dokumentacja Jetpack Compose*. URL: <https://developer.android.com/compose>.
- [2] *Dokumentacja KTagLib*. URL: <https://github.com/timusus/KTagLib>.
- [3] *Dokumentacja modułu AudioProcessor*. URL: <https://developer.android.com/reference/androidx/media3/common/audio/AudioProcessor>.
- [4] *Dokumentacja modułu MediaPlayer*. URL: <https://developer.android.com/media/platform/mediaplayer>.
- [5] *Dokumentacja Sensor API*. URL: https://developer.android.com/develop/sensors-and-location/sensors/sensors%5C_overview.
- [6] *Dokumentacja modułu MediaRecorder*. URL: <https://developer.android.com/media/platform/mediarecorder>.

Spis rysunków

1.1. Mockup widoku biblioteki - listing wykonawców	3
1.2. Mockup widoku albumów danego wykonawcy	4
1.3. Mockup widoku wyboru utworu	4
1.4. Mockup odtwarzacza	5
1.5. Mockup dyktafonu	5

Spis tabel

Spis listingów

1.	kotlin001 - Funkcje	8
2.	kotlin002 - Zmienne	8