AKADEMIA NAUK STOSOWANYCH
W NOWYM SĄCZU

Wydział Nauk Inżynieryjnych
Katedra Informatyki

DOKUMENTACJA PROJEKTOWA

PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

Raptor

Autor:
Mateusz Stanek
Dawid Szołdra
Filip Wąchała

Prowadzący:
mgr inż. Dawid Kotlarski

Nowy Sącz 2024

Spis treści

1. Ogólne określenie wymagań projektu

1.1. Ogólny zarys wymagań

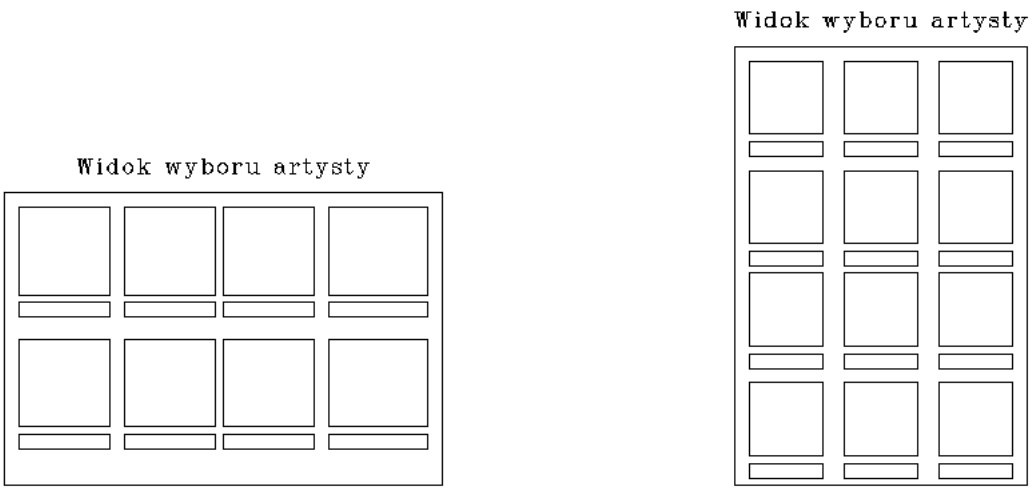
Celem programu jest pełnienie funkcji odtwarzacza muzyki oraz dodatkowo ma on pełnić rolę dyktafonu. Program będzie mógł skanować dany folder, a w nim tagi zawartych plików muzycznych i tworzyć na jego podstawie graficzną reprezentację biblioteki.

1.2. Wykorzystane czujniki

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

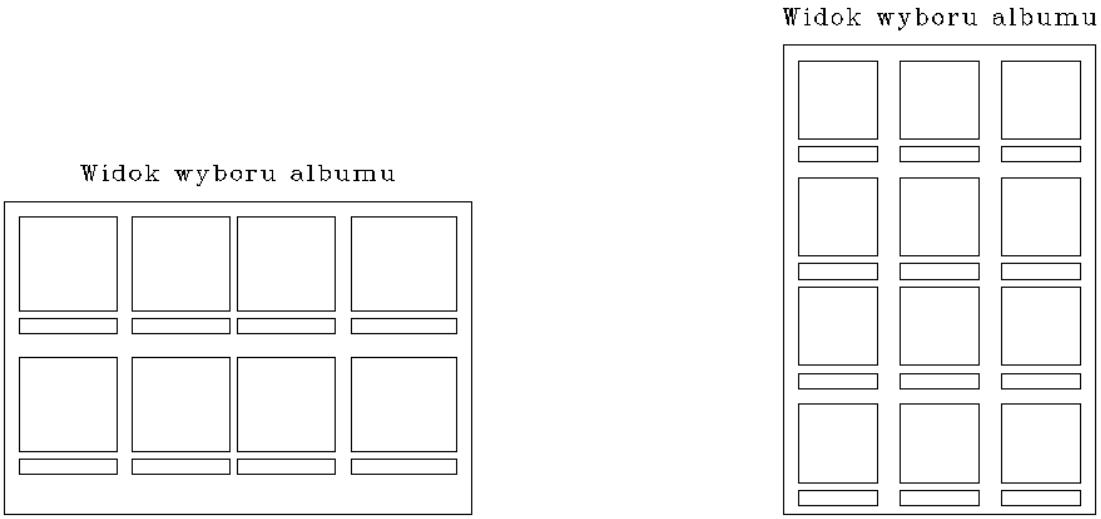
- Żyroskop - Interfejs programu będzie się zmieniał w zależności od orientacji urządzenia.
- Mikrofon - Program będzie posiadał funkcję nagrywania dźwięku. Nagrane pliki będzie można odtwarzać w odtwarzaczu
- Czujnik światła - Interfejs programu będzie mógł zmieniać swoje kolory w zależności od wykrytego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

1.3. Zarys interfejsu



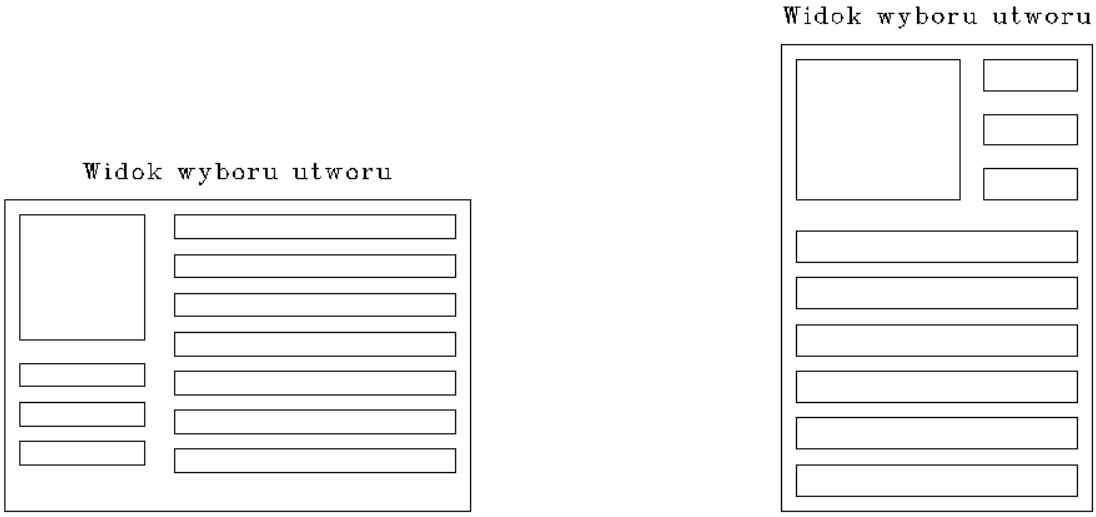
Rys. 1.1. Mockup widoku biblioteki - listing wykonawców

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



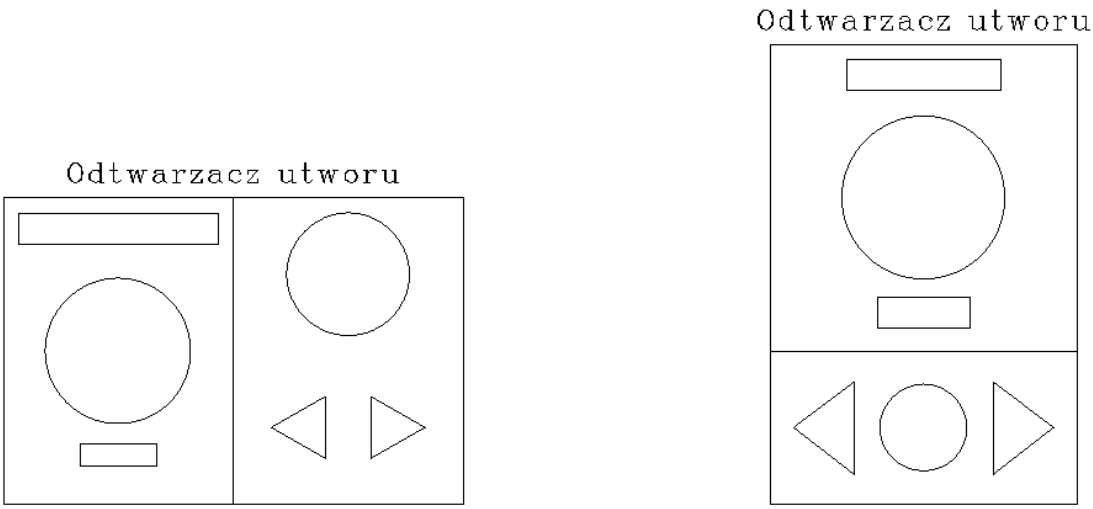
Rys. 1.2. Mockup widoku albumów danego wykonawcy

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



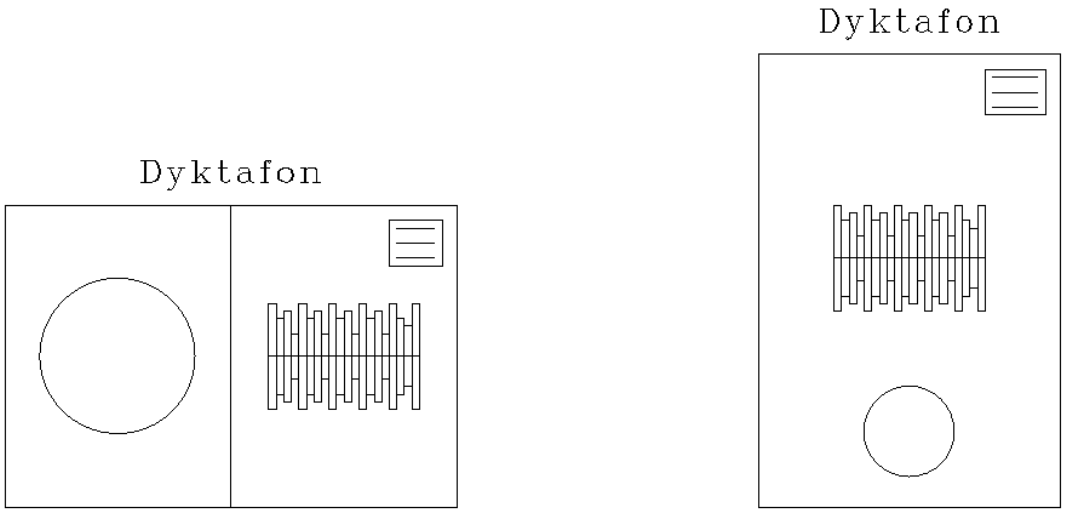
Rys. 1.3. Mockup widoku wyboru utworu

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 przesuwając 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 zarys narzędzi użytych w projekcie

2.1.1. Android Studio

Android Studio jest IDE stworzonym przez Google, na bazie InteliJ 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 zespól ozywa wielu systemów operacyjnych. Platformy takie jak `MAUI`, są zespolone z Visual Stdio, 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.

2.1.2. Kotlin

Wstęp 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 finlandzkiej.

Wykorzystanie kotlina Kotlin jest językiem który jest wykorzystywany między innymi do:

- Tworzenia i aktualizowania aplikacji mobilnych, szczególnie aplikacji na androida ale obsługuje też inne mobilne systemy operacyjne, takie ja na przykład IOS.

- Rozwoju aplikacji WEB, ich utrzymania oraz aktualizacji.
- Tworzenia, utrzymania i rozwoju aplikacji działających po stronie serwera.

Dlaczego kotlin

Kotlina warto używać między innymi dlatego że jest on kompatybilny z językiem Java. Oprócz tego, współpracuje on z wieloma platformami takimi jak:

- Windows
- Mac
- Linux
- Raspberry Pi

Kotlin jest ponadto językiem łatwym do nauki dla niedoświadczonego programisty. Jeżeli ma się wcześniejsze doświadczenie z Javą to jest jeszcze łatwiejszy.

Jest językiem darmowym w użytkowaniu, nie potrzeba żadnych opłat ani subskrypcji.

Posiada dużą, prężnie rozwijającą się społeczność oferującą wsparcie w wykonywanych projektach, bogatość zasobów do nauki oraz bibliotek do wykorzystania.

Wsparcie bibliotek Jetpack, jak na przykład Jetpack Compose który jest bardzo dobrym wyborem przy tworzeniu natywnego UI do aplikacji android.

Rozwój i historia kotlina

Kotlin został zaprojektowany w 2010 roku przez Firmę JetBrains. Według głównego programisty JetBrains, Dimitrija Jemerova, jedynym językiem który posiadał porządane funkcje był język Scala, aczkolwiek czas kompilacji był zbyt wysoki.

Pierwszy commit do repozytorium kotlina został wypuszczony 8 listopada 2010 roku. W 15 lutego 2016 została wydana pierwsza oficjalna wersja kotlina – kotlin 1.0.

W 2017 wraz z wydaniem Android Studio 3.0 oraz wypuszczeniem wersji 1.2 kotlina został on dodany przez Google jako alternatywa dla Javy

29 października 2018 roku ogłoszono wersję 1.3 kotlina a już 7 maja 2019 Google ogłosiło kotlin preferowanym językiem do pisania aplikacji mobilnych na androida.

Wraz z aktualizacją 1.4 we wrześniu 2020 kotlin otrzymał wsparcie do platform Apple.

W Maju 2021 wydano wersję 1.5 a już parę miesięcy później w listopadzie wydano wersję 1.6.

W czerwcu 2022 wydano wersję 1.7 dodając wersję alfa nowego kompilera Kotlin K2 compiler. Parę miesięcy później w grudniu wydano wersję 1.8.

Kotlin 1.9 został wydany w czerwcu 2023

Kotlin 2.0 – najnowsza wersja, została wydana w maju 2024.

Przykładowa składnia

Przykładowa składnia funkcji main

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

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 "var"mogą zostać modyfikowane natomiast zmienne oznaczone "val"już nie.

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

Listing 2. kotlin002 - Zmienne

Typ zmiennej jest wykrywahy automatycznie i nie trzebas go podawać, można ale nie trzeba.

2.2. 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[1], 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[2]

[1]https://developer.android.com/develop/sensors-and-location/sensors/sensors_overview
[2]https://developer.android.com/media/platform/mediarecorder

- 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[3]

2.3. 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.

2.4. 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]Patrz, przypis 5

3. Projektowanie

4. Implementacja

5. Testowanie

6. Podręcznik użytkownika

Spis rysunków

Spis tabel

Spis listingów