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

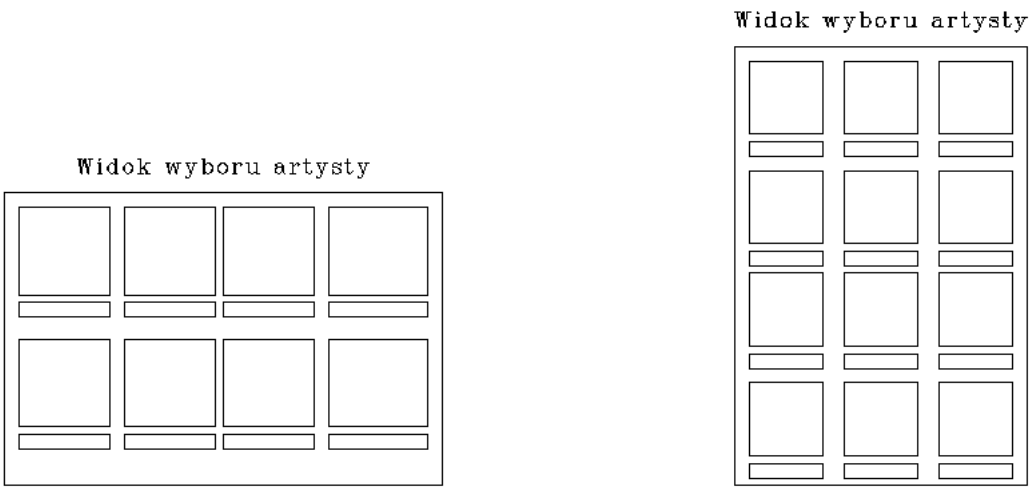
Celem programu jest pełnienie funkcji odtwarzacza muzyki oraz dodatkowo ma on pełnić rolę dyktafonu. Program będzie mógł skanować dany folder, a w nim tagi zawartych plików muzycznych i tworzyć na jego podstawie graficzną reprezentację biblioteki.

1.2. Wykorzystane czujniki

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

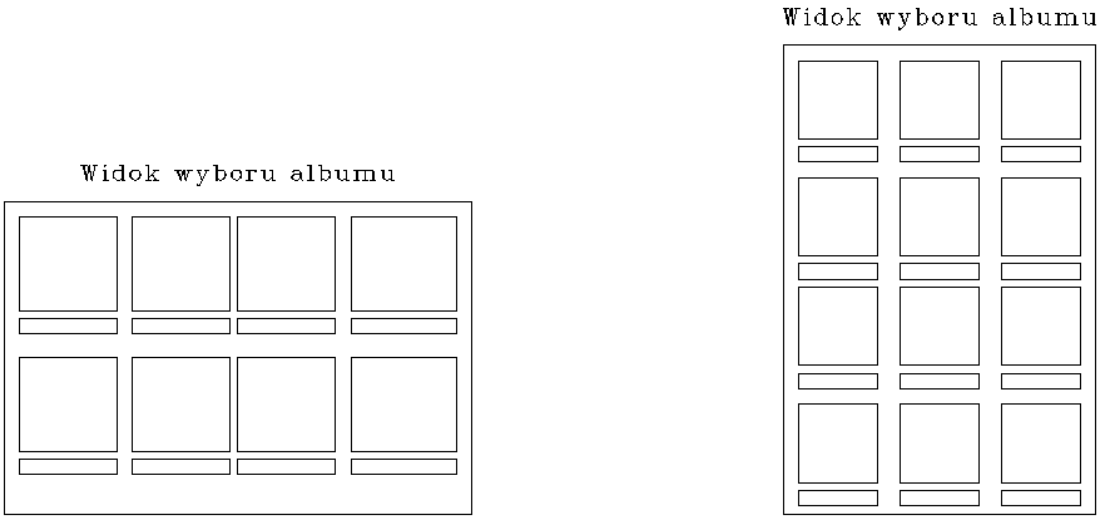
- Żyroskop - Interfejs programu będzie się zmieniał w zależności od orientacji urządzenia.
- Mikrofon - Program będzie posiadał funkcję nagrywania dźwięku. Nagrane pliki będzie można odtwarzać w odtwarzaczu
- Czujnik światła - Interfejs programu będzie mógł zmieniać swoje kolory w zależności od wykrytego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

1.3. Zarys interfejsu



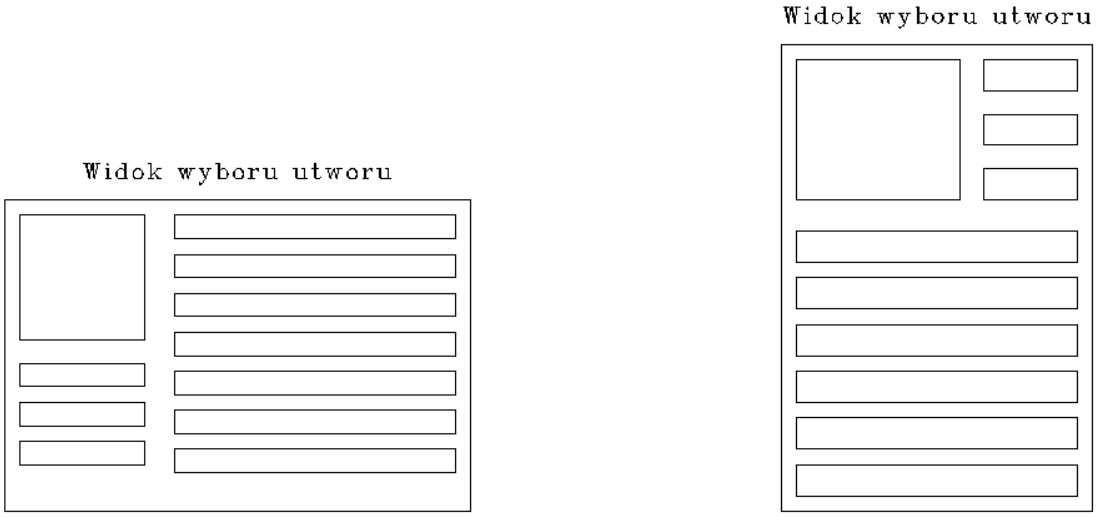
Rys. 1.1. Mockup widoku biblioteki - listing wykonawców

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



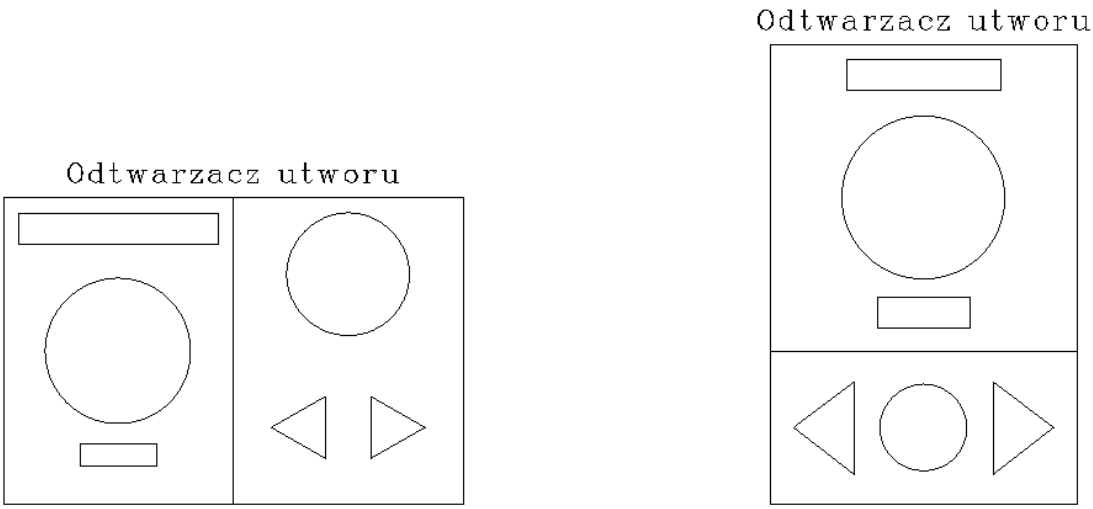
Rys. 1.2. Mockup widoku albumów danego wykonawcy

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



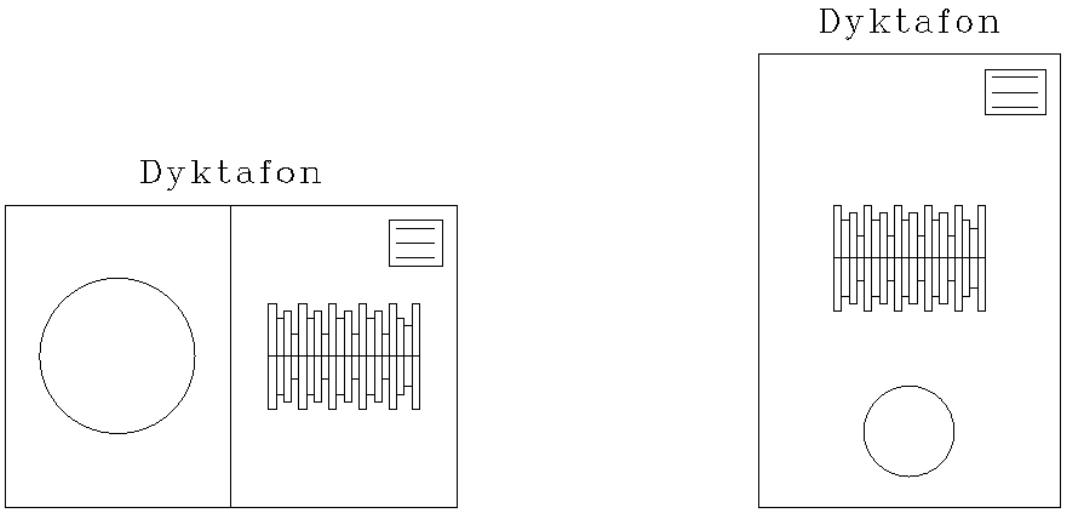
Rys. 1.3. Mockup widoku wyboru utworu

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

Android Studio to oficjalne zintegrowane środowisko programistyczne (IDE) stworzone przez Google, zaprojektowane specjalnie do tworzenia aplikacji na system Android. Bazuje na popularnym środowisku IntelliJ IDEA firmy JetBrains i oferuje dedykowane narzędzia oraz funkcje, które wspierają programistów w tworzeniu, debugowaniu i testowaniu aplikacji mobilnych. Android Studio jest wyposażone w intuicyjny edytor kodu wspierający różne języki programowania, takie jak Kotlin i Java, z funkcjami automatycznego uzupełniania kodu, podpowiedzi, refaktoryzacji i sprawdzania błędów w czasie rzeczywistym.

Środowisko to umożliwia tworzenie interfejsu graficznego za pomocą narzędzia typu drag-and-drop, które pozwala na projektowanie layoutów z możliwością podglądu na różnych urządzeniach i rozdzielczościach. Dzięki wbudowanemu emulatorowi Androida deweloperzy mogą testować swoje aplikacje bez potrzeby używania fizycznego urządzenia, a także korzystać z rozbudowanych narzędzi do debugowania, śledzenia wydajności aplikacji i analizowania zużycia zasobów. Android Studio korzysta z Gradle jako systemu budowania, co ułatwia zarządzanie zależnościami, automatyzację kompilacji i testowanie aplikacji. Środowisko wspiera także integrację z systemami kontroli wersji, takimi jak Git, co pozwala na łatwe zarządzanie projektami. Programiści mogą tworzyć aplikacje wspierające różne wersje systemu Android, od najstarszych do najnowszych, a także korzystać z Live Layout Editor do edycji interfejsu użytkownika w czasie rzeczywistym.

Kotlin to współczesny, zwięzły i statycznie typowany język programowania, opracowany przez firmę JetBrains, który od 2017 roku jest oficjalnie wspierany przez Google do tworzenia aplikacji na platformę Android. Kotlin oferuje wiele zalet w porównaniu z Javą, takich jak większa zwięzłość kodu, co prowadzi do jego lepszej czytelności i łatwiejszej konserwacji. Język ten wprowadza mechanizm bezpieczeństwa typu null, eliminując typowe problemy związane z NullPointerException, oraz wspiera programowanie funkcyjne, oferując funkcje takie jak lambdy, funkcje wyższego rzędu i rozszerzenia klas. Kotlin jest w pełni interoperacyjny z Javą, co umożliwia łatwą integrację z istniejącymi projektami napisanymi w Javie oraz używanie bibliotek Javy w nowych projektach opartych na Kotlinie.

Jedną z istotnych cech Kotlin jest wsparcie dla programowania asynchronicznego za pomocą mechanizmu coroutines, co ułatwia zarządzanie zadaniami asynchronicznymi, szczególnie ważnymi w aplikacjach mobilnych, gdzie nie można blokować inter-

fejsu użytkownika. Android Studio w połączeniu z Kotlinem służy przede wszystkim do tworzenia różnorodnych aplikacji mobilnych – od prostych gier po zaawansowane aplikacje biznesowe. Narzędzia te umożliwiają testowanie aplikacji za pomocą emulatora oraz frameworków testowych, jak również optymalizację i monitorowanie wydajności aplikacji dzięki wbudowanym narzędziom, takim jak profile pamięci czy CPU. Kotlin wspiera także rozwój aplikacji wieloplatformowych za pomocą Kotlin Multiplatform, co pozwala tworzyć aplikacje działające nie tylko na Androidzie, ale także na iOS, desktopie czy w przeglądarkach internetowych.

Android Studio to kompletne środowisko do tworzenia aplikacji Android, oferujące narzędzia wspierające cały proces deweloperski – od pisania kodu, przez testowanie, aż po wdrażanie aplikacji. Kotlin natomiast jest nowoczesnym językiem programowania, który upraszcza pisanie aplikacji poprzez zwięzłość kodu, poprawia jego bezpieczeństwo i dodaje nowoczesne funkcje. Razem te narzędzia pozwalają programistom szybko tworzyć, testować i wdrażać aplikacje mobilne wysokiej jakości.

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]

- 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]

[1]https://developer.android.com/develop/sensors-and-location/sensors/sensors_overview
[2]https://developer.android.com/media/platform/mediarecorder
[3]Patrz, przypis 5

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. Projektowanie

4. Implementacja

5. Testowanie

6. Podręcznik użytkownika

Spis rysunków

Spis tabel

Spis listingów