**Dokumentacja projektu**

**Programowanie urządzeń mobilnych**

*Lista zakupów*

**Prowadzący:** **Autorzy:**

Dr Marek Jaszuk Kacper Pacura w65559

Krzysztof Ryzak w65568

**Semestr, symbol kierunku i grupa:** 4IID/TiM SP01

Rzeszów 2023

Spis treści

[**1.** **Opis koncepcji programu – lista wymagań** 3](#_Toc139397792)

[**2.** **Opis techniczny projektu** 3](#_Toc139397793)

[**3.** **Opis implementacji – opis struktury programu z opisem roli każdego elementów** 4](#_Toc139397794)

[**4.** **Opis warstwy użytkowej z widokami ekranu** 5](#_Toc139397795)

[**5.** **Link do repozytorium z kodem źródłowym** 10](#_Toc139397796)

[**6.** **Literatura** 11](#_Toc139397797)

1. **Opis koncepcji programu – lista wymagań**

Celem projektu jest stworzenie aplikacji do zapisywania listy zakupów w języku programowania Kotlin. Aplikacja pozwala na dodawanie produktów, edycję utworzonej listy oraz usuwanie artykułów z listy. Dodatkowo projekt zawiera bazę danych przechowującą odpowiednie dane.

**Wymagania funkcjonalne**

* Dodawanie produktu (należy podać nazwę, ilość oraz jaką cenę chcemy przeznaczyć na zakup).
* Edycja dodanych produktów (należy zmienić lub dodać atrybuty).
* Usuwanie produktów z listy (należy kliknąć ikonę kosza znajdującą się obok odpowiedniego produktu).

**Wymagania niefunkcjonalne**

* Aplikacja jest przyjazna dla klienta oraz jest prosta w użyciu.
* Aplikacja działa w sposób płynny i stabilny
* Aplikacja tworzona jest w języku Kotlin
* Aplikacja posiada bazę danych SQLite i używa rekordów w niej zapisanych.

1. **Opis techniczny projektu**

* Środowisko programistyczne Kotlin: SDK BuildTools 33.0.2
* Środowisko programistyczne: Android Studio 2022.2.1
* Urządzenie z system operacyjnym Android 11

1. **Opis implementacji – opis struktury programu z opisem roli każdego elementów**

Implementacja aplikacji mobilnej pod nazwą "Lista zakupów" składa się z elementów, które współpracują ze sobą w celu umożliwienia użytkownikowi korzystania z funkcji programu takich jak dodawanie, edytowanie i usuwanie elementów z listy zakupów.

Klasa MainActivity - Odpowiada za zarządzanie interfejsem użytkownika i interakcję z użytkownikiem. Jest również implementacją interfejsu ProduktItemClickListener, który umożliwia reagowanie na interakcje z elementami listy. Główne zadania tej klasy to inicjalizacja interfejsu użytkownika, obsługa kliknięć przycisków i inicjalizacja RecyclerView.

Klasa DodajProduktLista - Jest to klasa odpowiedzialna za wyświetlanie formularza dodawania i edycji produktu na liście. Pozwala na wyświetlanie formularza w formie okna dialogowego. Klasa ta obsługuje interakcje użytkownika z formularzem, takie jak wprowadzanie danych produktu, zapisywanie zmian i zamykanie okna dialogowego.

Klasa ListaViewModel - Jest to klasa ViewModel, odpowiada za przechowywanie i zarządzanie danymi związanymi z listą produktów. Udostępnia metody, takie jak dodawanie, aktualizowanie i usuwanie produktów, które są wywoływane przez interakcje użytkownika. Komunikuje się z repozytorium ProduktItemRepository w celu wykonywania operacji na bazie danych.

Klasa ListaZakupowApp - Inicjalizuje bazę danych ProduktItemDB oraz repozytorium ProduktItemRepository, które są używane przez klasę ListaViewModel do zarządzania danymi.

Klasa ProduktItem - Jest to klasa reprezentująca pojedynczy produkt na liście zakupów. Jest to encja (Entity) używana w bazie danych Room. Zawiera pola takie jak nazwa, ilość, cena. Posiada również unikalne identyfikatory generowane automatycznie. Ta klasa jest również używana w adapterze ProduktItemAdapter do wyświetlania elementów listy w RecyclerView.

Klasa ProduktItemAdapter - Jest to adapter RecyclerView, który łączy dane produktów z ich prezentacją w interfejsie użytkownika. Tworzy widoki ViewHolderów i wiąże dane produktów z odpowiednimi elementami interfejsu użytkownika. Obsługuje również interakcje użytkownika poprzez interfejs ProduktItemClickListener.

Klasa ProduktItemViewHolder - jest odpowiedzialna za reprezentację pojedynczego elementu listy w RecyclerView w aplikacj

Interfejs ProduktItemClickListener - Jest to interfejs, który definiuje metody dla interakcji użytkownika z elementami listy. Jest implementowany przez klasę MainActivity i przekazywany do adaptera ProduktItemAdapter. Pozwala na reagowanie na zdarzenia takie jak edycja lub usunięcie produktu.

Interfejs ProduktItemDao - Jest to interfejs DAO (Data Access Object) z biblioteki Room. Definiuje metody związane z operacjami na bazie danych dla encji ProduktItem, takie jak pobieranie wszystkich produktów, dodawanie, aktualizowanie i usuwanie produktów.

Klasa ProduktItemDB - Jest to klasa abstrakcyjna bazy danych Room. Dostarcza instancję bazy danych. Zawiera definicje encji oraz dostarcza DAO (ProduktItemDao) do wykonywania operacji na bazie danych.

Klasa ProduktItemRepository - Jest to repozytorium, które działa jako pośrednik między ViewModel (klasa ListaViewModel), a DAO (klasa ProduktItemDao). Zarządza operacjami na bazie danych, takimi jak dodawanie, aktualizowanie i usuwanie produktów. Udostępnia również LiveData z listą wszystkich produktów, które są obserwowane przez ViewModel.

Każdy z tych elementów spełnia konkretną funkcję w implementacji aplikacji "Lista zakupów", co umożliwia użytkownikowi interakcję z listą, zarządzanie produktami i aktualizowanie widoku interfejsu użytkownika w czasie rzeczywistym.

1. **Opis warstwy użytkowej z widokami ekranu**

Po włączeniu telefonu (lub emulatora) na ekranie, gdzie znajdują się aplikacje należy wybrać „Lista zakupów”.

Obraz zawierający tekst, logo, Czcionka, krąg

Opis wygenerowany automatycznie

Rysunek 1 Ikona aplikacji

Po otwarciu aplikacji wyświetla się ekran startowy, w którym znajduje się przycisk „Dodaj produkt”, który dodaje nowy produktu do listy.

Obraz zawierający gadżet, Komunikator, Telefon komórkowy, Przenośne urządzenie do komunikacji

Opis wygenerowany automatycznie

Rysunek 2 Główne okno aplikacji

Po wciśnięciu przycisku „wysuwa się” okno, w którym dodajemy nazwę, ilość oraz jaką kwotę chcemy przeznaczyć na dodawany produkt.

Obraz zawierający tekst, gadżet, Komunikator, Urządzenie przenośne

Opis wygenerowany automatycznieObraz zawierający tekst, Komunikator, gadżet, Urządzenie przenośne

Opis wygenerowany automatycznie

Rysunek 3 Wypełnianie danych

Rysunek 4 Okno dodawanie produktu do listy

Po kliknięciu „Zapisz do listy” produkt zostaje dodany do listy, znajdującej się w głównym oknie aplikacji.

Obraz zawierający gadżet, Komunikator, Urządzenie elektroniczne, multimedia

Opis wygenerowany automatycznie

Rysunek 5 Główne okno aplikacji, wraz z zapisanym do listy produktem

Następie po kliknięciu w produkt, wyświetla się okno, w którym można sprawdzić szczegóły takie jak ilość oraz ile chcemy wydać na dodany produkt lub dokonać edycji przedmiotu, który po kliknięciu przycisku na dole, zostanie zaktualizowany.

Obraz zawierający tekst, gadżet, Komunikator, Urządzenie przenośne

Opis wygenerowany automatycznie

Przedmioty można dodawać, a następnie usuwać, za pomocą znajdującego się na końcu linii kosza.

Obraz zawierający zrzut ekranu, gadżet, Komunikator, multimedia

Opis wygenerowany automatycznieObraz zawierający zrzut ekranu, tekst, gadżet, Komunikator

Opis wygenerowany automatycznie

Rysunek 6 Okno aplikacji po usunięciu jednego przedmiotu

Rysunek 7 Okno aplikacji po dodaniu produktów

1. **Link do repozytorium z kodem źródłowym**

Projekt został zrealizowany z wykorzystaniem systemu kontroli wersji Git. Do kodu projektu stworzono dokumentację poprzez system komentarzy dokumentujących. Dokumentacja oraz projekt został umieszczony w repozytorium dostępnym pod adresem:

1. https://github.com/KacperPacuraW65559/Programowanie-Urzadzen-Mobilnych-Projekt

2. https://github.com/KrzysztofRyzak/Programowanie-Urz-dze-Mobilnych-w65568

1. **Literatura**
2. <https://stackoverflow.com> (data dostępu: 23.06.2023)
3. <https://youtube.com> (data dostępu: 23.06.2023)
4. <https://developer.android.com> (data dostępu: 26.06.2023)
5. <https://kotlinlang.org> (data dostępu: 25.06.2023)