



Politechnika Wrocławska

**Wydział Informatyki i Zarządzania**

kierunek studiów: Informatyka

## Praca dyplomowa - inżynierska

### Aplikacja mobilna: sklep z elektroniką

Tomasz Jopek

słowa kluczowe:

Android

Firebase

MVP

krótkie streszczenie:

Celem pracy jest opracowanie aplikacji mobilnej dedykowanej dla systemu Android w wersji 5.0(Lolipop) i wyższych. Pozwalającej na przeglądanie produktów, dokonywanie zakupów w sklepie elektronicznym. W pracy zostały przedstawione obecne rozwiązania dostępne na rynku, wymagania funkcjonalne oraz нефункционалне aplikacji oraz istotne punkty dotyczące implementacji całego systemu.

opiekun pracy dyplomowej	.....	.....	.....
	<i>Tytuł/stopień naukowy/imię i nazwisko</i>	<i>ocena</i>	<i>podpis</i>
Ostateczna ocena za pracę dyplomową			
Przewodniczący Komisji egzaminu dyplomowego	.....	.....	.....
	<i>Tytuł/stopień naukowy/imię i nazwisko</i>	<i>ocena</i>	<i>podpis</i>

Do celów archiwalnych pracę dyplomową zakwalifikowano do:\*

a) kategorii A (akta wieczyste)

b) kategorii BE 50 (po 50 latach podlegające ekspertyzie)

\*niepotrzebne skreślić

pieczęć wydziałowa

Wrocław 2017

## Spis treści

1. Wstęp .....	4
1.1. Wprowadzenie .....	4
1.2. System operacyjny Android .....	5
1.3. Przegląd aktualnych rozwiązań .....	9
1.4. Geneza .....	11
1.5. Cel pracy .....	11
2. Projekt .....	12
2.1. Wymagania funkcjonalne i нефункционалне .....	12
2.2. Diagram przypadków użycia .....	13
2.3. Wzorzec architektoniczny MVP – Model View Presenter .....	16
2.4. Diagram pakietów .....	17
3. Interfejs użytkownika .....	19
3.1. Widok startowy .....	19
3.2. Widok kategorii oraz produktów .....	20
3.3. Widok listy ulubionych produktów oraz listy złożonych zamówień .....	21
3.4. Widok podsumowujący zamówienie .....	22
3.5. Widok szczegółów zamówienia oraz wyszukiwania .....	23
3.6. Widok logowania oraz rejestracji .....	24
3.7. Widok koszyka .....	25
4. Implementacja .....	26
4.1. Wprowadzenie .....	26
4.2. Środowisko i narzędzia programistyczne .....	26
4.3. Widoki, komponenty .....	26
4.4. Implementacja wzorca MVP .....	27
4.5. Baza danych .....	28
4.6. Dodatkowe biblioteki .....	29
4.7. Wzorce projektowe .....	29
4.8. Problemy implementacyjne .....	31
5. Podsumowanie .....	33
5.1. Podsumowanie pracy .....	33
5.2. Plany rozwoju aplikacji .....	33
Spis ilustracji .....	34
Bibliografia .....	35

## **Streszczenie**

W dzisiejszych czasach trudno wyobrazić sobie osobę, która nie wykorzystywałaby smartfona w codziennym życiu. Urządzenia te pozwalają na szybszy dostęp do informacji, usprawnienie codziennych czynności takich jak dokonywanie zakupów, opłacanie rachunków, kontrolowanie swojego grafiku dnia. Dzięki natywnym aplikacjom oraz serwisom webowym stworzonym także z myślą o smartfonach jest to możliwe. Z roku na rok obserwujemy trend mówiący o tym, iż zakupy realizowane przy pomocy smartfonów rosną w zatrważającym tempie, a obecnie stanowią już około 60% wszystkich transakcji.

W pracy zostanie przedstawiona wizja, projekt i implementacja aplikacji natywnej dedykowanej dla systemu Android w wersji 5.0 i wyższych. Aplikacja ta będzie służyła do dokonywania zakupów w sklepie z asortymentem elektronicznym. Pozwoli na przeglądanie aktualnej oferty sklepu dzięki zastosowaniu bazy NoSQL czasu rzeczywistego Firebase, dokonania zakupu wybranych produktów, obserwowania najbardziej interesujących nas towarów.

Zostaną także omówione technologie wykorzystane przy implementacji aplikacji oraz wzorce architektoniczne i projektowe dzięki, którym projekt będzie mógł być łatwo skalowalny oraz rozbudowywany w przyszłości.

## **Abstract**

Nowadays it is hard to imagine a person which doesn't use smatphone in daily life. This devices allow faster access to information, minimize time of daily activities like for instance doing shoping, paying bills or cotrolling your daily schedule. Thanks to native applications and web services which are created to work fine also on mobile devices, these all things are possible. Year by year we notice trend which shows that doing shoping by using smartphones grows really fast. Today it is something about 60% of all transactions.

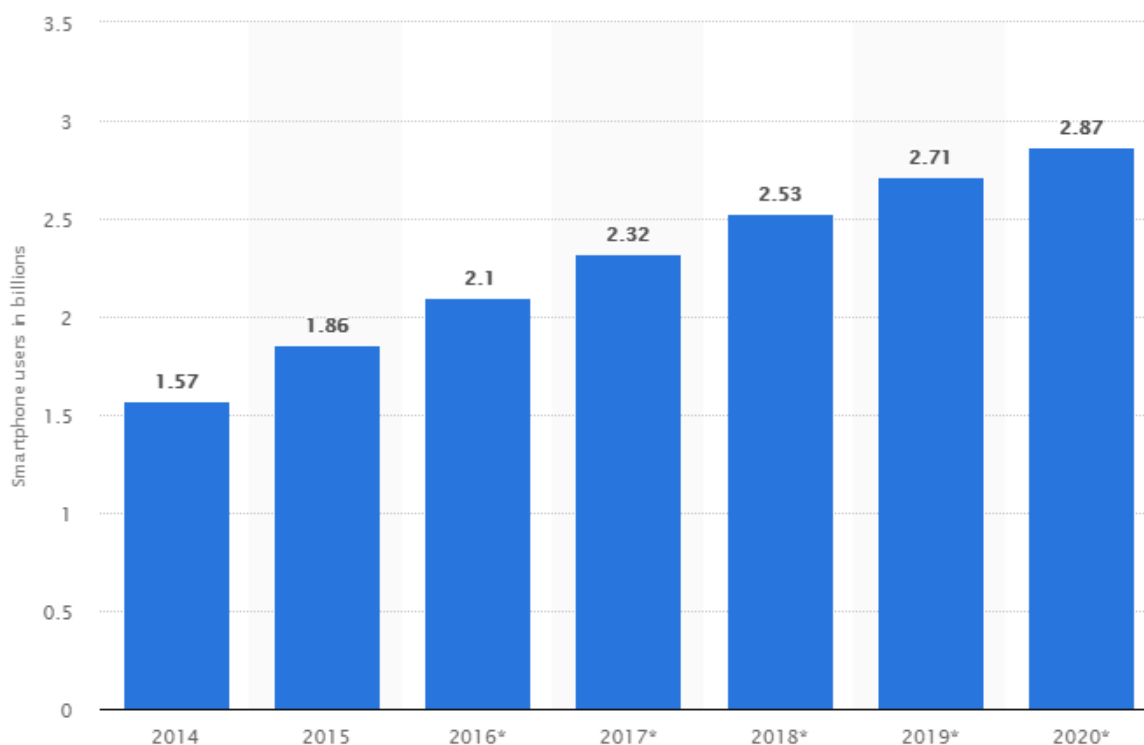
In this paper there will be presented vision, project and implementation of native application dedicated to Android system with 5.0 version and above. Application will be used for making orders in shop with electronics products. It will allow to watch whole shop's offer thanks to use of NoSQL database – Firebase, make an order of selected products, watching the most interesting us products.

This paper will also cover technologies which has been used in development, architectonic and design patterns thanks to which the project will might be easily scalable and developed in the future.

# 1. Wstęp

## 1.1. Wprowadzenie

Początek technologii informacyjnej, informatyki i komputerów rozpoczął się kilkadziesiąt lat temu od powstania pierwszych kalkulatorów, a następnie pojawienia się ENIAC'a, czyli Electronic Numerical Integrator And Computer. Maszyny, która została zbudowana wyłącznie z elementów elektronicznych. W 1946 roku John von Neymann zaproponował architekturę, którą wykorzystuje się w budowie komputerów do dnia dzisiejszego. Kolejnym krokiem milowym w informatyce było powstanie komputerów osobistych. Pierwszym takim był Altair wyprodukowany w 1975 roku. Przez kolejne lata powstają procesory o coraz większej liczbie tranzystorów, a co za tym idzie większej mocy obliczeniowej[12]. Komputery osobiste zaczynają być wykorzystywane coraz częściej do rozrywki, a nie jedynie pracy. Następnie wraz z pojawieniem się Internetu, komputery oferują kolejne możliwości i rozpoczynają swoją drogę ku dominacji w obrębie dostarczania oraz wymiany informacji pomiędzy użytkownikami. W danym momencie czasu do wymiany informacji bardzo chętnie wykorzystywane są telefony komórkowe oferujące możliwość połączeń głosowych oraz wymianę SMSów. W pewnym momencie czasu dwa trendy zaczęły się ze sobą integrować. Niektórzy uznają za ten moment rok 2007 czyli premierę pierwszego iPhone'a, chociaż pierwsze smartfony powstały pod koniec lat dziewięćdziesiątych. Tym samym powstał nowy trend, który zaczął się bardzo dynamicznie rozwijać za sprawą silnej konkurencji, głównie pomiędzy takimi firmami jak Apple oraz Samsung. Smartfony z upływem czasu zaczęły wypierać z rynku takie urządzenia jak odtwarzacze MP3, MP4, aparaty fotograficzne. Stały się głównym narzędziem, z którego korzystamy codziennie by sprawdzić najnowsze informacje, skontaktować się ze znajomymi przy pomocy portali społecznościowych, czy dokonać zakupów z dowolnego miejsca, w którym mamy dostęp do Internetu.



Rys. 1.1 Liczba użytkowników smartfonów na świecie w latach 2014-17(w miliardach)[11]

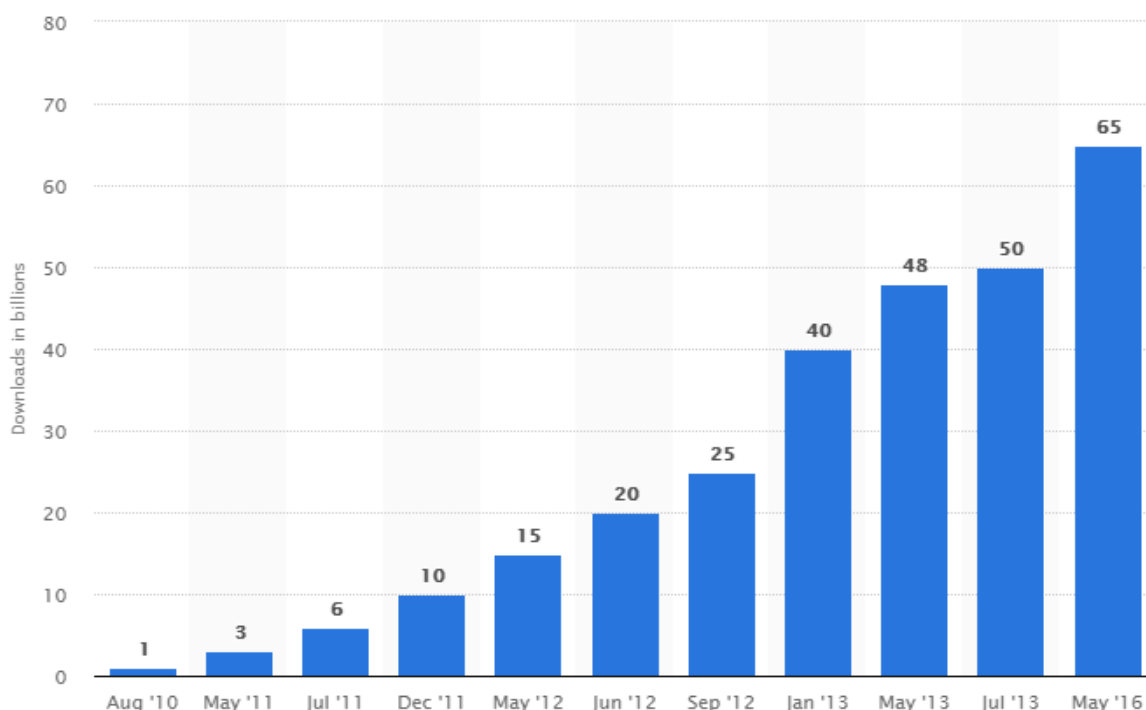
Na powyższym wykresie(Rys.1.1) widać ciągle trend rosnący w sprzedaży smartfonów. Jeśli chodzi o polski rynek firma IDC twierdzi, że w pierwszym kwartale 2017 roku sprzedano w Polsce ponad 2 miliony smartfonów, co oznacza 10% wzrost w porównaniu z rokiem ubiegłym. Należy także dodać, że 89% stanowiły urządzenia z systemem Android. Prognozy na rok 2017 mówią o tym, iż sprzedaż smartfonów wyniesie około 8,5 miliona sztuk, a urządzenia z technologią LTE będą stanowiły 90% smartfonów sprzedanych w bieżącym roku.

Zatem na podstawie przedstawionych statystyk należy przyjąć, że kilkukrotnie zwiększy się również rynek aplikacji mobilnych dedykowanych na urządzenia mobilne.

## 1.2. System operacyjny Android

System operacyjny Android pojawił się niewiele ponad 10 lat temu, jako projekt open-source, licencji Apache 2.0, a obecnie jest najpopularniejszym systemem operacyjnym stosowanym na urządzeniach mobilnych. W czasie swojego rozwoju pokonał takich rywali jak Symbian, BlackBerry czy Windows Phone. Obecnie jedynym konkurentem dla Android jest system operacyjny firmy Apple.

Gwałtowny rozwój systemu Android zaczął się, gdy w 2005 roku Google kupiło firmę Android Inc., założoną między innymi przez Andy Rubina. Rubin i inni założyciele dalej rozwijali swój system dla nowego właściciela. W listopadzie 2007 roku Google ogłosiło utworzenie Open Handset Alliance, czyli zrzeszenia firm, których celem był rozwój standardów dla urządzeń mobilnych, a także po raz pierwszy przedstawiło Androida. We wrześniu 2008 roku została wydana wersja 1.0. Jej głównymi funkcjonalnościami była wbudowana przeglądarka internetowa, obsługa kamery, podstawowy ekran główny, synchronizacja z aplikacjami Google'a(Gmail, Kontakty, Kalendarz) oraz podstawowa wersja map, odtwarzacz multimedialny i powiadomienia pokazywane na pasku powiadomień. Pierwszym urządzeniem działającym pod kontrolą systemu Android 1.0 był HTC Dream. Kolejnym ważnym punktem w historii Androida był lipiec 2013, kiedy sklep Google Play posiadał ponad milion aplikacji dla systemu Android.

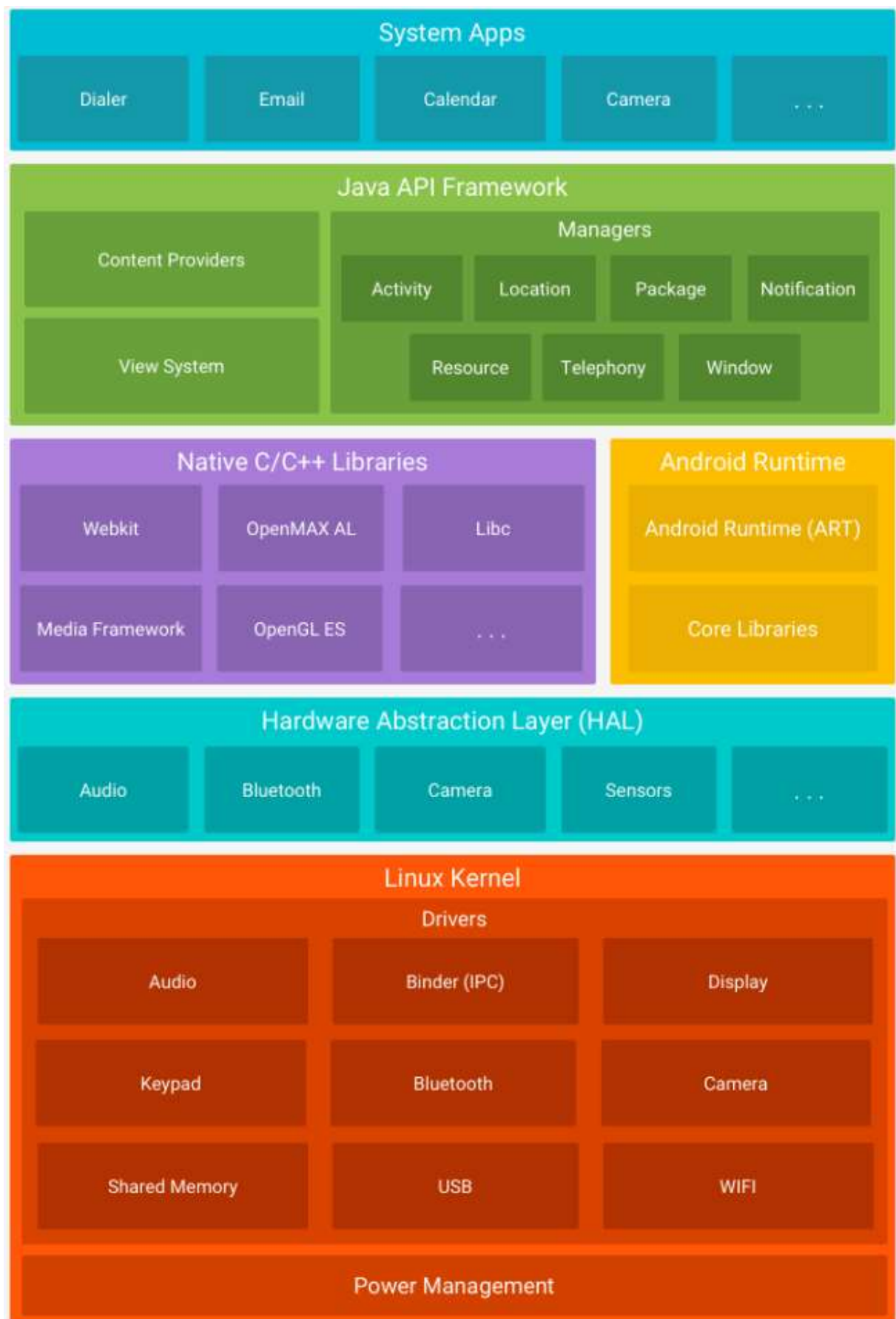


Rys. 1.2 Liczba pobrań aplikacji ze sklepu Google Play[13]

*Tabela 1.1 Tabela przedstawiająca udział w rynku poszczególnych wersji systemu[5]*

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.6%
4.1.x	Jelly Bean	16	2.3%
4.2.x		17	3.3%
4.3		18	1.0%
4.4	KitKat	19	14.5%
5.0	Lollipop	21	6.7%
5.1		22	21.0%
6.0	Marshmallow	23	32.0%
7.0	Nougat	24	15.8%
7.1		25	2.0%
8.0	Oreo	26	0.2%

Na podstawie Rys.1.2 można postawić hipotezę, iż w najbliższych latach zapotrzebowanie na aplikacje, strony internetowe dostosowane do urządzeń mobilnych będzie duże oraz będzie wciąż rosło. Z Tab 1.1 wynika, iż ponad 75% urządzeń mobilnych działających pod kontrolą systemu Android posiada wersję powyżej 5.0.



Rys. 1.3 Architektura systemu Android[5]

System Android oparty jest jądro Linuxa(Rys.1.3). Zajmujące się między innymi wielowątkowością oraz zarządzaniem pamięcią. Warstwa abstrakcji sprzętowej dostarcza interfejsów pozwalających na komunikację z poszczególnymi komponentami sprzętowymi takimi jak kamera czy Bluetooth. Urządzenia działające na wersji systemu 5.0 wyższych, uruchamiają każdą aplikację jako oddzielny process, który posiada własną instancję Android Runtime.

Biblioteki natywne napisane są głównie przy pomocy języków C oraz C++. W warstwie tej zawarto biblioteki OpenGL, WebKit służące na przykład do renderowania grafiki 3D czy obsługi przeglądarki internetowej.

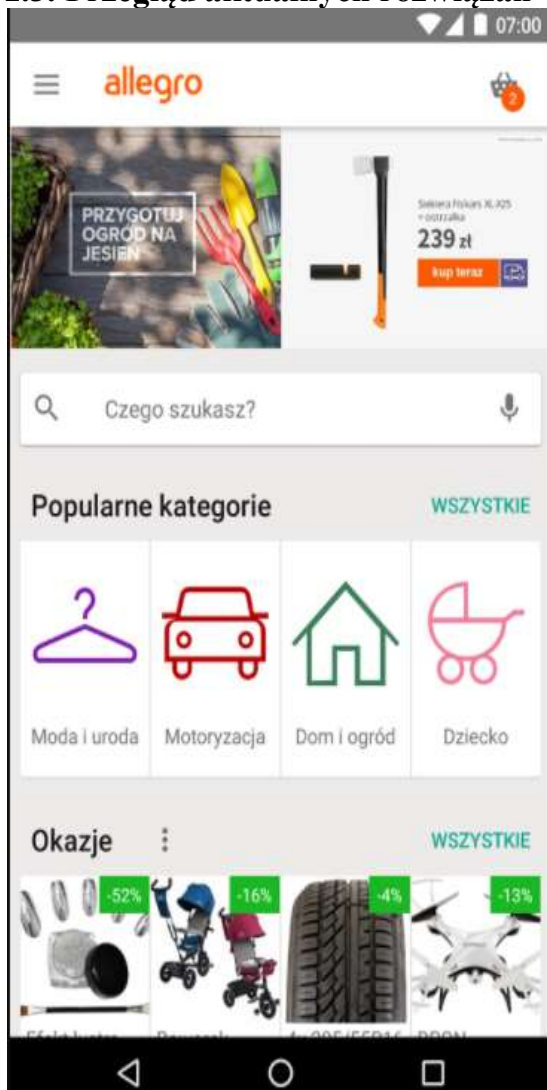
Środowisko wykonawcze(Android Runtime) jest umieszczone na tym samym poziomie co biblioteki natywne. W jego skład wchodzi między innymi biblioteki Java oraz maszyna wirtualna Dalvik.

Java Api Framework to warstwa zawierająca klasy służące tworzeniu aplikacji dla systemu Android. Umożliwia zarządzanie zasobami urządzenia, połączeniami czy lokalizacją urządzenia.

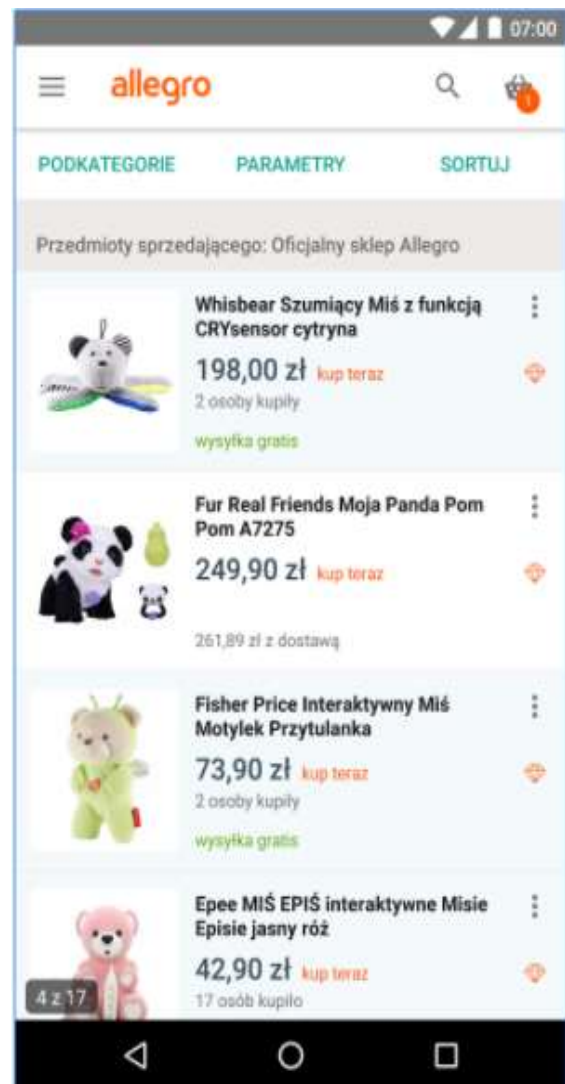
System Apps to najwyższa warstwa zawierająca wszystkie aplikacje standardowe oraz pobrane ze zewnętrznych źródeł.



### 1.3. Przegląd aktualnych rozwiązań



Rys. 1.4 Widok startowy aplikacji allegro



Rys. 1.5 Widok listy produktów

Aplikacja serwisu allegro(Rys.1.4, Rys.1.5) jest jedną z najbardziej znanych w Polsce aplikacji pozwalających na dokonywanie zakupów za pośrednictwem urządzeń mobilnych. Umożliwia dokonywania zakupów z wielu kategorii, dodawanie własnych ogłoszeń, komentowanie zakończonych transakcji, obserwowanie wybranych produktów, ostatnio przeglądane produkty.

Do głównych zalet wymienionej wyżej aplikacji należą:

- Intuicyjny i przejrzysty interfejs użytkownika
- Łatwy dostęp do ostatnio przeglądanych produktów
- Zaawansowane filtrowanie i sortowanie produktów



Rys. 1.6 Widok główny aplikacji



Rys. 1.7 Widok listy produktów

Następnym przykładem dobrze zaprojektowanej i funkcjonalnej aplikacji pośredniczącej w zakupach przy pomocy urządzenia mobilnego jest produkt sklepu X-kom(Rys.1.6, Rys.1.7).

Jako plusy aplikacji należy wskazać między innymi:

- Stabilność oraz szybkie działanie
- Zapamiętane, domyślne adresy wysyłki zamówień
- Powiadomienia o zmianach cen obserwowanych produktów

Z kolei minusami mogą być:

- Limit obserwowanych produktów
- Brak sortowania według liczby opinii
- Brak możliwości kopiowania nazw produktów

#### **1.4. Geneza**

Genezą powstania aplikacji, która jest przedmiotem danej pracy inżynierskiej były rozmowy autora z ludźmi, którzy zajmują się na co dzień elektroniką, automatyką i robotyką. Zgodnie twierdzili, iż często kupują podobne elementy elektroniczne do swoich prac. Przykładem mogą być rezystory, układy scalone, przewody, płytki stykowe, diody oraz inne przedmioty używane do tworzenia układów elektronicznych. Stwierdzili także, iż sklepy, w których dokonują kupna wyżej wymienionych materiałów nie posiadają responsywnych stron internetowych, które umożliwiałyby wygodne i przyjemne korzystanie z ich usług przy użyciu urządzeń mobilnych, w szczególności ze smartfonów. Wyrazili szczerą chęć korzystania z aplikacji, która umożliwiłaby im szybkie oraz komfortowe dokonywanie zakupów przy użyciu urządzeń mobilnych.

#### **1.5. Cel pracy**

Celem pracy jest opracowanie oraz implementacja natywnej aplikacji mobilnej, przeznaczonej dla systemu Android w wersji 5.0 i wyższych, pozwalającej na dokonywanie zakupów w sklepie z elektroniką, przeglądanie aktualnej oferty sklepu, wsparcie dla listy ulubionych produktów. Historię oraz aktualnie realizowane zamówienia będą dostępne dla użytkownika by mógł kontrolować swoje zakupy.

Zakres pracy obejmuje zbudowanie aplikacji przy użyciu platformy Android Studio, języka Java oraz bibliotek open-source wspomagających oraz przyspieszających tworzenie aplikacji w danym środowisku oraz bazy danych Firebase.

Praca składa się z pięciu rozdziałów. Wstęp zawiera zarys historii dotyczącej komputerów oraz samej platformy Android, a także przegląd aktualnie dostępnych aplikacji, oferujących podobne funkcjonalności. Rozdział drugi definiuje wymagania funkcjonalne oraz нефункционалне projektowanego systemu, a także przedstawia opracowywany system przy pomocy diagramów języka UML. Rodział trzeci przedstawia interfejs użytkownika aplikacji. Rozdział czwarty przedstawia technologie oraz główne punkty dotyczące implementacji. Ostatni rozdział podsumowuje całość wykonanych prac oraz prezentuje możliwe dalsze kierunki rozwoju opracowywanej aplikacji.

## **2. Projekt**

### **2.1. Wymagania funkcjonalne i нефункционалне**

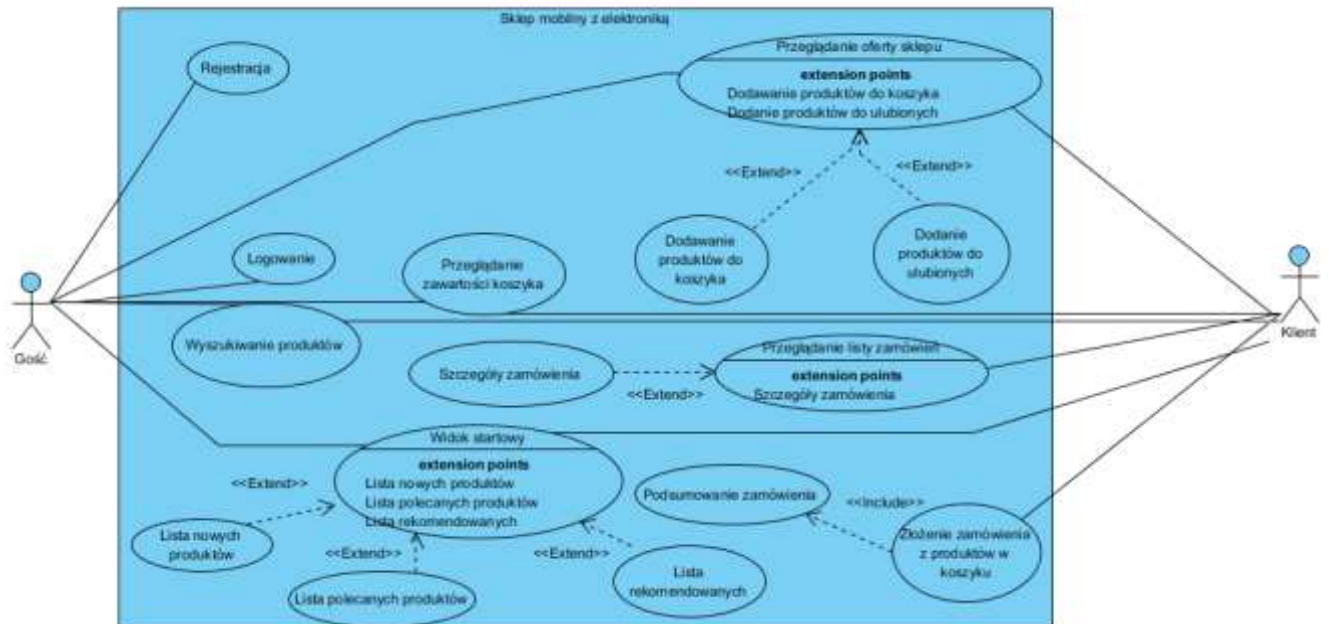
#### Wymagania funkcjonalne:

- Przeglądanie listy produktów z podziałem na kategorie dostępnych w sklepie
- Przeglądanie listy produktów ulubionych
- Przeglądanie listy dokonanych, złożonych zamówień
- Przeglądanie listy dodanych do koszyka
- Możliwość dodania produktów do koszyka
- Możliwość dodania produktu do listy ulubionych
- Możliwość usunięcia produktów z koszyka
- Możliwość usunięcia produktu z listy ulubionych
- Możliwość złożenia zamówienia, które obejmuje produkty będące w koszyku
- Możliwość wyszukiwania pośród wszystkich produktów
- Możliwość utworzenia nowego konta użytkownika
- Możliwość zalogowania się w aplikacji
- Możliwość przeniesienia zawartości koszyka z konta gościa do koszyka zalogowanego użytkownika
- Wyświetlanie podsumowania składanego zamówienia
- Wyświetlanie listy nowo dodanych produktów do oferty w widoku startowym
- Wyświetlanie listy polecanych produktów w widoku startowym
- Wyświetlanie listy produktów na promocji w widoku startowym

#### Wymagania нефункционалне:

- Działanie na systemach Android wersji 5.0 i wyższych
- Dostęp do Internetu
- Synchronizacja zmian w bazie danych ograniczona do 300ms

## 2.2. Diagram przypadków użycia



Rys. 2.1 Diagram przypadków użycia

### Przypadek użycia: Rejestracja

**Aktor:** Gość

**Opis:** Zarejestrowanie się w serwisie.

**Warunki wstępne:** Użytkownik niezalogowany. Nie posiadający konta w systemie. Przeglądanie ekranu logowania.

**Przebieg:**

1. Gość klika w napis "Create new account".
2. Otwiera się nowy widok z formularzem rejestracyjnym
3. Użytkownik wypełnia wymagane pola i zatwierdza swój wybór
4. System weryfikuje poprawność wprowadzonych danych
  - 4.a. Wprowadzone dane są poprawne.
    - 4.a.1 Zalogowany użytkownik zostaje przeniesiony do głównego widoku.
  - 4.b. Wprowadzone dane są niepoprawne.
    - 4.b.1 System informuje użytkownika o wprowadzeniu błędnych danych.

### Przypadek użycia: Logowanie

**Aktor:** Gość

**Opis:** Logowanie się w serwisie.

**Warunki wstępne:** Użytkownik niezalogowany w serwisie. Widoczny, boczny pasek menu.

**Przebieg:**

1. Gość klika w przysisk "Login".
2. Otwiera się nowy widok z formularzem logowania.
3. Użytkownik podej swój email oraz hasło.
4. System weryfikuje poprawność wprowadzonych danych.

- 4.a. Wprowadzone dane są poprawne.
  - 4.a.1. Użytkownik zostaje przeniesiony do ekranu startowego.
- 4.b. Wprowadzone dane są błędne.
  - 4.b.1 System wyświetla informację o błędnych danych i nieudanym logowaniu.

**Przypadek użycia:** Przeglądanie oferty sklepu

**Aktor:** Gość, Klient

**Opis:** Przeglądanie oferty sklepu, podzielonej na kategorie oraz poszczególne produkty.

**Warunki wstępne:** Otwarty widok głównych kategorii.

**Przebieg:**

1. Użytkownik wybiera interesującą go kategorię.
- 2.a Kategoria posiada podkategorie.
  - 2.a.1 Powrót do punktu 1.
- 2.b Otwierany jest widok produktów przypisanych do wybranej kategorii.
  - 2.b.1 Użytkownik wybiera interesujący go produkt.
  - 2.b.2 Otwierany jest widok zawierający szczegóły dotyczące wybranego produktu.

**Przypadek użycia:** Dodanie produktu do koszyka

**Aktor:** Gość, Klient

**Opis:** Dodanie do koszyka wybranej liczby produktów.

**Warunki wstępne:** Otwarty widok ze szczegółami produktu.

**Przebieg:**

1. Użytkownik wybiera interesującą go liczbę sztuk do dodania oraz zatwierdza dodanie przyciskiem.
2. System wyświetla komunikat potwierdzający dokonanie operacji.

**Przypadek użycia:** Dodanie produktu do ulubionych

**Aktor:** Klient

**Opis:** Dodanie produktu do listy ulubionych

**Warunki wstępne:** Użytkownik musi być zalogowany. Otwarty widok ze szczegółami produktu.

**Przebieg:**

1. Użytkownik klika przycisk dodający produkt do ulubionych.
2. System wyświetla komunikat informujący o wyniku operacji.

**Przypadek użycia:** Złożenie zamówienia

**Aktor:** Klient

**Opis:** Złożenie zamówienia składającego się z produktów znajdujących się w koszyku.

**Warunki wstępne:** Użytkownik jest zalogowany.

**Przebieg:**

1. Użytkownik klika przycisk z ikoną koszyka.
2. System otwiera widok przedstawiający listę produktów w koszyku.
3. Użytkownik potwierdza kliknięciem poprawność zawartości.
4. Otwiera się widok podsumowujący całe zamówienie.
5. Użytkownik wybiera metodę płatności.
6. Pojawia się przycisk umożliwiający złożenie zamówienia.
7. Użytkownik klika przycisk zatwierdzający zamówienie.
8. Pojawia się alert proszący o ponowne potwierdzenie.
- 9.a. Użytkownik potwierdza swój wybór.
  - 9.a.1 System wyświetla użytkownikowi komunikat o dokonaniu zamówienia.

9.b Użytkownik odrzuca zamówienie.

9.b.1 Użytkownik pozostaje w widoku podsumowującym zamówienie.

**Przypadek użycia:** Przeglądanie listy zamówień

**Aktor:** Klient

**Opis:** Przeglądanie listy złożonych zamówień.

**Warunki wstępne:** Otwarty pasek boczny menu.

**Przebieg:**

1. Użytkownik wybiera opcję – “kupione”
2. System otwiera widok z listą złożonych zamówień.

**Przypadek użycia:** Szczegóły zamówienia.

**Aktor:** Klient

**Opis:** Ukazanie wszystkich produktów wchodzących w skład zamówienia.

**Warunki wstępne:** Otwarty widok listy zamówień.

**Przebieg:**

1. Użytkownik wybiera interesujące go zamówienie.
2. System wyświetla w formie okna dialogowego listę wszystkich przedmiotów wchodzących w skład zamówienia.

**Przypadek użycia:** Wyszukiwanie produktów

**Aktor:** Gość, Klient

**Opis:** Wyszukiwanie produktów przy użyciu paska nawigacyjnego.

**Warunki wstępne:** Otwarty widok z paskiem zawierającym ikonę lupy.

**Przebieg:**

1. Użytkownik klika ikonę lupy na pasku nawigacyjnym
2. System udostępnia pole tekstowe w pasku nawigacyjnym do wprowadzenia frazy do wyszukania.
3. Użytkownik wpisuje interesującą go frazę.
4. System wyświetla po wpisaniu minimum dwóch znaków podpowiedzi dopasowujące się do aktualnie wpisywanego tekstu.
5. Po zatwierdzeniu wyszukiwanej frazy przechodzimy do widoku produktów spełniających kryteria wyszukiwania.

**Przypadek użycia:** Przeglądanie zawartości koszyka.

**Aktor:** Gość, Klient

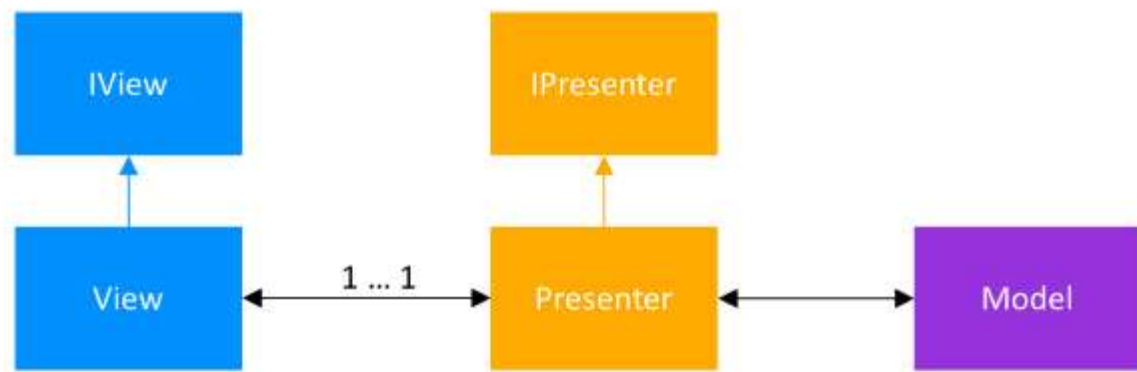
**Opis:** Przeglądanie aktualnej zawartości koszyka.

**Warunki wstępne:** Otwarty widok zawierający w prawym dolnym rogu przycisk z ikoną koszyka.

**Przebieg:**

1. Użytkownik klika przycisk z ikoną koszyka.
2. System wyświetla w postaci listy, produkty znajdujące się w koszyku.

### 2.3. Wzorzec architektoniczny MVP – Model View Presenter



Rys. 2.2 Wzorzec architektoniczny MVP - Model View Presenter[14]

Warstwa modelu jest odpowiedzialna za implementację logiki biznesowej, realizację komunikacji sieciowej, na przykład z zewnętrznymi serwisami, API oraz obsługę i komunikację z warstwą bazy danych. Prezentator zarządza stanem widoku. Przetwarza dane zebrane przez warstwę modelu danych i przekazuje je do warstwy widoku. Z drugiej strony obsługuje zdarzenia występujące w widoku, czyli na przykład wciśnięcie przycisku. Warstwa widoku odpowiedzialna jest za wyświetlanie, przedstawianie treści użytkownikowi oraz przekazywanie akcji użytkownika do warstwy prezentera[9][14]. Omawianą architekturę przedstawia Rys.2.2.

Zalety użycia wzorca MVP:

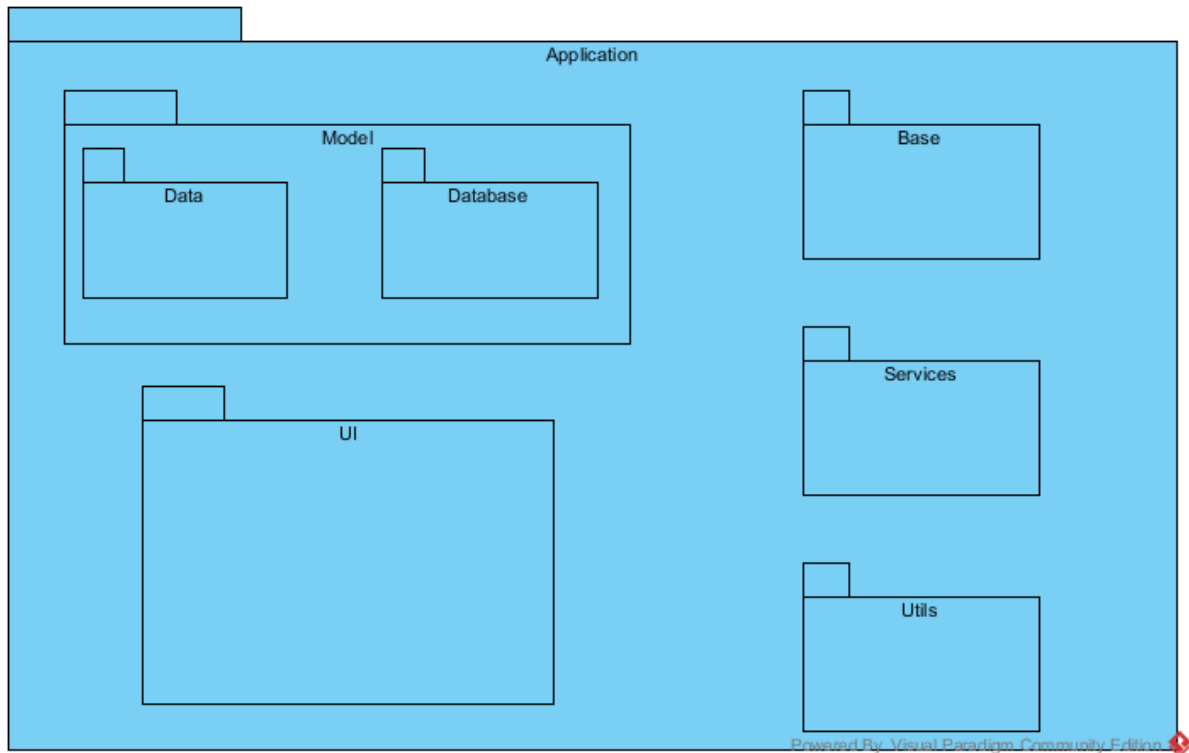
- Łatwiej testowalny kod – mniej zależności z frameworkiem Androida
- Aplikacja prostrza w utrzymaniu – kod mniej podatny za zmiany
- Skalowalność – prosta rozbudowa o nowe funkcjonalności
- Kod niezależny od obiektów interfejsu i bazy danych
- Bardziej przejrzysty kod – klasy aktywności i fragmentów stają się krótsze, nie zawierają wszystkich funkcjonalności w sobie

Wady:

- Większa liczba klas



## 2.4. Diagram pakietów



Rys. 2.3 Diagram pakietów

Przy modelowaniu diagramów przypadków użycia oraz wyżej przedstawionego diagramu pakietów, pomocna okazała się następująca książka[4].

### Base

Pakiet ten zawiera klasy oraz interfejsy bazowe, które służą jako punkt wyjścia dla kolejnych komponentów systemu. Znajdują się w nim na przykład takie klasy abstrakcyjne jak `BaseFragment` oraz `BasePresenter` uogólniające funkcjonalności wykorzystywane przez konkretne klasy implementujące fragmenty oraz prezenterów.

### Services

Pakiet ten zawiera serwisy odpowiedzialne za funkcjonalności działające w tle aplikacji. Przykładem takiego obiektu oraz jego zastosowania jest serwis, który nasłuchuje zmian na przykład w cenie produktów, które użytkownik posiada na swojej liście ulubionych. Po wystąpieniu zmiany na wyżej wymienionych produktach użytkownik otrzymuje dotyczące tego powiadomienie.

### Utils

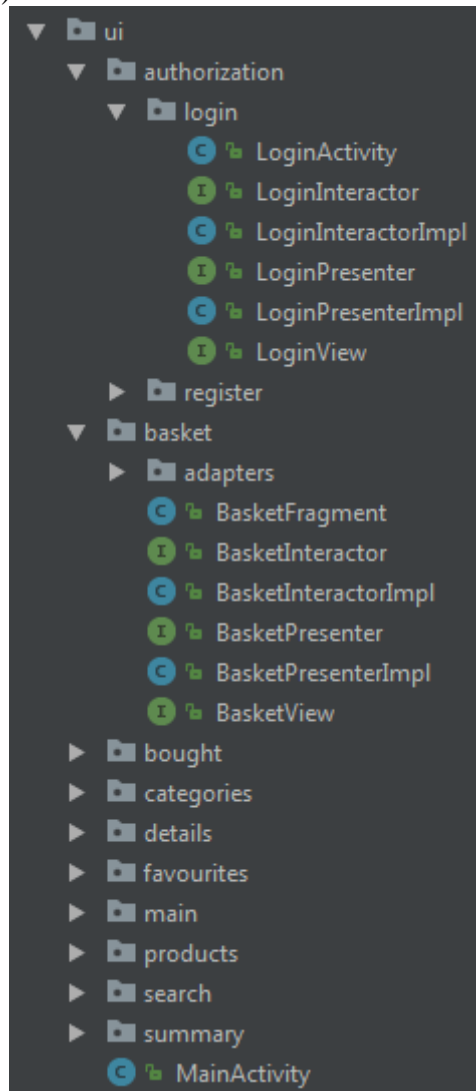
Pakiet `utils` zawiera klasy implementujące narzędzia wspomagające wykonywanie pewnych operacji. Znajdują się tam między innymi implementacje komparatorów, walidatorów, animacji.

## Model

Zawiera w sobie dwa pakiety: data oraz database. Odpowiadają one kolejno za organizację klas typu POJO(Plain Old Java Objects) oraz za komunikację z warstwą bazy danych.

## UI

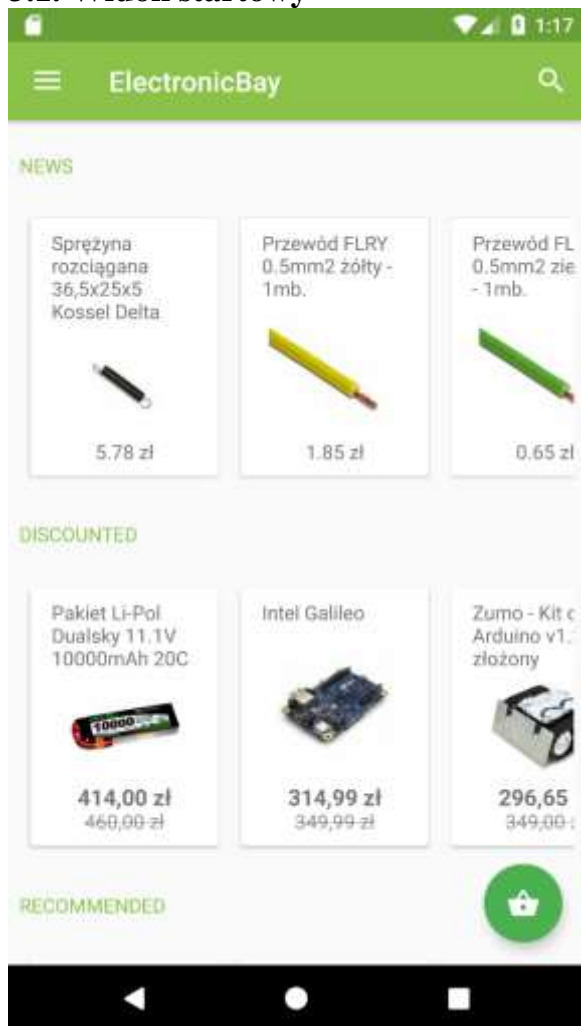
To pakiet zawierający w sobie pakiety organizujące klasy odpowiedzialne za implementację poszczególnych widoków systemu, wykorzystując architekturę MVP omówioną we wcześniejszym rozdziale. Struktura pakietu ui została przedstawiona na poniższym rysunku(Rys.2.4).



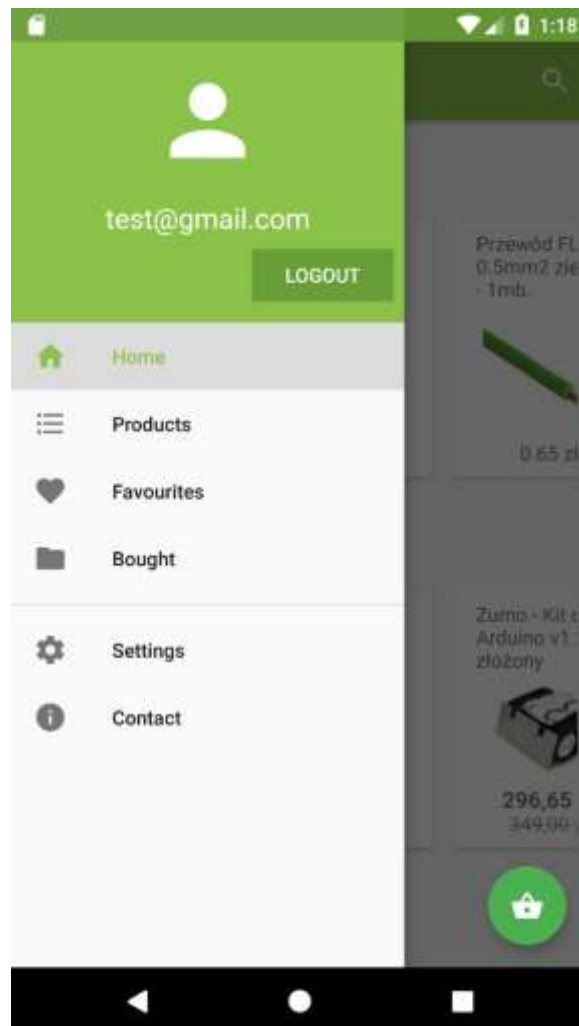
Rys. 2.4 Struktura pakietu ui

### 3. Interfejs użytkownika

#### 3.1. Widok startowy



Rys. 3.1 Widok ekranu startowego aplikacji



Rys. 3.2 Widok bocznego menu

W widoku ekranu startowego aplikacji (Rys.3.1) użytkownik widzi listę nowych produktów dostępnych w ofercie sklepu, listę produktów, które obowiązują obniżka cen oraz listę produktów dla niego rekomendowanych. Z poziomu przedstawionego widoku mamy możliwość nawigowania do widoku szczegółów wybranych produktów, widoku koszyka. Ponadto mamy dostęp do funkcjonalności wyszukiwania produktów w całym serwisie oraz dostęp do bocznego paska służącego do nawigacji pomiędzy poszczególnymi sekcjami aplikacji. Menu nawigacyjne (Rys.3.2) pozwala nam do przejścia do takich widoków jak lista produktów, ulubionych, dokonanych zamówień. W nagłówku mamy również przycisk pozwalający na przejście do aktywności logowania lub rejestracji.

### 3.2. Widok kategorii oraz produktów



Rys. 3.3 Widok kategorii

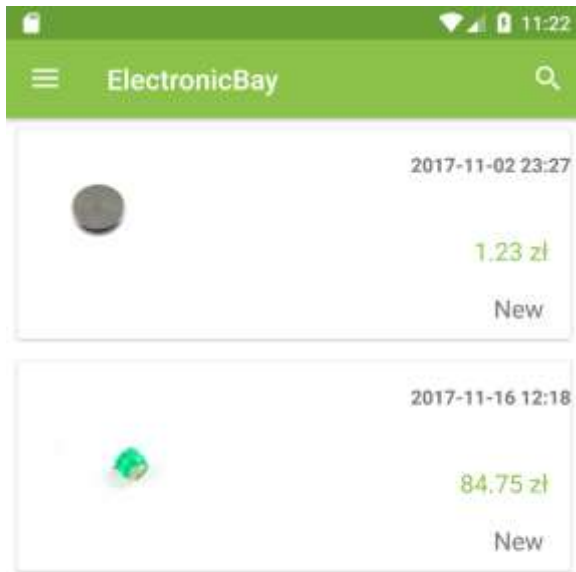


Rys. 3.4 Widok listy produktów

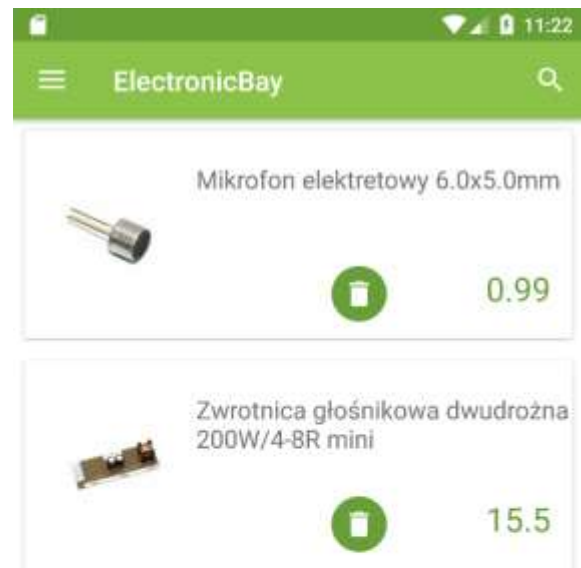
Pierwszy widok(Rys.3.3) pozwala na przegląd listy kategorii, na które zostały podzielone produkty. Po wybraniu jednej z nich można przejść do widoku zawierającego dalsze podkategorie lub do widoku produktów.

Widok produktów(Rys.3.4) zawiera listę produktów należących do wybranej kategorii. Pozwala na sortowanie danej listy ze względu na kolejność alfabetyczną lub cenę malejąco oraz rosnąco. Po wybraniu produktu klient zostaje przeniesiony do widoku szczegółów wybranego produktu. Z poziomu obu widoków mamy dostęp do widoku koszyka przy pomocy wiszącego przycisku. Użytkownik ma również dostęp do menu nawigacyjnego oraz możliwości wyszukiwania produktów. Obie te akcje dostępne są w górnym pasku akcji.

### 3.3. Widok listy ulubionych produktów oraz listy złożonych zamówień



Rys. 3.5 Widok listy złożonych zamówień

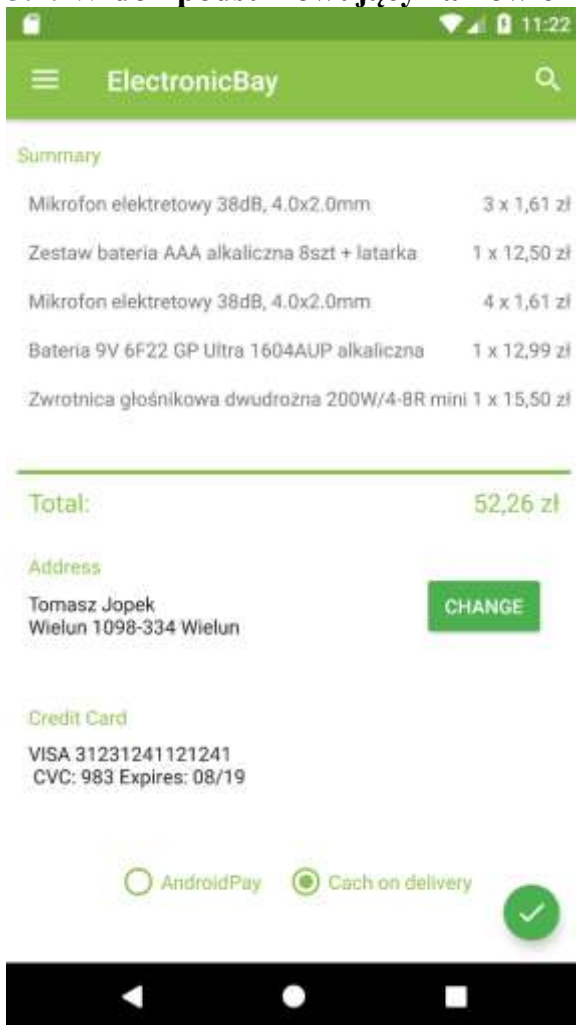


Rys. 3.6 Widok listy ulubionych

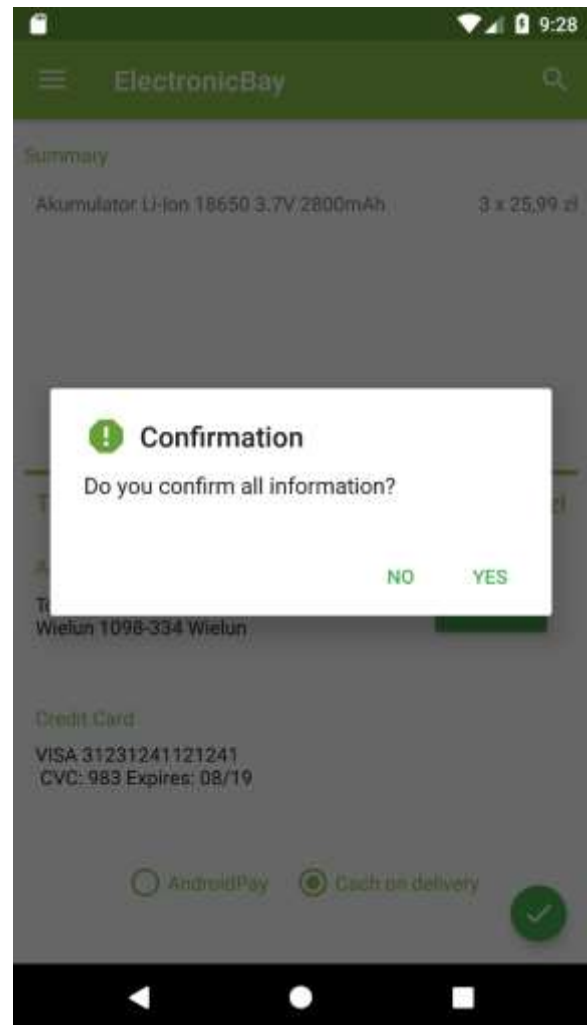
Widok listy ulubionych(Rys.3.6) zawiera wszystkie produkty, które zostały wybrane przez użytkownika z powodu ich częstego kupna, chęci otrzymywania powiadomień o zmianach w cenie danego produktu lub by mieć do niego szybki i łatwy dostęp. W widoku tym użytkownik może usunąć produkt z listy, przejść do widoku koszyka.

Jeśli chodzi o widok złożonych zamówień(Rys.3.5) prezentuje on listę zamówień dokonanych przez użytkownika wraz z jego szczegółami takimi jak data dokonania zamówienia, łączna cena zakupionych produktów, status w jakim dane zamówienie się znajduje oraz listę produktów, które znalazły się w tym zamówieniu. Z poziomu tego widoku użytkownik może także przejść do widoku koszyka, otworzyć boczne menu nawigacyjne oraz skorzystać z funkcjonalności wyszukiwania produktów w całym sklepie.

### 3.4. Widok podsumowujący zamówienie



Rys. 3.7 Widok podsumowujący zamówienie



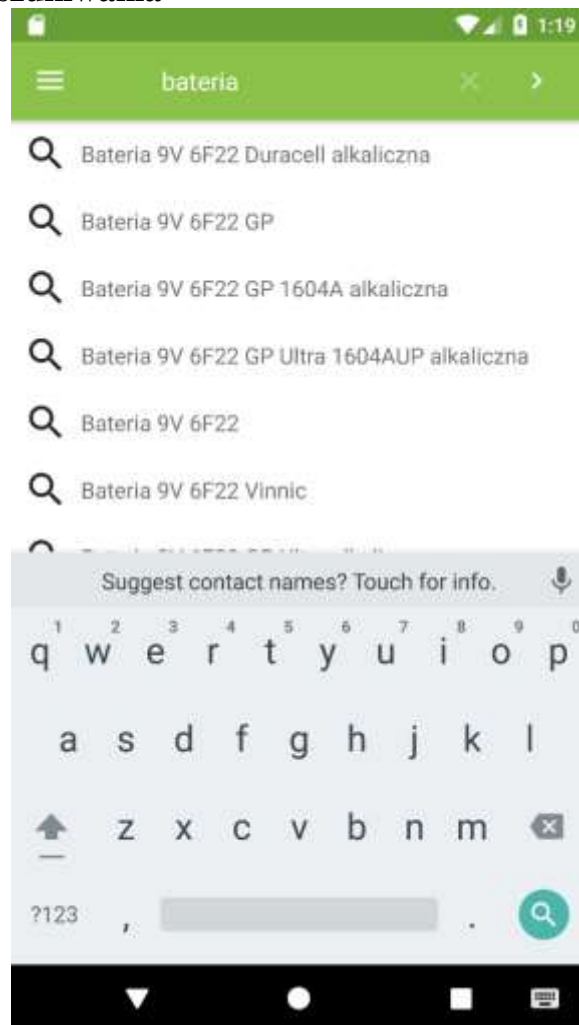
Rys. 3.8 Potwierdzenie zamówienia

Widok podsumowujący(Rys.3.7) składane zamówienie dostarcza informacji na temat produktów wchodzących w skład całego zamówienia, łącznej wartości zamówienia. Wyświetla domyślny adres zdefiniowany podany przez użytkownika, pod którego należy dostarczyć zamówienie oraz umożliwia jego zmianę, wyświetla także informacji na temat karty kredytowej, jeśli została podana przez użytkownika. Po wybraniu opcji zapłaty pojawia się przycisk umożliwiający złożenie zamówienia. Po jego kliknięciu pojawia się okno dialogowe, z pytaniem o ostateczne potwierdzenie zamówienie(Rys.3.8).

### 3.5. Widok szczegółów zamówienia oraz wyszukiwania



Rys. 3.9 Widok szczegółów produktu

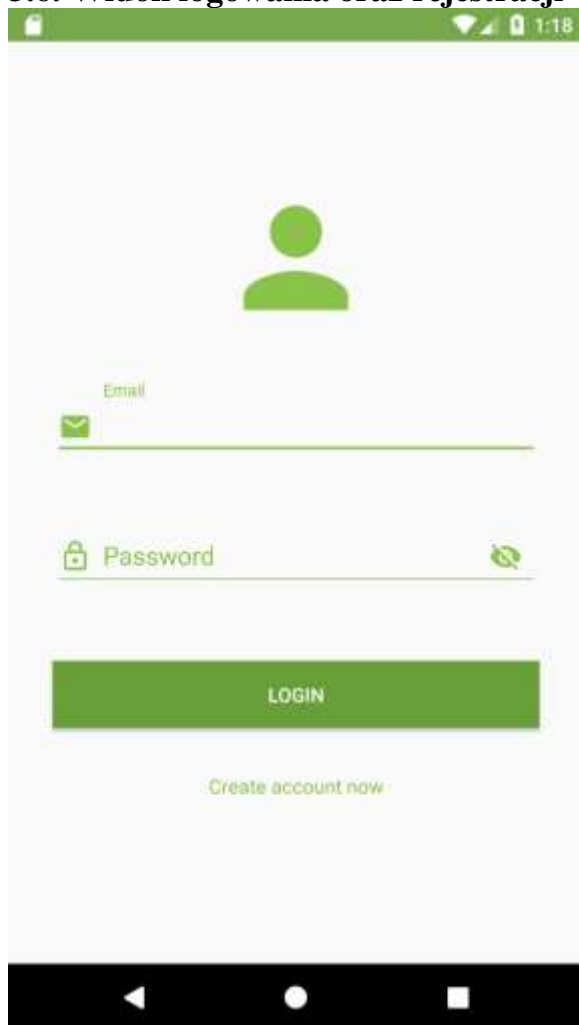


Rys. 3.10 Widok fragmentu wyszukiwania

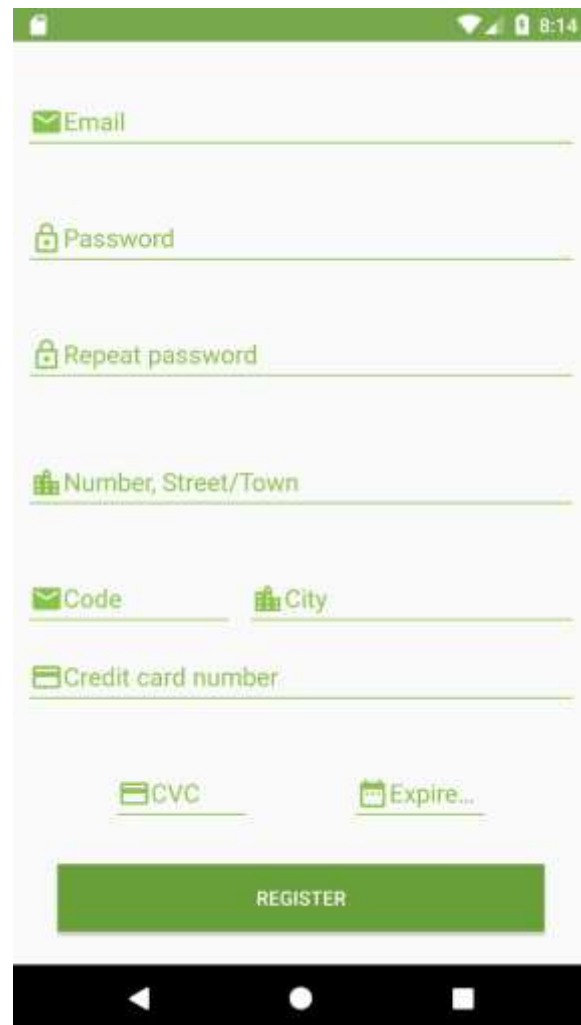
Widok szczegółów produktu (Rys. 3.9) pozwala na dostęp do szczegółowego opisu wybranego przedmiotu, możliwości dodania go do listy towarów ulubionych oraz dodanie wybranej liczby wybranego elementu do koszyka.

W widoku wyszukiwania (Rys. 3.10) uaktywnia się pole tekstowe na górnym pasku, które umożliwia wpisywanie szukanej frazy użytkownikowi. Wyszukiwanie jest zaimplementowane w następujący sposób: kolejne podpowiedzi aktualizują się użytkownikowi co trzysta milisekund, po wpisaniu minimum dwóch znaków. Klikając na propozycję z listy, użytkownik jest przekierowywany do widoku szczegółów wybranego produktu. Jeśli zatwierdzi wpisaną przez siebie frazę następuje przekierowanie do widoku produktów, które spełniają kryteria wyszukiwania.

### 3.6. Widok logowania oraz rejestracji



Rys. 3.11 Widok logowania



Rys. 3.12 Widok rejestracji

Widok logowania(Rys.3.11) zawiera formularz umożliwiający zalogowanie się do aplikacji przy użyciu podanego przy rejestracji adresu email oraz hasła. Pozwala także na przejście do widoku aktywności rejestracji. Widok rejestracji(Rys.3.12) również posiada rozbudowany formularz pozwalający na podanie podstawowych informacji użytkownika podczas zakładania konta.



### 3.7. Widok koszyka



Rys. 3.13 Widok koszyka

Widok koszyka(Rys.3.13) pozwala na sprawdzenie aktualnego stanu kompletowanego zamówienia. Widzimy listę produktów wraz z ich nazwą, ceną oraz liczbą sztuk wchodzących w skład zamówienia. Użytkownik może usunąć pozycję z koszyka poprzez przycisk z ikoną kosza, dostępny w każdym elemencie listy. Wiszący przycisk w tym widoku przemienia swoją ikonę oraz pozwala na potwierdzenie składu zamówienia, a następnie przekierowuje użytkownika do widoku podsumowującego.

## **4. Implementacja**

### **4.1. Wprowadzenie**

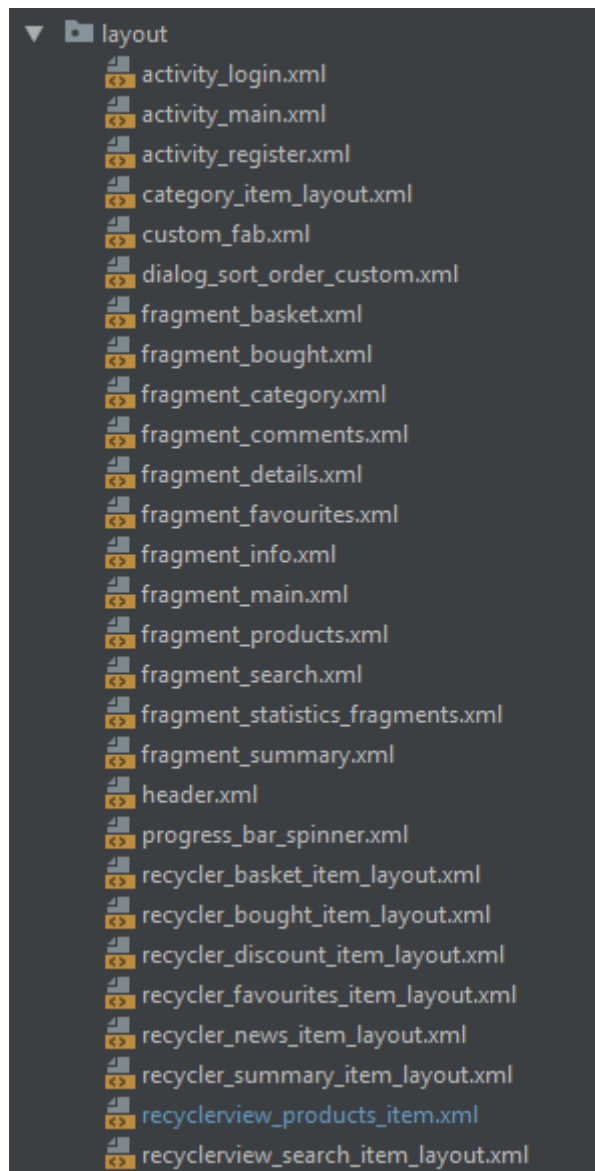
Cała aplikacja była budowana z wykorzystaniem wzorca architektonicznego MVP(Model, View, Presenter). Projekt powstał głównie w oparciu o paradygmat programowania obiektowego, który wspiera język Java. Dzięki wykorzystaniu pomocniczych bibliotek takich jak Retrolambda, RxJava pewna część kodu została oparta także o paradygmat programowania funkcyjnego. Kod powstały w ten sposób ma jasną i przejrzystą strukturę oraz jest prosty w interpretacji. Aplikacja do poprawnego działania wymaga urządzenia mobilnego z systemem Android wersji 5.0 lub wyższej. Kod aplikacji napisany jest zgodnie ze standardami języka Java wersji 7, przy uwzględnieniu technik dobrego i czytelnego programowania opisanego w następującej książce[1].

### **4.2. Środowisko i narzędzia programistyczne**

Środowisko wykorzystane przy implementacji to IDE Android Studio. Wspiera ono w pełni proces implementacji aplikacji mobilnych. Pozwala na sprawne debugowanie oraz monitorowanie zachowań rozwijanej aplikacji. W projekcie została wykorzystana baza danych czasu rzeczywistego firmy Google – Firebase. Wyżej wymienione środowisko udostępnia prostą integrację poszczególnych funkcjonalności danej bazy danych. Podczas pracy nad projektem wykorzystywano także system kontroli wersji Git udostępniany przez firmę Microsoft oraz jego wtyczkę dla Android Studio by mieć łatwy dostęp i kontrolę nad repozytorium. Jeśli chodzi o modelowanie i tworzenie diagramów UML został wykorzystany program Visual Paradigm.

### **4.3. Widoki, komponenty**

Aplikacja dedykowana na platformę Android składa się z aktywności i widoków. Aktywności to jedne z głównych elementów tworzących aplikację. Reprezentują pojedynczy widok programu choć nie jest to regułą. Ich zadaniem jest realizowanie komunikacji z użytkownikiem oraz generowanie okna aplikacji. Widoki to obiekty tworzące interfejs użytkownika. Są definiowane przy pomocy plików xml. W aplikacji zostały zdefiniowane widoki poszczególnych fragmentów oraz elementów list(Rys.4.1). Tworzenie widoków oraz własnych, nietypowych elementów zostało przedstawione w następujących pozycjach literaturowych[2][3][5].



Rys. 4.1 Widoki zaimplementowane w aplikacji

Widoki były tworzone przy wykorzystaniu ConstraintLayout, standardowych elementów widoków dostępnych w AndroidSDK. Układ elementów, rozmiary czcionek, odstępów, cienie oraz inne elementy konfiguracyjne poszczególnych elementów zostały zdefiniowane zgodnie z przewodnikiem stylów udostępnionym przez Google – Material Design[8]. Dzięki zastosowaniu tych środków znacząco zwiększają się walory UX użytkowników.

#### 4.4. Implementacja wzorca MVP

W celu implementacji wzorca architektonicznego MVP, w omawianej aplikacji tworzone następujące klasy oraz interfejsy w poszczególnych pakietach, zawierających klasy odpowiedzialne za interakcje z użytkownikiem. Przykładowy pakiet odpowiedzialny za wyświetlanie listy produktów zawiera klasę ProductsFragment dziedziczącą po klasie BaseFragment z pakietu base, która dziedziczy również po klasie Fragment należącej do frameworka Androida. Ponadto implementuje interfejs ProductsView, który posłuży do udostępniania wybranych akcji na widoku z poziomu klasy Presentera. W danym pakiecie następnie występuje interfejs ProductsPresenter, który jest implementowany przez klasę

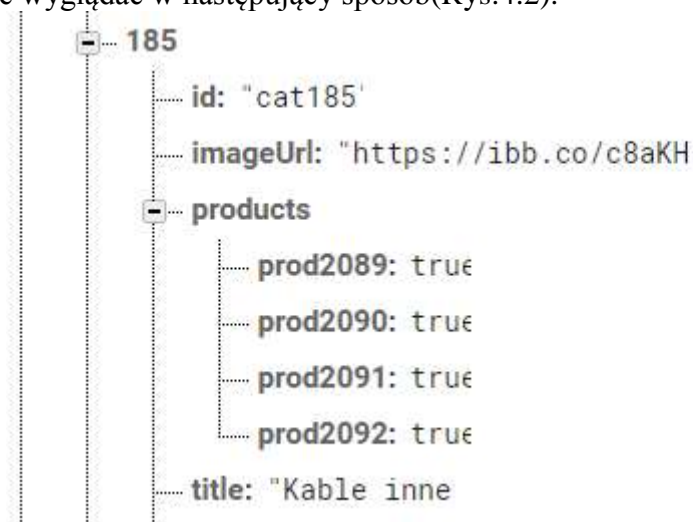
ProductsPresenterImpl dziedziczącą po klasie abstrakcyjnej z pakietu base. Ostatecznie tworzony jest interfejs ProductsInteractor oraz klasa go implementująca ProductsInteractorImpl. Tak stworzona struktura odzwierciedla poszczególne elementy występujące w przedstawionym wzorcu. Warstwę widoku tworzy klasa ProductsFragment wraz z interfejsem, warstwę prezentera klasa ProductsPresenterImpl oraz warstwę modelu ProductsInteractorImpl. Klasa ProductsFragment jako jedyna używa obiektów z Android SDK oraz posiada referencję do obiektu prezentera. Prezenter zajmuje się jedynie filtrowaniem, mapowaniem, przygotowywaniem danych oraz odpowiednim powiadamianiem warstwy widoku o zmianach. Posiada on referencję zarówno do fragmentu oraz interaktora. Interaktor natomiast posiada referencję do obiektu prezentera. Kontrola nad udostępnianymi funkcjonalnościami odbywa się dzięki zastosowaniu wcześniej omawianych interfejsów.

#### 4.5. Baza danych

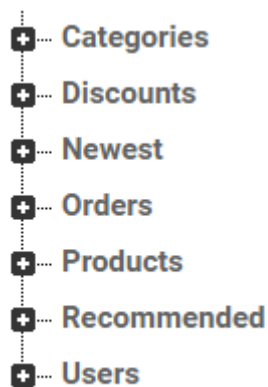
W projekcie została wykorzystana baza danych czasu rzeczywistego Firebase. Jest to baza danych hostowana w chmurze jako usługa backend as a service(BaaS). Wspiera synchronizację danych pośród wszystkich klientów, którzy subskrybują daną instancję serwera. Firebase dostarcza API dla wielu platform oraz frameworków takich jak na przykład AngularJS, iOS X, Ruby, Node.js, Java, Python oraz również Android.

Dane w omawianej bazie danych przechowywane są jako jeden, wielki dokument JSON. Jest to najczęściej używany format danych dla baz NoSQL. Stosują go również na przykład bazy danych MongoDB, Cassandra, CouchDB. Dane w dokumencie są zorganizowane jako pary klucz – wartość. Kluczem może być wartość typu String, Number lub inny złożony obiekt. Firebase dostarcza przyjaznego GUI użytkownikowi by sprawnie zarządzać oraz kontrolować akcje wykonywane na bazie.

W bazie NoSQL by zdefiniować relacje pomiędzy obiektami oraz zwiększyć wydajność naszej aplikacji, należy zastosować się do kilku zasad, które pozwolą na dokonywanie efektywnych operacji czytania oraz pisania do bazy. Przykładowo modelowanie relacji może wyglądać w następujący sposób(Rys.4.2).



Rys. 4.2 Modelowanie powiązań w bazie NoSQL



Rys. 4.3 Główna struktura bazy danych

## 4.6. Dodatkowe biblioteki

### Glide

Glide to szybka i wydajna biblioteka open source, pozwalająca na proste i sprawne zarządzanie mediami oraz obrazami, przy implementacji aplikacji skierowanych na system Android, tworzona przez Sama Judda. Oferuje gotowe rozwiązania w zakresie dekodowania mediów, zarządzanie pamięcią oraz cachem[6].

### Butterknife

Biblioteka tworzona przez Jack'a Burtona pozwalająca na łączenie elementów interfejsu definiowanych w pliku xml z obiektami definiowanymi w kodzie Javy. Pozwala zaoszczędzić sporą ilość czasu, linii kodu oraz zachować czytelność i przejrzystość implementowanej klasy[7].

### Reactive Extensions – RxJava

Dana biblioteka pozwala na programowanie reaktywne, które jest rozszerzeniem wzorca obserwatora. Udostępnia dwa główne typy:

- Observable – dostarcza dane, zajmuje się także ich wcześniejszym filtrowaniem, mapowaniem.
- Observer – przetwarza dane.

Dzięki zastosowaniu tej biblioteki możemy przetwarzać dane, zdarzenia w postaci strumieni. Obiekt typu Observable dostarcza strumień danych, a obiekt typu Observer obsługuje kolejne elementy jeden po drugim. W przeciwieństwie do standardowego wywoływania metod – nie wywołujemy metody i czekamy na rezultat jej wykonania, tylko czekamy i od razu reagujemy. Biblioteka ta jest kompatybilna z wyrażeniami lambda co pozwala na uzyskanie krótkiego, przejrzystego kodu, który może obsługiwać złożony łańcuch przetwarzania.

### Retrolambda

Biblioteka umożliwiająca wykorzystanie składni wyrażeń lambda wprowadzonych w wersji Java 8, w implementacji opierającej się na Javie 7.

## 4.7. Wzorce projektowe

### Singleton

Singleton to jeden z najprostrzych wzorców projektowych. Należy do grupy wzorców kreacyjnych i dostarcza możliwości stworzenia tylko jednej instancji klasy w całym

programie. Wzorzec ten zakłada istnienie pojedynczej klasy odpowiedzialnej za stworzenie obiektu oraz zapewnienie istnienia jedynie jednej instancji poprzez wykorzystanie prywatnego konstruktora, statycznego pola oraz statycznej metody zwracającą żądany obiekt(Rys.4.4).

```
private final DatabaseReference mDatabaseRef;
private static DatabaseHelper mInstance;

private DatabaseHelper() { mDatabaseRef = FirebaseDatabase.getInstance().getReference(); }

public synchronized static DatabaseHelper getInstance() {
    if(mInstance == null)
        mInstance = new DatabaseHelper();

    return mInstance;
}
```

*Rys. 4.4 Implementacja klasy DatabaseHelper*

### ViewHolder

Wzorzec służący optymalizacji tworzenia oraz podpinania widoków poszczególnych złożonych elementów list. Używany w klasach adapterów. ViewHolder to nie tylko klasa służąca do przechowywania wartości poszczególnych elementów kolekcji będącej źródłem danych dla adaptera. Pozwala na powtórne wykorzystanie już raz stworzonego elementu kolekcji(Rys.4.5).

```
public class ViewHolder extends RecyclerView.ViewHolder {

    @BindView(R.id.products_recycler_item_thumb)
    ImageView thumb;
    @BindView(R.id.products_recycler_item_title)
    TextView title;
    @BindView(R.id.products_recycler_item_price)
    TextView price;
    @BindView(R.id.products_recycler_item_availability)
    TextView count;

    public ViewHolder(View itemView) {
        super(itemView);
        ButterKnife.bind(target this, itemView);

        itemView.setOnClickListener(v -> productsView.productClickHandler(dataSet.get(getAdapterPosition())));
    }
}
```

*Rys. 4.5 Implementacja wzorca ViewHolder*

### Obserwator

Wzorzec obserwatora używany jest pośrednio poprzez wykorzystanie biblioteki RxJava, która go wykorzystuje i rozszerza. Założeniem, intencją wzorca obserwatora jest zdefiniowanie relacji jeden do wielu, pomiędzy obiektami by zmiana jednego, powiadomiła o tym resztę(Rys.4.6).

```

public class RxSearch {

    public static Observable<String> fromSearchView(@NonNull final SearchView searchView) {
        final BehaviorSubject<String> subject = BehaviorSubject.create();

        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {
                subject.onComplete();
                return true;
            }

            @Override
            public boolean onQueryTextChange(String newText) {
                if(!newText.isEmpty() && newText.length() > 2)
                    subject.onNext(newText);

                return true;
            }
        });

        return subject;
    }
}

```

Rys. 4.6 Implementacja obiektu *Observable*, generującego strumień zmian tekstu w polu wyszukiwania

## 4.8. Problemy implementacyjne

### Zarządzanie fragmentami

Jednym z głównych komponentów tworzących aplikację są aktywności jak wspomniano w rozdziale powyżej. Tworzą interfejs graficzny użytkownika oraz posiadają własny cykl życia. Fragmenty zostały wprowadzone w wersji Androida 3.0(Honeycomb). Obiekty te są doskonałym narzędziem, które przydaje się w wielu sytuacjach, w których do tej pory tylko klasy Activity były dostępne. Dzięki fragmentom, które również posiadają swój własny cykl życia, nieco różniący się od cyklu życia aktywności, możliwe stało się tworzenie aplikacji modularnych. W opracowywanej aplikacji główna aktywność zawiera kontener dla fragmentów co pozwala na korzystanie z jednego paska wyszukiwania – SearchView w całej aplikacji oraz bocznego paska menu zdefiniowanego jako DrawerLayout. Jednak wraz z używaniem fragmentów wiąże się także między innymi zarządzanie ich cyklem życia oraz implementacją nawigacji i komunikacji pomiędzy poszczególnymi fragmentami. Z wymienionych przyczyn prawidłowa implementacja wykorzystująca fragmenty wiązała się z odpowiednim nadpisaniem metody `onBackPressed` głównej aktywności, definicji obiektów, które miały być przekazywane pomiędzy fragmentami, jako implementujące interfejs `Parcelable`. Do zarządzania fragmentami i całymi transakcjami z nimi związanymi posłużyły klasy oferowane przez Android SDK takie jak `FragmentManager` oraz `FragmentTransaction`.

### Zawieszanie się interfejsu graficznego

Wykonywanie aplikacji odbywa się na jednym głównym wątku. Wykonywanie operacji długotrwałych takich jak połączenia internetowe, operacje pobierania danych i ogólnie manipulowania na bazie danych, czy operacje wymagające dużo czasu wykonania ze względu na operacje obliczeniowe powodują zawieszanie, blokowanie głównego wątku, a co z tego wynika interfejsu graficznego użytkownika, sprawiając bardzo złe wrażenia UX użytkownika. Rozwiązaniem danego problemu jest przeprowadzenie wyżej wymienionych

operacji zabierających znaczną część czasu w tle, przy użyciu innego wątku. Operacje takie można wykonać przy użyciu serwisu, klasy AsyncTask oraz przy pomocy wcześniej wspomnianej biblioteki RxJava. W prezentowanym rozwiązaniu zdecydowano się na wykorzystanie biblioteki RxJava, która pozwala na proste zarządzanie, tworzenie nowych wątków oraz kontrolę nad ich użyciem.

#### Obsługa zdarzeń asynchronicznych

Wykorzystana w implementacji baza danych dostarcza API pozwalające na budowanie zapytań, łączenie się z bazą, oraz przetwarzanie pobranych informacji. Metody, listenery, które udostępnia działają asynchronicznie. Do obsługi tych zadań wykorzystano zdefiniowane Listenery, które były implementowane przez poszczególne klasy presenterów.

#### Responsywność

W celu uzyskania interfejsów responsywnych, renderujących się poprawnie na urządzeniach mobilnych wyposażonych w różnej wielkości i rozdzielczości ekrany, zastosowano przy tworzeniu widoków – ConstraintLayout.



## 5. Podsumowanie

### 5.1. Podsumowanie pracy

Celem niniejszej pracy inżynierskiej było zaprojektowanie oraz implementacja aplikacji natywnej dla systemu Android wersji 5.0 oraz wyższych, świadczącej funkcjonalność sklepu mobilnego z asortymentem elektronicznym. Miała ona za zadanie wspieranie działającego sklepu internetowego. Motywacją do opracowania omawianej aplikacji był fakt istnienia dużej liczby aplikacji mobilnych o podobnym charakterze jednak nie w branży elektronicznej. Badanie opinii użytkowników korzystających z usług sklepów internetowych z elementami elektronicznymi wykazało, iż witryny z których korzystają nie są przystosowane do prezentaowania w dogodny sposób treści na urządzeniach mobilnych. Wyrazili oni zatem szczerą chęć powstania oraz używania aplikacji o takim charakterze.

Aplikację udało się zrealizować w swojej podstawowej wersji. Pozwala na logowanie, rejestrację użytkowników. Umożliwia użytkownikom przeglądanie dostępnego asortymentu sklepu, pozwala na definiowanie listy ulubionych produktów, dokonywanie zakupów poprzez skompletowanie zamówienia w koszyku a następnie jego potwierdzenie i realizację. Interfejs aplikacji dostarcza bieżących i aktualnych danych ze strony serwisu dzięki zastosowaniu bazy danych czasu rzeczywistego Firebase.

### 5.2. Plany rozwoju aplikacji

W obecnych czasach podstawową formą płatności stała się już płatność elektroniczna. Codziennie miliony użytkowników korzystają z kart płatniczych, przelewów elektronicznych, BLIK'a, serwisów typu PayPal obsługujących płatności. W przedstawionym projekcie nie udało się zrealizować możliwości dokonywania zapłaty poprzez płatność elektroniczną z wykorzystaniem AndroidPay API, które wspiera już takie serwisy jak

- Adyen,
- Braintree,
- PaySafe,
- Stripe,
- Vantiv,
- WorldPay.

Ze względu na użycie w projekcie starszych wersji na przykład Google Play Services zadanie to jest niemożliwe do wykonania bez użycia nowszych technologii. Wiąże się to niestety z koniecznością, użycia nowszych bibliotek, co wymaga zmiany części kodu źródłowego. Ramy czasowe, które zostały wyznaczone na realizację projektu uniemożliwiły dokonanie wymienionego celu, a udostępnienie użytkownikom dokonywania płatności elektronicznych jest obecnie standardem i powinno zostać zrealizowane w pierwszej kolejności jeśli chodzi o kolejne funkcjonalności.

Należy również zaimplementować system ocen i komentarzy użytkowników odnośnie zakupionych przez nich towarów oraz statystyk prezentujących trend sprzedaży, ceny poszczególnych towarów. Prezentowane wyżej wymienione statystyki w przystępny użytkownikowi sposób mogły by go skłonić do kupna w większej ilości lub po prostu na wybranie konkretnego przedmiotu.

## Spis ilustracji

Rys. 1.1 Liczba użytkowników smartfonów na świecie w latach 2014-17(w miliardach).....	4
Rys. 1.2 Liczba pobrań aplikacji ze sklepu Google Play .....	5
Rys. 1.3 Architektura systemu Android .....	7
Rys. 1.4 Widok startowy aplikacji allegro .....	9
Rys. 1.5 Widok listy produktów.....	9
Rys. 1.6 Widok główny aplikacji .....	10
Rys. 1.7 Widok listy produktów.....	10
Rys. 2.1 Diagram przypadków użycia .....	13
Rys. 2.2 Wzorzec architektoniczny MVP - Model View Presenter .....	16
Rys. 2.3 Diagram pakietów .....	17
Rys. 2.4 Struktura pakietu ui .....	18
Rys. 3.1 Widok ekranu startowego aplikacji.....	19
Rys. 3.2 Widok bocznego menu.....	19
Rys. 3.3 Widok kategorii.....	20
Rys. 3.4 Widok listy produktów.....	20
Rys. 3.5 Widok listy złożonych zamówień .....	21
Rys. 3.6 Widok listy ulubionych.....	21
Rys. 3.7 Widok podsumowujący zamówienie .....	22
Rys. 3.8 Potwierdzenie zamówienia .....	22
Rys. 3.9 Widok szczegółów produktu.....	23
Rys. 3.10 Widok fragmentu wyszukiwania .....	23
Rys. 3.11 Widok logowania .....	24
Rys. 3.12 Widok rejestracji .....	24
Rys. 3.13 Widok koszyka.....	25
Rys. 4.1 Widoki zaimplementowane w aplikacji .....	27
Rys. 4.2 Modelowanie powiązań w bazie NoSQL.....	28
Rys. 4.3 Główna struktura bazy danych.....	29
Rys. 4.4 Implementacja klasy DatabaseHelper .....	30
Rys. 4.5 Implementacja wzorca ViewHolder.....	30
Rys. 4.6 Implementacja obiektu Observable, generującego strumień zmian tekstu w polu wyszukiwania .....	31

## Bibliografia

1. Robert C. Martin, *Clean Code*, Boston, Pearson Education Inc., 2009, ISBN 0-13-235088-2
2. Griffiths, Down, *Android: programowanie aplikacji*, Gliwice, Helion, 2016, ISBN 978-83-283-2063-5
3. Annuzzi Joseph, *Android: wprowadzenie do programowania aplikacji*, Gliwice, Helion, 2016, ISBN 978-83-283-2612-5
4. Bruegge Bernd, *Inżynieria oprogramowania w ujęciu obiektowym: UML, wzorce projektowe i Java*, Gliwice, Helion, 2011, ISBN 978-83-246-2872-8
5. Android, dokumentacja, [dostęp: 5 grudnia 2017]  
<https://developer.android.com/index.html>
6. Biblioteka Glide, dokumentacja, [dostęp: 5 grudnia 2017]  
<https://bumptech.github.io/glide/>
7. Biblioteka ButterKnife, dokumentacja, [dostęp 5 grudnia 2017]  
<http://jakewharton.github.io/butterknife/>
8. Przewodnik stylów, Material Design, [dostęp: 5 grudnia 2017]  
<https://material.io/guidelines/>
9. ThinkMobile, *MVP vs MVVM: A Review of Architectural Patterns for Android* [dostęp: 5 grudnia 2017]  
<https://thinkmobiles.com/blog/mvp-vs-mvvm-android-patterns/>
10. GSM Online, [dostęp 5 grudnia 2017]  
<http://gsmonline.pl/artykuly/idc-sprzedaz-smartfonow-i-kw-2017-w-polsce>
11. Statista, [dostęp 5 grudnia 2017]  
<https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/>
12. Moskat, *Naprawdę krótka historia informatyki* [dostęp: 5 grudnia 2017]  
[http://www.moskat.pl/szkola/informatyka/zz\\_historia.php](http://www.moskat.pl/szkola/informatyka/zz_historia.php)
13. Statista, [dostęp 5 grudnia 2017]  
<https://www.statista.com/statistics/281106/number-of-android-app-downloads-from-google-play/>
14. Muntenescu Florina, *Android Architecture Patterns Part 2: Model-View-Presenter* [dostęp 5 grudnia 2017]  
<https://medium.com/upday-devs/android-architecture-patterns-part-2-model-view-presenter-8a6faae14a5>