

# AKADEMIA NAUK STOSOWANYCH W NOWYM SĄCZU

Wydział Nauk Inżynierskich  
Katedra Informatyki

## DOKUMENTACJA PROJEKTOWA PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

### **Menadżer wydatków**

Autor:  
Marek Pichniarczyk  
Mikołaj Kwiatek  
Karol Wolski

Prowadzący:  
mgr inż. Dawid Kotlarski

Nowy Sącz 2022

# Spis treści

<b>1. Ogólne określenie wymagań</b>	<b>3</b>
<b>2. Określenie wymagań szczegółowych</b>	<b>4</b>
2.1. Opis aplikacji . . . . .	4
2.2. Schemat . . . . .	4
2.3. Wykorzystywane technologie . . . . .	6
2.3.1. Android . . . . .	6
2.3.2. Java . . . . .	6
2.3.3. Android Studio . . . . .	6
2.3.4. Firebase . . . . .	7
2.3.5. Github . . . . .	8
<b>3. Projektowanie</b>	<b>9</b>
<b>4. Implementacja</b>	<b>14</b>
4.1. Firebase Auth . . . . .	14
4.2. Baza danych . . . . .	17
4.3. Tryb ciemny . . . . .	17
4.4. Nawigacja . . . . .	20
4.5. Dane biometryczne . . . . .	22
4.6. Text Spinner - wysuwane pole wyboru . . . . .	25
<b>5. Testowanie</b>	<b>27</b>
<b>6. Podręcznik użytkownika</b>	<b>29</b>
<b>Literatura</b>	<b>41</b>
<b>Spis rysunków</b>	<b>41</b>
<b>Spis tabel</b>	<b>42</b>
<b>Spis listingów</b>	<b>43</b>

## 1. Ogólne określenie wymagań

Aplikacja ma za zadanie podsumowywać wydatki użytkownika. Klient będzie mógł wprowadzać dane dotyczące danego wydatku oraz podporządkować je kategorii (np. spożywcze/transport/podatki). Ma istnieć możliwość dodania także miesięcznych przychodów z których odejmowane będą wydatki. Do każdej pozycji będzie możliwość dodania zdjęcia paragonu lub faktury. Program użytkowy ma obliczać statystyki z okresu. Użytkownik będzie miał możliwość wprowadzenia dziennego limitu wydatków, po przekroczeniu zostanie o tym poinformowany. Wraz z końcem dnia aplikacja wyśle podsumowanie o wydanych przez użytkownika wydatkach. Użytkownik ma mieć możliwość wyboru trybu ciemnego lub jasnego. Odblokowanie aplikacji ma być za pomocą danych biometrycznych, głównym celem aplikacji ma być osobiste kontrolowanie wydatków oraz możliwość ich podporządkowania do różnych kategorii by mieć wgląd w różne pozycje. Ważną funkcją w całej aplikacji ma być wgląd do ostatnich wydatków, by móc zobaczyć na co ostatnio wydaliśmy pieniądze oraz szczegóły ich, tzn. data, kategoria oraz kwota. Aplikacja ma być bardzo prosta w obsłudze, bez niepotrzebnych funkcji, aczkolwiek potrzebny będzie tryb ciemny oraz jasny aplikacji, ponieważ niektórzy użytkownicy preferują tryb ciemny, który nie razi w oczy w wieczornych porach, ale również musi być możliwość ustawienia trybu jasnego. Baza danych musi być umieszczona w sieci, ponieważ użytkownik musi mieć możliwość korzystania z programu na różnych urządzeniach oraz mieć możliwość integracji w wydatki realtime - tzn. użytkownik na jednym telefonie może dodać wydatek, a na drugim w tym samym momencie, bez odświeżania aktywności ma się wyświetlić ten wydatek.



Rys. 1.1. Logo

## 2. Określenie wymagań szczegółowych

### 2.1. Opis aplikacji

Aplikacja będzie pisana w środowisku Android Studio językiem Java. Celem aplikacji ma być kontrolowanie wydatków użytkownika, zakresem działań będą między innymi dodanie swojego dochodu z którego będziemy odejmować wydatki i wyliczać później kwotę pozostałą. Przy włączeniu aplikacji ukazuje nam się formularz logowania, poniżej jest możliwość zarejestrowania użytkownika, po pomyślnym zalogowaniu aplikacja odsyła nas do strony głównej. Saldo konta będzie wyświetlane na głównej części aplikacji, będzie też przycisk do dodania wydatku - on przekieruje nas do widoku formularza dodawania wydatku. Użytkownik będzie podawać tytuł wydatku, kwotę wydatku, kategorię (np. transport, żywność etc.), oraz zdjęcie paragonu lub faktury, możliwe będzie również sporządzenie nagrania notatki głosowej do danego wydatku, do wydatku będzie pobierana data dodania z systemu. Będzie możliwość usunięcia każdego wydatku - to wszystko będzie w głównym widoku. Podkategoria statystyka będzie wyliczała dane statystyki. Aplikacja będzie wysyłała powiadomienia o przekroczeniu limitu dziennego wydatków oraz pod koniec dnia wydanych pieniędzy. Baza danych będzie prowadzona na Firebase Realtime Database, dodana będzie możliwość również odblokowania aplikacji za pomocą danych biometrycznych. Logowanie do aplikacji będzie oparte na systemie Firebase Authentication, dzięki czemu możliwe będzie rejestracja/logowanie użytkowników przy użyciu internetu.

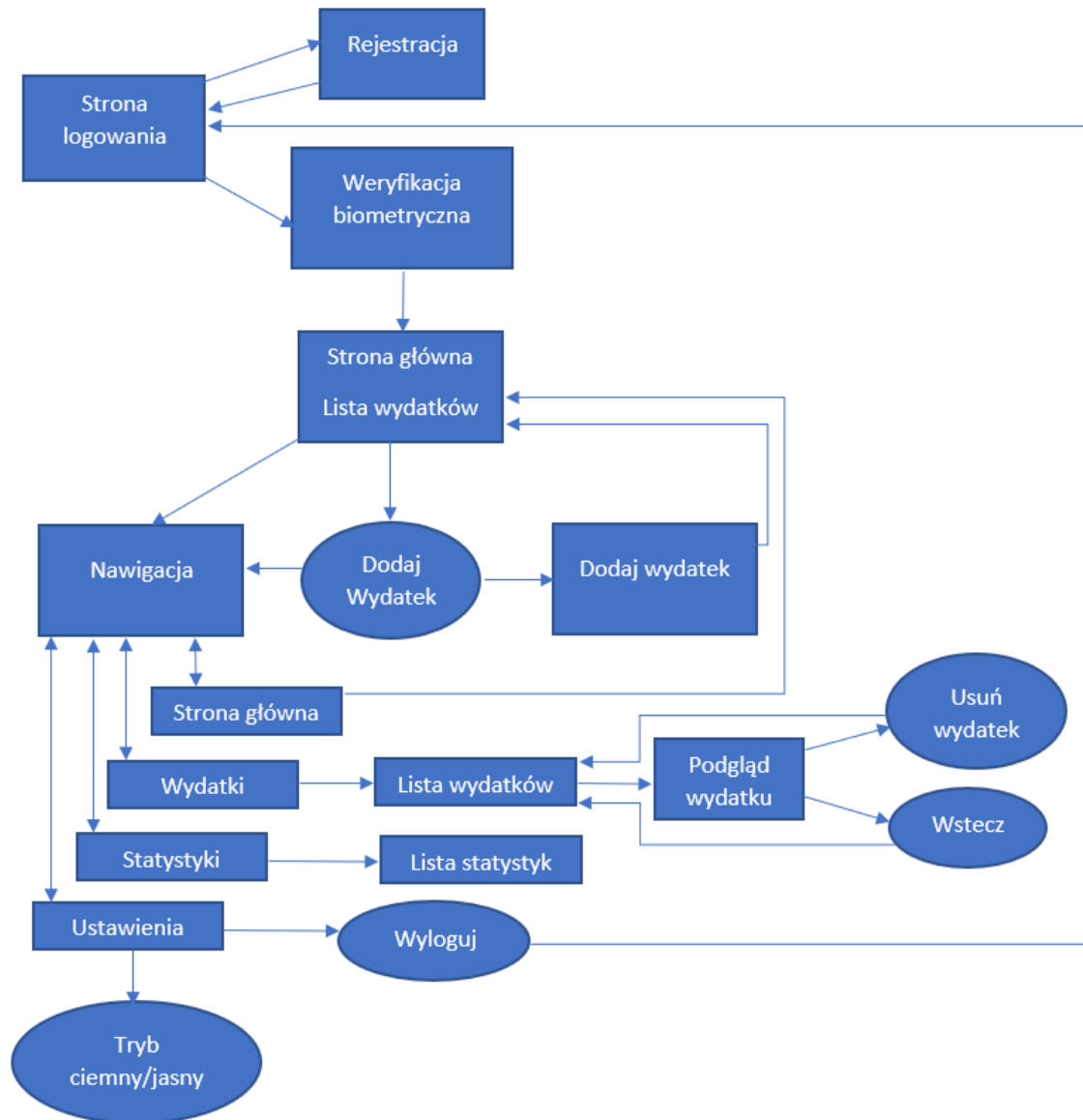
Podkategoria statystyki: suma wydatków, dochód, suma wydatków podporzątkowana kategoriom.

Podkategoria wydatki: wszystkie wydatki, możliwość usunięcia dowolnych wydatków.

Podkategoria ustawienia: przycisk switch dzięki któremy będziemy mieć możliwość przechodzenia pomiędzy trybem jasnym i ciemnym aplikacji, dane użytkownika oraz przycisk wyloguj.

### 2.2. Schemat

Rysunek 2.1 (s. 5) przedstawia schemat aplikacji. Poruszanie się po aplikacji rozpoczynamy od strony logowania, następnie możemy przejść do rejestracji lub zalogować się i zweryfikować dane biometryczne, następnie znajdujemy się na stronie głównej, tam mamy przycisk "Dodaj wydatek", dzięki niemu można przejść do ak-



Rys. 2.1. Schemat aplikacji

tywności która umożliwia nam tą funkcję, po dodaniu wydatku przenosimy się na stronę główną. Nawigacja umożliwia nam poruszanie się po aktywnościach aplikacji, przejście do "wydatki" przenosi nas do listy wydatków, a po kliknięciu w wydatek przenosimy się do podglądu tego wydatku, tam jest możliwość usunięcia wydatku oraz funkcja "wstecz", która przenosi nas z powrotem na listę wydatków. "Statystyki" ukazuje nam listę statystyk. "Ustawienia" posiada funkcję zmiany trybu jasny-/ciemny po kliknięciu, oraz przycisk "Wyloguj", który wylogowuje nas z aplikacji i przenosi do strony logowania.

## **2.3. Wykorzystywane technologie**

### **2.3.1. Android**

Android – system operacyjny z jądrem Linux dla urządzeń mobilnych takich jak telefony komórkowe, smartfony, tablety (tablety PC) i netbooki. W 2013 roku był najpopularniejszym systemem mobilnym na świecie. Wspomniane jądro oraz niektóre inne komponenty, które zaadaptowano do Androida opublikowane są na licencji GNU GPL. Android nie zawiera natomiast kodu pochodzącego z projektu GNU. Cecha ta odróżnia Androida od wielu innych istniejących obecnie dystrybucji Linuksa. Początkowo był rozwijany przez firmę Android Inc. (kupioną później przez Google), następnie przeszedł pod skrzydła Open Handset Alliance.

Android zrzesza przy sobie dużą społeczność deweloperów piszących aplikacje, które poszerzają funkcjonalność urządzeń. W sierpniu 2014 było dla tego systemu dostępnych ponad 1,3 miliona aplikacji w Google Play (wcześniej Android Market).

Według danych serwisu StatCounter z kwietnia 2017 roku Android miał największe udziały na rynku systemów operacyjnych.

### **2.3.2. Java**

Java – współbieżny, oparty na klasach, obiektowy język programowania ogólnego zastosowania. Został stworzony przez grupę roboczą pod kierunkiem Jamesa Goslinga z firmy Sun Microsystems. Java jest językiem tworzenia programów źródłowych kompilowanych do kodu bajtowego, czyli postaci wykonywanej przez maszynę wirtualną. Język cechuje się silnym typowaniem. Jego podstawowe koncepcje zostały przejęte z języka Smalltalk (maszyna wirtualna, zarządzanie pamięcią) oraz z języka C++ (duża część składni i słów kluczowych).

Autorzy języka Java określili kilkanaście kluczowych koncepcji swojego języka. Najważniejsze z nich to:

Obiektość, dziedziczenie, niezależność od architektury, sieciowość i obsługa programowania rozproszonego.

### **2.3.3. Android Studio**

Android Studio – oficjalne środowisko programistyczne na platformę Android, zbudowane na podstawie oprogramowania IntelliJ IDEA od JetBrains. Może zostać zainstalowane na systemach Windows, macOS i opartych na Linuksie. Przed wydaniem Android Studio, głównym środowiskiem programistycznym dla systemu Android było Eclipse.

Android Studio zaprezentowano 15 maja 2013 roku na konferencji Google I/O. Pierwsze stabilne wydanie (1.0) nastąpiło w grudniu następnego roku. Środowisko obsługuje języki programowania: Java, C++ a także Kotlin. 7 maja 2019 ogłoszono, że ten ostatni zastąpił Javę jako preferowany do pisania aplikacji na system Android. Nie spowodowało to usunięcia wsparcia dla któregośkolwiek z języków.

W Android Studio wbudowane jest narzędzie do projektowania widoków aplikacji w trybie WYSIWYG. Środowisko wspiera również emulator Androida, dzięki któremu aplikacje można testować bez dostępu do urządzenia mobilnego.

Instalacja Android Studio jest możliwa na komputerach wyposażonych w 64-bitowy procesor ze wsparciem dla wirtualizacji, co najmniej 8 GB pamięci RAM i tyle samo miejsca na dysku.

#### **2.3.4. Firebase**

Firebase to stworzona przez Firebase Inc. w 2011 roku platforma, która umożliwia tworzenie aplikacji mobilnych oraz internetowych. W 2014 roku została wykupiona przez Google. Gigant z Mountain View stale rozwija tę platformę, dzięki czemu użytkownicy dostają coraz lepsze narzędzie do tworzenia swoich aplikacji. Google umożliwia swoim użytkownikom stworzenie maksymalnie 10 projektów. Każdy z nich na starcie posiada darmowy plan Spark. Jest to podstawowy plan, który jest idealny na początek. Można na nim sprawdzić, czy stworzona aplikacja, będzie przyciągać odbiorców czy też nie.

Gdy zapotrzebowanie na zasoby zacznie rosnąć, może wystąpić potrzeba zakupienia planu Flame, który miesięcznie kosztuje 25 dolarów. Limity możliwe do wykorzystania przez użytkowników w tym planie znacząco zostały podniesione. Dzięki temu aplikacja dalej może się rozwijać, a programiści mogą skupiać się na dodawaniu nowych funkcjonalności.

Ostatni z dostępnych planów to Blaze, który nie posiada stałej kwoty. Osoba będąca właścicielem projektu dokonuje opłaty na podstawie wykorzystanych zasobów w danym okresie rozliczeniowym.

<https://firebase.google.com/pricing> pod tym linkiem można zobaczyć limity dla pierwszych dwóch planów oraz stawki za dane zasoby dla ostatniego planu. Po utworzeniu projektu menu zostało podzielone na cztery części: Programowanie, Jakość, Analiza oraz Rozwój. Dzięki temu całą funkcjonalność mamy rozmieszczoną w dwóch kategoriach deweloperskich (Programowanie, Jakość) oraz dwóch kategoriach marketingowych (Analiza i Rozwój). Poniżej przedstawię pokrótce, to co możemy znaleźć w każdej z kategorii.

Funkcje wykorzystane w aplikacji:

Authentication – moduł ten odpowiada za określenie sposobów autoryzacji użytkowników. Do dyspozycji mamy logowanie poprzez adres e-mail, Facebooka, Twittera oraz wiele innych opcji.

Database – odpowiada za przechowywanie wprowadzanych przez użytkowników danych, np. dane potrzebne do wysyłki zamówionego przedmiotu. Firebase udostępnia dwa typy baz danych: Realtime Database oraz Cloud Firestore.

Storage – zadaniem tego modułu jest przechowywanie plików. Użytkownik może za pomocą aplikacji przysyłać różne pliki do projektu Firebase, np. pliki z rozszerzeniem pdf.

### 2.3.5. Github

GitHub – hostingowy serwis internetowy przeznaczony do projektów programistycznych wykorzystujących system kontroli wersji Git. Stworzony został przy wykorzystaniu frameworka Ruby on Rails i języka Erlang. Serwis działa od kwietnia 2008 roku. GitHub udostępnia darmowy hosting programów open source i prywatnych repozytoriów (część funkcji w ramach prywatnych repozytoriów jest płatna). W czerwcu 2018 ogłoszono, iż serwis zostanie przejęty przez przedsiębiorstwo Microsoft za kwotę 7,5 miliarda dolarów. W maju 2019 roku GitHub informuje, że ma około 37 milionów użytkowników i więcej niż 100 milionów repozytoriów (w tym co najmniej 28 milionów repozytoriów publicznych).

Kilka z funkcji, które występują w systemie GitHub:

bugtracker,

forki repozytoriów, czyli kopia w osobnym repozytorium należąca do innego użytkownika,

pull requesty – osoba mająca forka może zgłosić swój kod do złączenia z głównym repozytorium,

statystyki,

organizacje zrzeszające programistów pracujących nad repozytoriami,

web hooks – wywołanie operacji na repozytorium wysyła informację do innego serwisu lub skryptu,

wiki dla celów dokumentacji.



### 3. Projektowanie

Aplikacja wymagała przygotowania narzędzi do rozpoczęcia programowania, rozpoczęliśmy od instalacji IDE Android Studio, który umożliwia wygodne tworzenie layoutów każdej aktywności oraz kodowanie w różnych językach programowania. Na potrzeby projektowe wybraliśmy język JAVA, również jednym z ważniejszych narzędzi które wykorzystujemy jest GitHub - hosting umożliwiający tworzenie zdalnego repozytorium oraz kontrolę wersji oprogramowania, dzięki temu możliwa jest współpraca z członkami projektu. Wykorzystujemy także Firebase - zestaw usług hostingowych dla każdego typu aplikacji. Oferuje NoSQL i hosting w czasie rzeczywistym baz danych, treści, uwierzytelnianie społecznościowe i powiadomienia lub usługi, takie jak serwer komunikacji w czasie rzeczywistym. Jako bazę danych wykorzystujemy Firebase Firestore Database. Do rejestracji i logowania używamy Firebase Authentication.

`com.google.firebase.auth.FirebaseAuth` - biblioteka umożliwiająca połączenie z systemem Firebase Auth, odpowiada za cały system logowania oraz rejestrowania nowych użytkowników. Rozwijając myśl tej biblioteki, pobieramy z formularzy id zmiennych (e-mail, hasło) oraz inicjalizujemy bazę danych. Strona logowania (pierwsza po uruchomieniu aplikacji) w pierwszej kolejności sprawdza czy użytkownik jest zalogowany, jeżeli jest to przekierowuje go na stronę główną aplikacji, jeżeli nie to zostaje na stronie do logowania, gdzie po wpisaniu danych sprawdza je z danymi serwera i loguje się gdy są poprawne dane poprzez przycisk "zaloguj", jeżeli dane są nie poprawne odpowiada błędem. przycisk "Zarejestruj" przekierowuje do aktywności w której zawarty jest formularz rejestracji, również podłączony pod `FirebaseAuth`, jeżeli rejestracja przebiegnie pomyślnie, przekierowuje do strony logowania i wyświetla komunikat o poprawnej rejestracji, jeżeli nie to wyświetla komunikat o niepoprawnej rejestracji.

Kontynuując, jeżeli logowanie przejdzie pomyślnie zanim uzyskamy dostęp do strony głównej aplikacja poprosi nas o zweryfikowanie właściciela poprzez dane biometryczne, dzięki bibliotece `androidx.biometric:biometric:1.1.0`.

`com.google.firebase.firestore.FirebaseFirestore` - biblioteka umożliwiająca obsługę bazy danych znajdującej się w Firebase, odpowiada za system przechowywania, odczytu i zapisu danych poszczególnych użytkowników. Cloud Firestore jest elastyczną, skalowalną, hierarchiczną bazą danych NoSQL w chmurze służącą do przechowywania i synchronizowania w czasie rzeczywistym danych między klientami i serwerem. Wy-

korzystujemy ją jako system do zapisywania wydatków w bazie danych dla poszczególnych użytkowników oraz do wyświetlania ich.

`com.google.firebase.firestore.Query` - biblioteka, którą wykorzystujemy do obsługi zapytań.

`FirebaseUI` ułatwia powiązanie danych z Cloud Firestore z interfejsem aplikacji.

`com.google.firebase.firestore.DocumentReference` - biblioteka daje nam możliwość odwoływania się do poszczególnych dokumentów w bazie danych.

`java.util.Calendar` - biblioteka, którą wykorzystujemy do pobierania systemowego czasu oraz daty dodania wydatku.

`android.provider.MediaStore` - biblioteka odpowiadająca za tworzenie indeksowanej kolekcji zdjęć, wykorzystujemy ją do zapisywania zdjęć po ich zrobieniu.

`android.graphics.Bitmap` - biblioteka, którą używamy do kompresji zdjęć do formatu bitmapy.

## Frontend - XAML

Wygląd naszej aplikacji jest obsługiwany przez język XAML, dzięki niemu jesteśmy w stanie ustytytuować różne elementy aplikacji w wybranym przez nas miejscu. Oczywiście każdy element możemy edytować w wybrany przez nas sposób, odpowiadają za to takie opcje jak np. `width`, `height`, `background`, `textColor`, `textStyle`, `textSize`, `paddingi` oraz `marginesy`, również dla każdego elementu przypisujemy `id`.

Elementy, które wykorzystujemy:

`Button` - wywołuje przycisk.

`ImageView` - tutaj wyświetlamy obraz.

`EditText` - tworzy pole do wpisania tekstu.

`TextView` - wyświetla tekst.

`Spinner` - lista rozwijana

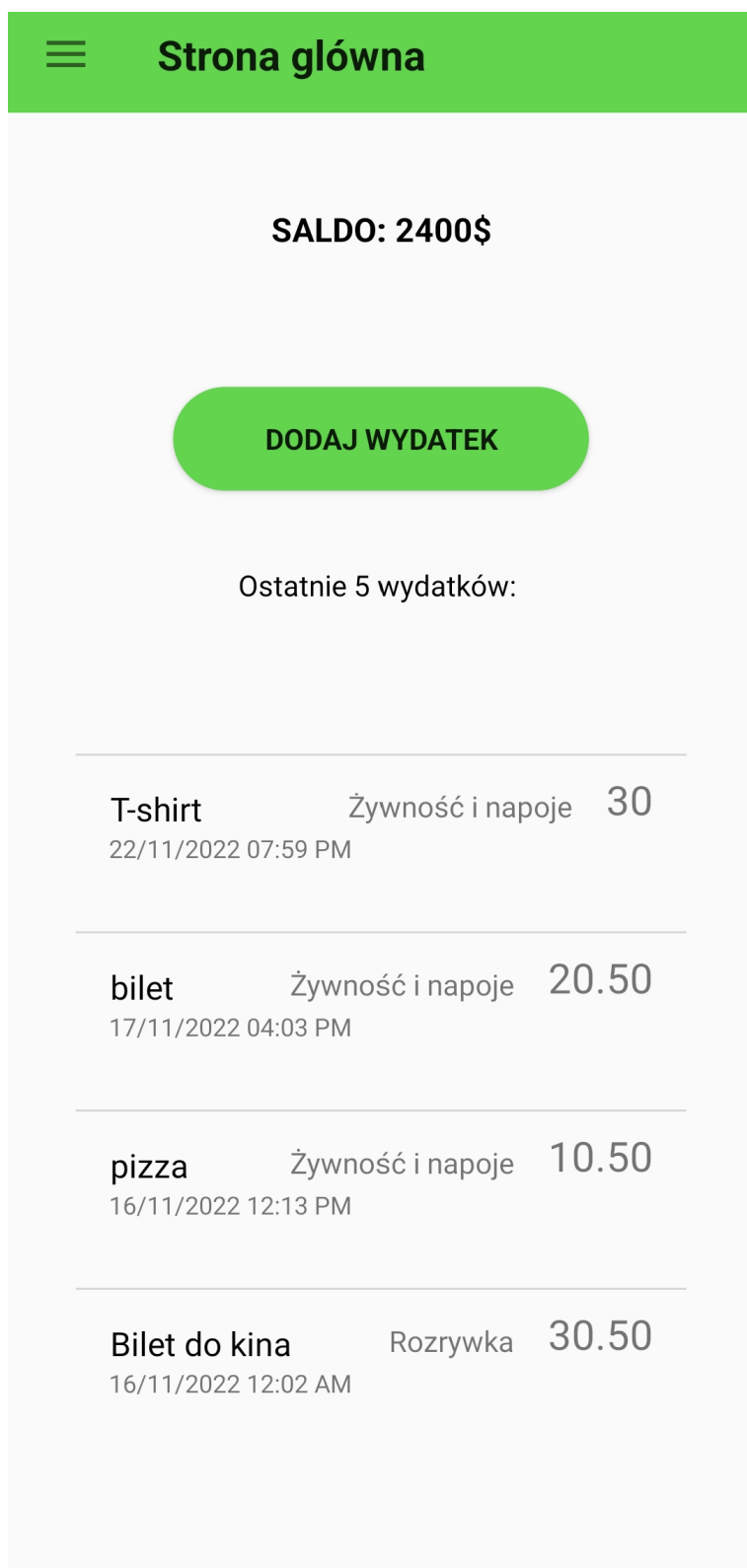
`androidx.recyclerview.widget.RecyclerView` - specjalna lista do wyświetlania wydatków z bazy danych.

`androidx.appcompat.widget.Toolbar` - górna belka.

`com.google.android.material.navigation.NavigationView` - nawigacja aplikacji.

`LinearLayout` - kolekcja która uporządkowuje elementy w cały layout.

Rysunek 3.1 (s. 11) przedstawia stronę główną aplikacji po zalogowaniu i potwierdzeniu danych biometrycznych. Górna część jest odpowiedzialna za nawigację po aplikacji, poniżej znajdują się saldo naszego konta, przycisk dodaj wydatek oraz



Rys. 3.1. Strona główna aplikacji

ostatnie 5 wydatków. Rysunek 3.2 (s. 13) przedstawia aktywność, gdzie pobrane są wszystkie wydatki użytkownika.

Wydatki			
T-shirt	Żywność i napoje	30	22/11/2022 07:59 PM
bilet	Żywność i napoje	20.50	17/11/2022 04:03 PM
pizza	Żywność i napoje	10.50	16/11/2022 12:13 PM
Bilet do kina	Rozrywka	30.50	16/11/2022 12:02 AM
Garnitur	Ubrania	500	15/11/2022 10:03 PM
Kebab	Zywnosc	28	15/11/2022 09:31 PM
Bilet autobusowy	Transport	5.50	15/11/2022 08:44 PM

Rys. 3.2. Strona podglądu wszystkich wydatków

## 4. Implementacja

### 4.1. Firebase Auth

RegisterPage:

```
1 regBtn.setOnClickListener(new View.OnClickListener() {
2     @Override
3     public void onClick(View view) {
4         String email = regMail.getText().toString().trim();
5         String password = regPassword.getText().toString().
6         trim();
7         String login = regLogin.getText().toString();
8         if(TextUtils.isEmpty(email)){
9             regMail.setError("Email is Required");
10            return;
11        }
12        if (TextUtils.isEmpty(password)) {
13            regPassword.setError("Password is Required");
14            return;
15        }
16
17        if (password.length() < 6) {
18            regPassword.setError("Password must be >=6
19            characters");
20            return;
21        }
```

**Listing 1.** Firebase Auth naciśnięcie przycisku rejestracji

Listing 1 (s. 14) to metoda, po naciśnięciu przycisku rejestracji, tworzy zmienne email, login i hasło, a następnie sprawdza czy pole email nie jest puste, czy pole hasło nie jest puste oraz czy hasło zawiera przynajmniej 6 znaków. Linijka 1 ustawia przycisk w tryb nasłuchiwanie czy przycisk jest kliknięty, linijka 4 i 5 przypisuje text wpisany w formularzu do odpowiadających im stringom, linijka 7 sprawdza czy został wprowadzony jakiś tekst - tak samo w linii 12, a w linii 17 sprawdzamy długość hasła.

Metoda Firebase Auth w listingu (2 (s. 15)) to metoda tworząca użytkownika za pomocą emaila i hasła. Funkcja onComplete w linii 3 sprawdza czy rejestracja się powiodła. Jeśli tak, to w linii 6 z Firebase Auth pobrane zostaje ID użytkownika i zapisane jako zmienna String. Następnie w linii 8 zostaje utworzona Hashmapa do której zapisane zostają login oraz email pobrany z formularza rejestracji. Hashmapa

w linii 11 zostaje dodana do bazy danych do kolekcji "users" w dokumencie o id poszczególnego użytkownika. Następnie przenosi do strony głównej aplikacji.

```

1  fAuth.createUserWithEmailAndPassword(email, password).
   addOnCompleteListener(new OnCompleteListener<AuthResult>() {
2      @Override
3      public void onComplete(@NonNull Task<AuthResult> task) {
4          if (task.isSuccessful()) {
5              Toast.makeText(registerPage.this, "Utworzono użytkownika.",
               Toast.LENGTH_SHORT).show();
6              userID = fAuth.getCurrentUser().getUid();
7              DocumentReference documentReference = fStore.collection("users"
               ).document(userID);
8              Map<String, Object> user = new HashMap<>();
9              user.put("login", login);
10             user.put("email", email);
11             documentReference.set(user).addOnSuccessListener(new
               OnSuccessListener<Void>() {
12                 @Override
13                 public void onSuccess(Void unused) {
14                     Log.d(TAG, "onSuccess: Konto zostało utworzone dla " + userID);
15                     startActivity(new Intent(getApplicationContext(), startPage.
               class));
16                 } else {\\
17                     Toast.makeText(registerPage.this, "Bład! " + task.getException
               ().getMessage(), Toast.LENGTH_SHORT).show();
18                 } } });\\
19

```

**Listing 2.** Firebase Auth - Rejestracja

```

1  if (fAuth.getCurrentUser() != null) {
2      startActivity(new Intent(getApplicationContext(),
               startPage.class));
3      finish();
4  }
5

```

**Listing 3.** Sprawdza zalogowanie

Instrukcja warunkowa, opisana 3 (s. 15) w linii 1 sprawdza czy jakiś użytkownik nie jest już zalogowany. Jeśli użytkownik jest zalogowany to w linii 2 przenosi go do strony głównej.

```

1 RegisterBtn.setOnClickListener(new View.OnClickListener() {
2     @Override
3     public void onClick(View view) {
4         startActivity(new Intent(loginPage.this,
5             registerPage.class));
6     }
7 });

```

Listing 4. Listing przycisku register

Listing 4 (s. 16) nasłuchuje, czy zostanie naciśnięty przycisk rejestracji, a jeśli zostanie przenosi użytkownika na stronę z formularzem rejestracji, dzięki linii 4.

```

1 LoginBtn.setOnClickListener(new View.OnClickListener() {
2     @Override
3     public void onClick(View view) {
4
5         String email = Login.getText().toString().trim();
6         String password = Password.getText().toString().trim();
7
8         if(email.isEmpty() || password.isEmpty()){
9             Toast.makeText(loginPage.this, "Wprowadz dane logowania
10             !", Toast.LENGTH_SHORT).show();
11             startActivity(new Intent(loginPage.this, loginPage.
12                 class));
13         }
14
15         fAuth.signInWithEmailAndPassword(email,password).
16         addOnCompleteListener(new OnCompleteListener<AuthResult>() {
17             @Override
18             public void onComplete(@NonNull Task<AuthResult> task)
19             {
20                 if(task.isSuccessful()){
21                     Toast.makeText(loginPage.this, "Zalogowano",
22                     Toast.LENGTH_SHORT).show();
23                     startActivity(new Intent(getApplicationContext()
24                     (),fpPage.class));
25                 }else {
26                     Toast.makeText(loginPage.this, "Bład logowania!
27                     " + task.getException().getMessage(), Toast.LENGTH_SHORT).show
28                     ();
29                 }
30             }
31         });
32     }
33 });

```



```

25     }
26 });

```

**Listing 5.** System logowania

Listing 5 (s. 16) po naciśnięciu przycisku logowania metoda w linii 5 i 6 zapisuje ciągi znaków email i hasło, wpisanych do odpowiednich pól i sprawdza czy jakieś pole nie zostało puste - jeśli tak, wyświetla odpowiedni komunikat. Jeśli email oraz hasło zostały wpisane, następuje sprawdzenie w Firebase czy w usłudze istnieje użytkownik o takim emailu i hasle - jeśli tak, wyświetla komunikat o zalogowaniu oraz przenosi użytkownika do strony z weryfikacją biometryczną. Jeśli nie znajdzie użytkownika o takich danych, wyświetli komunikat o błędzie logowania.

## 4.2. Baza danych

Rysunek 4.1 (s. 18) przedstawia wygląd bazy danych, pierwsza kolumna jest to kolekcja "users", następnie przechodzimy z niej do dokumentu - które noszą nazwę ID użytkownika. Następnie mamy wybór kolekcji, tutaj umieściliśmy "wydatki", poniżej również widać dane użytkownika - jego email oraz login.

Rysunek 4.2 (s. 18) to ciąg dalszy bazy danych, zatem po kliknięciu w kolekcję "wydatki" ukazują nam się dokumenty, które również jak użytkownicy noszą nazwę swojego ID - po kliknięciu w poszczególny dokument otrzymujemy dane wydatku, takie jak: data, kategoria, kwota oraz nazwa.

## 4.3. Tryb ciemny

Listing 6 (s. 17) to ustawienia dotyczące trybu ciemnego, linia pierwsza będzie odpowiadać za kolor tekstu na naszych stronach a linia druga za kolor tła - w pierwszej trzeba utworzyć plik w folderze values, który będzie odpowiadał za kolory, plik do trybu jasne już został wygenerowany automatycznie, zatem tworzymy plik /values/night/colors.xml. Dopisujemy w pliku sekcję, która będzie odpowiadała za kolor:

```

1  <color name="textColor">#000000</color>
2  <color name="backgroundColor">#ffffff</color>
3

```

**Listing 6.** night/colors.xml

W listingu 7 (s. 17) w pliku xml danej aktywności do elementu odnosimy się do wyżej określonych kolorów .

```

1  android:background="@color/background_color"

```

<a href="#">🏠</a> > <a href="#">users</a> > wxGls6zp1YQvS... <a href="#">More in Google Cloud</a>		
<a href="#">📁</a> menadzerwydatkow-5b241	<a href="#">📁</a> users	<a href="#">📁</a> wxGls6zp1YQvS2Q1kD13JnIlzVA3
<a href="#">+ Start collection</a>	<a href="#">+ Add document</a>	<a href="#">+ Start collection</a>
<a href="#">users</a> >	0MQinBSI7AVW1jjzdZsapf6LQNg2 CZ02Eq25U2VdRM12MZK50yK46Fw1 H0nv8ELQvfSD1tDYbIXAezzuzNx1 Tt28Ri4Nq0WPvMagLiTmnjy4bZ23 nbzEjlorAahuHhKELVtb82AQEqo2 <b>wxGls6zp1YQvS2Q1kD13JnIlzVA3</b> > zMmALayoQzSFBVQLnVKGovzaj0q2	wydatki
		<a href="#">+ Add field</a>
		email: "tester1@gmail.com" login: "tester1"

Rys. 4.1. Wygląd bazy danych.

<a href="#">🏠</a> > <a href="#">users</a> > <a href="#">wxGls6zp1YQvS...</a> > <a href="#">wydatki</a> > 10PI5MM1AvKC... <a href="#">More in Google Cloud</a>		
<a href="#">📁</a> wxGls6zp1YQvS2Q1kD13JnIlzVA3	<a href="#">📁</a> wydatki	<a href="#">📁</a> 10PI5MM1AvKC31dSNgf5
<a href="#">+ Start collection</a>	<a href="#">+ Add document</a>	<a href="#">+ Start collection</a>
<a href="#">wydatki</a> >	<b>10PI5MM1AvKC31dSNgf5</b> >	<a href="#">+ Add field</a>
<a href="#">+ Add field</a>	22WJZePcnBasVw3tTTg0 2Mk9kJof1cbWZS01zsCx WYAEJrPdt4eEQ0IpBn05 fB0aRj9Izse6cK8L8NUA hr5V2gDX17gjo3wwG398 l5UEBhQR02U8SashyX4D pcXTt8D1Gmbn7dxLS5eD qkgv7bY801BfstU18MbG s9WgzAgYo1GfkmEGSS6f	Data: November 15, 2022 at 10:03:39 PM UTC+1 Kategoria: "Ubrania" Kwota: "500" Nazwa: "Garnitur"
email: "tester1@gmail.com" login: "tester1"		

Rys. 4.2. Wygląd bazy danych cd.

**Listing 7.** Xml DarkMode

Listing 8 (s. 19) pobiera przycisk w linii 1 oraz obsługuje go, w linii 3 do zmiennej `isDarkModeOn` pobiera status przez funkcję `getDarkModeStatus()`, linia od 4 do 8 to instrukcja warunkowa, jeżeli DarkMode jest aktywny to zmieniamy nazwę przycisku oraz odwrotnie w zależności od trybu. Funkcja `onClick` w linii 12 zawiera ustawienie tego trybu dzięki "AppCompatActivity.MODE\_NIGHT\_YES/NO" oraz przypisuje `true` lub `false` do pomocniczej zmiennej `isDarkModeOn`.

```

1 btnlightdark = findViewById(R.id.btndarklight);
2
3 isDarkModeOn = getDarkModeStatus();
4 if(isDarkModeOn){
5     btnlightdark.setText("On Light Mode");
6 }else {
7     btnlightdark.setText("On Dark Mode");
8 }
9
10 btnlightdark.setOnClickListener(new View.OnClickListener() {
11     @Override
12     public void onClick(View view) {
13         if(isDarkModeOn){
14             AppCompatActivity.setDefaultNightMode(AppCompatActivity
15                 .MODE_NIGHT_NO);
16             btnlightdark.setText("On Dark Mode");
17             isDarkModeOn=false;
18         }else {
19             AppCompatActivity.setDefaultNightMode(AppCompatActivity
20                 .MODE_NIGHT_YES);
21             btnlightdark.setText("On Light Mode");
22             isDarkModeOn=true;
23         }
24     }
25 });

```

**Listing 8.** DarkMode Java Button obsługa

Listing 9 (s. 19) odpowiada za sprawdzanie trybu aplikacji w której jesteśmy aktualnie dzięki linii 2 gdzie pobieramy konfigurację flagi, oraz zwracamy w liniach 6, 8 i 10 `true` lub `false` co odpowiada trybom.

```

1 private boolean getDarkModeStatus(){
2     int nightModeFlags =
3         ustawieniaPage.this.getResources().getConfiguration().
4         uiMode & Configuration.UI_MODE_NIGHT_MASK;
5
6     if(nightModeFlags == Configuration.UI_MODE_NIGHT_YES){
7         return true;
8     }
9     if(nightModeFlags == Configuration.UI_MODE_NIGHT_NO){
10        return false;
11    }
12    return false;
13 }

```

```

4  switch (nightModeFlags){
5      case Configuration.UI_MODE_NIGHT_YES:
6          return true;
7      case Configuration.UI_MODE_NIGHT_NO:
8          return false;
9      case Configuration.UI_MODE_NIGHT_UNDEFINED:
10         return false;
11 }
12 return false;
13 }

```

**Listing 9.** DarkMode sprawdzanie motywu

## 4.4. Nawigacja

Tworzenie menu nawigacji w listingu 10 (s. 20) – Najpierw w layoucie strony trzeba zaimplementować widget, który umożliwi wysuwanie się szuflad z krawędzi okna:

```

1  <androidx.drawerlayout.widget.DrawerLayout>

```

**Listing 10.** Tworzenie menu nawigacji

Następnie w listingu 11 (s. 20) dodajemy bibliotekę NavigationView aby można było zaimplementować zawartość wszystkich typów szuflad nawigacyjnych:

```

1  <com.google.android.material.navigation.NavigationView>

```

**Listing 11.** Tworzenie NavigationView

Listing 12 (s. 20) odpowiada za wygląd naszego menu wysuwanego, do itemów przypisane są ikony, tytuł oraz id.

```

1  <menu xmlns:android="http://schemas.android.com/apk/res/android">
2
3      <group android:checkableBehavior="single">
4          <item
5              android:id="@+id/nav_start"
6              android:icon="@drawable/ic_start"
7              android:title="Strona główna" />
8          <item
9              android:id="@+id/nav_wydatki"
10             android:icon="@drawable/ic_wydatki"
11             android:title="Wydatki" />
12             <item
13                 android:id="@+id/nav_statystyki"
14                 android:icon="@drawable/ic_statystyki"
15                 android:title="Statystyki" />

```

```

16     <item
17         android:id="@+id/nav_ustawienia"
18         android:icon="@drawable/ic_ustawienia"
19         android:title="Ustawienia" />
20     </group>
21
22 </menu>

```

**Listing 12.** menu/main menu.xml

W listingu 13 (s. 21) obsługujemy wysuwanie i chowanie się całej nawigacji.

```

1  navigationView.bringToFront();
2  ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this,
3      drawerLayout, toolbar, R.string.navigation_drawer_open, R.string
4      .navigation_drawer_close);
5  drawerLayout.addDrawerListener(toggle);
6  toggle.syncState();

```

**Listing 13.** Wysuwanie się nawigacji

Listing 14 (s. 21) odpowiada za wywoływanie wybranej pozycji w menu nawigacji:

```

1  navigationView.setNavigationItemSelectedListener(this);

```

**Listing 14.** Wywoływanie pozycji nawigacja

Listing 15 (s. 21) tworzy metodę odpowiedzialną za poruszanie się między stronami. W linii 2 `onNavigationItemSelectedListener` odpowiada za to, że metoda zostaje wywołana po wybraniu elementu z menu nawigacji. `Intent` w linii np. 7 służy do przechodzenia między aktywnościami, `startActivity` w linii np. 8 uruchamia daną aktywność.

(Przykład na podstawie poruszania się ze strony startowej)

```

1  Override
2  public boolean onNavigationItemSelectedListener(@NonNull MenuItem menuItem)
3  {
4      switch (menuItem.getItemId()) {
5
6          case R.id.nav_wydatki:
7              Intent a = new Intent(startPage.this, wydatkiPage.class);
8              startActivity(a);
9              break;
10
11         case R.id.nav_statystyki:

```

```

12         Intent b = new Intent(startPage.this, statystykiPage.
13         class);
14         startActivity(b);
15         break;
16
17         case R.id.nav_ustawienia:
18             Intent c = new Intent(startPage.this, ustawieniaPage.
19             class);
20             startActivity(c);
21             break;
22
23         case R.id.nav_start:
24             Intent d = new Intent(startPage.this, startPage.class);
25             startActivity(d);
26             break;
27     }
28
29     drawerLayout.closeDrawer(GravityCompat.START);
30
31     return true;
32 }

```

Listing 15. Tworzenie poruszanie się między stronami

## 4.5. Dane biometryczne

Pierwszą najważniejszą rzeczą, którą należy wykonać do obsługi danych biometrycznych jest zaimplementowanie biblioteki androida, która obsługuje system weryfikacji biometrycznej. Implementacja następuje w pliku app/build.gradle do której dopisujemy następującą linijkę kodu zawartą w listingu 16 (s. 22):

```

1 implementation "androidx.biometric:biometric:1.1.0"

```

Listing 16. Implementacja biblioteki biometric

Następnie w naszym projekcie utworzyliśmy nową aktywność, która będzie odpowiadała za weryfikację biometryczną - nazwaliśmy ją fpPage. Na ekranie layoutu dodaliśmy przycisk odpowiadający za wywołanie weryfikacji - przycisk ikoniczny widniejący na obrazie 4.3 (s. 23).

Listing 17 (s. 23) jest to zawartość pliku fppage.java, który odpowiada za obsługę danych biometrycznych. Linijka 1 pobiera obrazek zaimplementowany wyżej do zmiennej imageViewLogin, następnie wywołujemy BiometricManager'a, oraz od linii 4 do 14 dzięki switchach i casach sprawdzamy czy operacje biometryczne powiodły się, efekt zwracamy w logach, lub w powiadomieniu w Toast'cie. W linii 23



**Rys. 4.3.** Ikoniczny przycisk biometriki

wywołujemy okienko autoryzacji, które opisane są w linii od 52 do 56 gdzie ustawiamy tytuł, podtytuł oraz przycisk, po udanej autoryzacji w linii 38 przenosimy się do naszej głównej strony aplikacji czyli `startPage.class`

```
1  imageViewLogin=findViewById(R.id.imageView);
2
3  BiometricManager biometricManager = BiometricManager.from(this);
4  switch (biometricManager.canAuthenticate(BIOMETRIC_STRONG |
5      DEVICE_CREDENTIAL)) {
6      case BiometricManager.BIOMETRIC_SUCCESS:
7          Log.d("MY_APP_TAG", "Aplikacja używa danych
8      biometrycznych.");
9          break;
10         case BiometricManager.BIOMETRIC_ERROR_NO_HARDWARE:
11             Toast.makeText(this, "Sensor niedostępny", Toast.LENGTH
12             LONG);
13             break;
14         case BiometricManager.BIOMETRIC_ERROR_HW_UNAVAILABLE:
15             Log.e("MY_APP_TAG", "Dane biometryczne niedostępne.");
16             break;
17         case BiometricManager.BIOMETRIC_ERROR_NONE_ENROLLED:
18             // Prompts the user to create credentials that your app
19             accepts.
20             final Intent enrollIntent = new Intent(Settings.
21             ACTION_BIOMETRIC_ENROLL);
22             enrollIntent.putExtra(Settings.
23             EXTRA_BIOMETRIC_AUTHENTICATORS_ALLOWED,
24                 BIOMETRIC_STRONG | DEVICE_CREDENTIAL);
25             startActivityForResult(enrollIntent, REQUEST_CODE);
26             break;
27     }
```

```
22 executor = ContextCompat.getMainExecutor(this);
23 biometricPrompt = new BiometricPrompt(fpPage.this,
24     executor, new BiometricPrompt.AuthenticationCallback() {
25     @Override
26     public void onAuthenticationError(int errorCode,
27         @NonNull CharSequence
28         errString) {
29         super.onAuthenticationError(errorCode, errString);
30         Toast.makeText(getApplicationContext(),
31             "Bład autoryzacji: " + errString, Toast.
32             LENGTH_SHORT)
33             .show();
34     }
35
36     @Override
37     public void onAuthenticationSucceeded(
38         @NonNull BiometricPrompt.AuthenticationResult result)
39     {
40         super.onAuthenticationSucceeded(result);
41         startActivity(new Intent(fpPage.this, startPage.class));
42         Toast.makeText(getApplicationContext(),
43             "Autoryzacja powiodła się!", Toast.LENGTH_SHORT).
44         show();
45     }
46
47     @Override
48     public void onAuthenticationFailed() {
49         super.onAuthenticationFailed();
50         Toast.makeText(getApplicationContext(), "Autoryzacja
51         niepowiodła się!",
52         Toast.LENGTH_SHORT)
53         .show();
54     }
55 });
56
57 promptInfo = new BiometricPrompt.PromptInfo.Builder()
58     .setTitle("Biometryczne odblokowywanie")
59     .setSubtitle("Odblokuj używając danych biometrycznych")
60     .setNegativeButtonText("Użyj hasła")
61     .build();
62
63 imageViewLogin.setOnClickListener(view -> {
64     biometricPrompt.authenticate(promptInfo);
65 })
```



```
60    });
```

**Listing 17.** Biometryka w pliku fppage.java

## 4.6. Text Spinner - wysuwane pole wyboru

Text Spinner czyli wysuwane pole wyboru, jest dosyć oczywistą rzeczą w aplikacji mobilnej, tym bardziej w menadżerze wydatków, akurat w nim zdecydowaliśmy się umieścić kategorię naszych wydatków na (Żywność i napoje/Odzież,etc.). Jest to dosyć istotna opcja w naszej aplikacji, ponieważ dzięki niemu nie będzie problemów z sortowaniem wydatków w bazie danych. Listing 18 (s. 25) jest to kod utworzenia właśnie tego "sztywnego" pola wyborów wydatków. Jak widać większość kodu przedstawia zwykłą personalizację elementu, aczkolwiek najważniejsze znajduje się w linii 8 w niej przypisujemy w nasze sztywne opcje wyboru.

```
1    <Spinner
2        android:id="@+id/kategoriaspinner"
3        android:layout_width="match_parent"
4        android:layout_height="wrap_content"
5        android:backgroundTint="#9C9C9C"
6        android:ems="10"
7        android:inputType="textPersonName" />
8        android:entries="@array/kategorie"
9        android:paddingTop="15dp"
10       android:paddingBottom="15dp" />
```

**Listing 18.** TextSpinner XAML

Elementy które znajdują się w spinnerze są określane właśnie w listingu 18 (s. 25) w pliku app/src/main/res/values/array.xml, oraz wywołujemy je w XAML'u dzięki nazwie string array'a, którą przypisujemy w linijce 2,a pozostałe w znacznikach "item" to nasze pola wyboru.

```
1    <Spinner
2        <string-array name="kategorie">
3        <item>Zywnosc i napoje</item>
4        <item>Kredyty</item>
5        <item>Mieszkanie i opłaty</item>
6        <item>Transport i komunikacja</item>
7        <item>Chemia domowa</item>
8        <item>Zdrowie i uroda</item>
9        <item>Hobby i rozrywka</item>
10       <item>Urlop i wyjazdy</item>
11       <item>Odziez</item>
```

```
12   <item>Inne</item>  
13 </string-array>
```

**Listing 19.** Array kategorie

## 5. Testowanie

Tab. 5.1. Tabela testowania

Element testowany	Wynik oczekiwany	Wynik ostateczny	Ocena
Logowanie	Kliknięcie przycisku zaloguj wczyta konto	Wynik otrzymany równy oczekiwanemu	Zalczony
Rejestracja	Przycisk rejestruj przenosi do formularza	Wynik otrzymany równy oczekiwanemu	Zalczony
Rejestracja formularz	Po wypełnieniu formularza oraz kliknięcie przycisku przenosi do strony głównej i rejestruje użytkownika	Wynik otrzymany równy oczekiwanemu	Zalczony
Walidacja danych	Aplikacja sprawdza złożoność danych oraz składnię i czy pola zostały wypełnione	Wynik otrzymany równy oczekiwanemu	Zalczony
Biometria	Odblokowanie aplikacji danymi biometrycznymi	Wynik otrzymany równy oczekiwanemu	Zalczony
Wyswietlanie danych	Na stronie głównej wyświetlanie salda oraz ostatnich wydatków z bazy danych	Wynik otrzymany równy oczekiwanemu	Zalczony
Aparat	W formularzu dodawania wydatków przywołuje aparat oraz zapisuje do podglądu	Wynik otrzymany równy oczekiwanemu	Zalczony
Zapisanie wydatku	Po wypełnieniu formularza przyciskiem "dodaj wydatek" zapisujemy wydatek w bazie danych	Wynik otrzymany równy oczekiwanemu	Zalczony
Wyswanie menu	Kliknięcie na "hamburger" wysuwa się menu aplikacji oraz przechodzi do aktywności	Wynik otrzymany równy oczekiwanemu	Zalczony
Szczegóły wydatków	Po kliknięciu na wydatek w "Wydatki" wyświetlają się dane wydatku	Wynik otrzymany równy oczekiwanemu	Zalczony
Usunięcie wydatku	Po wybraniu wydatku i kliknięciu "Usuń wydatek" wydatek zostaje usunięty	Wynik otrzymany równy oczekiwanemu	Zalczony
Tryb ciemny	Po kliknięciu na przycisk "Włącz tryb ciemny" w ustawieniach zmienia się tryb na ciemny	Wynik otrzymany równy oczekiwanemu	Zalczony
Limit wydatków	Po dodaniu limitu wydatków zostaje dodany do bazy	Wynik otrzymany równy oczekiwanemu	Zalczony
Miesięczne przychody	Po dodaniu miesięcznego przychodu zostaje on wyświetlony jako saldo i odjęta suma wydatków	Przychód nie resetuje się co miesiąc i nie odejmuje sumy	Częściowo zalczony
Wyloguj	Naciśnięcie przycisku wyloguj, wylogowuje i przenosi do strony logowania	Wynik otrzymany równy oczekiwanemu	Zalczony
Sprawdzanie zalogowania	Aplikacja zaraz po uruchomieniu sprawdza czy użytkownik jest zalogowany, jeśli jest nie ma ponownej konieczności logowania	Wynik otrzymany równy oczekiwanemu	Zalczony
Powiadomienia	Po dodaniu wydatku aplikacja sumuje wydatki i porównuje sume z limitem i wysła powiadomienie, gdy przekroczy próg	Wysła powiadomienie, ale nie sumuje wydatków	Częściowo zalczony
Statystyki	Zakładka "statystyki" wyświetla statystyki wydatków i kategorii z bazy oraz sumuje	Wyświetla diagram kołowy ale nie pobiera z bazy i nie sumuje	Częściowo zalczony
Pobieranie czasu	Aplikacja pobiera do wydatków automatycznie czas i jeśli trzeba do konwertuje	Wynik otrzymany równy oczekiwanemu	Zalczony
Chronologiczne wyświetlanie wydatków	Wydatki wyświetlane chronologicznie	Wynik otrzymany równy oczekiwanemu	Zalczony

## 6. Podręcznik użytkownika

Rysunek 6.1 (s. 30) jest to strona logowania, która ukazuje nam się zaraz po uruchomieniu aplikacji, pole 1 na rysunku odpowiada za wprowadzenie loginu użytkownika, pole 2 za hasło użytkownika, natomiast przycisk oznaczony cyfrą 3 odpowiada za zalogowanie się, a oznaczony numerem 4 przenosi do strony rejestracji użytkownika. Rysunek 6.2 (s. 31) jest to strona rejestracji, pole 1 to pole adresu e-mail, 2 to login użytkownika, w polu 3 wprowadzamy hasło oraz w 4 potwierdzamy hasło, przycisk oznaczony numerem 5 potwierdza naszą rejestrację w aplikacji.

Rysunek 6.3 (s. 32) jest to strona logowania biometrycznego, gdzie odblokowujemy aplikację za pomocą odcisku palcu lub tzw. "face id", po naciśnięciu na pole oznaczone cyfrą 1 uaktywniamy odblokowanie a następnie przenosi nas do strony głównej aplikacji.

Rysunek 6.4 (s. 33) jest to strona główna aplikacji. Pola oznaczone, odpowiadają za:

- 1 - Wysunięcie menu bocznego aplikacji.
- 2 - Saldo naszego konta.
- 3 - Przycisk przenoszący do formularza dodawania wydatków.
- 4 - Lista ostatnich 5 wydatków użytkownika.

Rysunek 6.5 (s. 34) jest to strona formularza dodawania wydatków. Pole:

- 1 - nazwa wydatku.
- 2 - kategoria wydatku.
- 3 - kwota wydatku.
- 4 - uruchomienie aparatu oraz dodanie zdjęcia wydatku.
- 5 - dodanie wydatku do bazy danych.

Rysunek 6.6 (s. 35) jest to wysuwane menu boczne aplikacji.

- 1 - Strona główna aplikacji
- 2 - Strona wyświetlająca wszystkie wydatki z bazy.
- 3 - Strona statystyk wydatków.
- 4 - Strona ustawień aplikacji oraz ustawień konta.

Rysunek 6.7 (s. 37) jest to strona przedstawiająca wszystkie wydatki użytkownika z bazy danych, po kliknięciu na poszczególny wydatek przenosi nas do szczegółów wydatku.



Rys. 6.1. Strona logowania

## REJESTRACJA



E-Mail **1**

---

Login **2**

---

Hasło **3**

---

Potwierdź hasło **4**

---

**ZAREJESTRUJ 5**

Rys. 6.2. Strona rejestracji

Odblokuj aplikacje!



**Rys. 6.3.** Strona logowania biometrycznego





Rys. 6.4. Strona główna aplikacji

☰ **Dodaj wydatek**

**1** Podaj nazwe

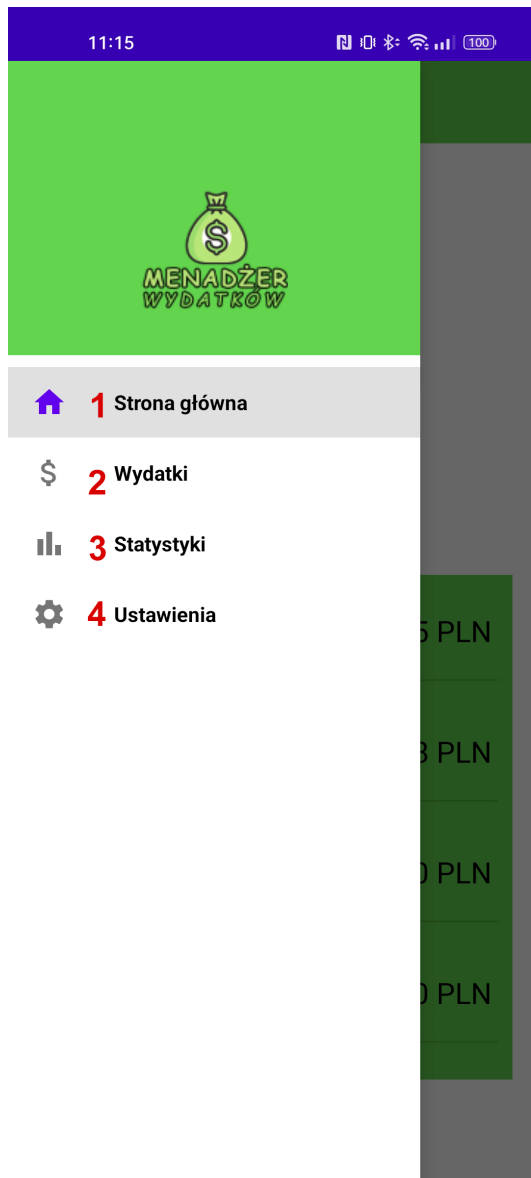
**2** Żywność i napoje ▼

**3** Podaj kwote:

**4** ZRÓB ZDJECIE PARAGONU/FAKTURY

**5** DODAJ WYDATEK

Rys. 6.5. Strona formularza dodawania wydatków



Rys. 6.6. Menu aplikacji

Rysunek 6.8 (s. 38) jest to strona przedstawiająca szczegóły danego wydatku.

- 1 - nazwa wydatku.
- 2 - kategoria wydatku.
- 3 - data dodania wydatku.
- 4 - kwota wydatku.
- 5 - zdjęcie wydatku.
- 6 - usuwanie wydatku.
- 7 - wróć do listy wydatków.

Rysunek 6.9 (s. 39) jest to strona przedstawiająca ustawienia aplikacji.

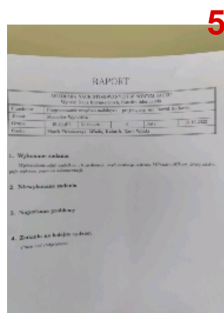
- 1 - zmiana trybu ciemnego/jasnego.
- 2 - wprowadzanie dziennego limitu wydatków.
- 3 - wprowadzanie miesięcznego przychodu.
- 4 - zapis limitu do bazy.
- 5 - zapis przychodu do bazy.
- 6 - wylogowanie użytkownika z aplikacji.

Rysunek 6.10 (s. 40) jest to strona przedstawiająca statystyki oraz diagram kołowy wydatków.

☰	<b>Wydatki</b>
1	
Test	35 PLN
Żywność i napoje	
pt. 30.12.2022   13:21	
Jhj	68 PLN
Żywność i napoje	
pt. 30.12.2022   13:20	
Dfff	500 PLN
Żywność i napoje	
pt. 30.12.2022   13:18	
Cfff	140 PLN
Żywność i napoje	
pt. 30.12.2022   13:11	
Ddd	2 PLN
Żywność i napoje	
pt. 30.12.2022   13:08	
Test limit	130 PLN
Żywność i napoje	
pt. 30.12.2022   13:07	
Test	12 PLN
Żywność i napoje	
czw. 22.12.2022   12:15	
Test	25 PLN
Żywność i napoje	
czw. 22.12.2022   11:13	

Rys. 6.7. Strona listy wydatków

Nazwa:	Garnitur <sup>1</sup>
Kategoria:	Żywność i napoje <sup>2</sup>
Data:	15/12/2022 04:04 PM <sup>3</sup>
Kwota:	500 PLN <sup>4</sup>



## USUŃ WYDATEK

WSTECZ

**Rys. 6.8.** Strona szczegółów wydatku

Ustawienia

Tryb ciemny/jasny:

1 WŁĄCZ TRYB CIEMNY

Dzienna maksymalna kwota wydatków

2

4 ZAPISZ

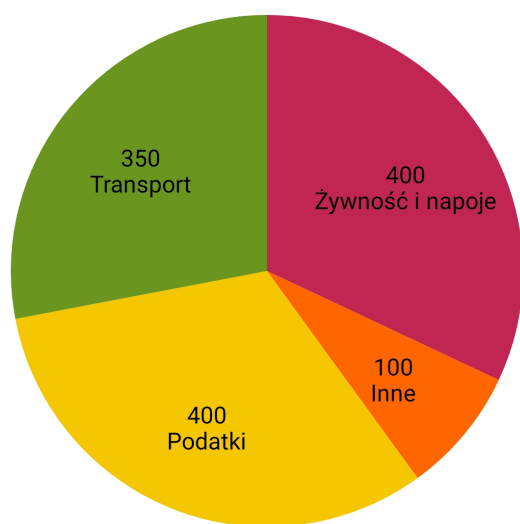
Miesięczne przychody

3

5 ZAPISZ

WYLOGUJ 6

Rys. 6.9. Strona ustawień aplikacji



Rys. 6.10. Strona statystyk



## Spis rysunków

1.1. Logo . . . . .	3
2.1. Schemat aplikacji . . . . .	5
3.1. Strona główna aplikacji . . . . .	11
3.2. Strona podglądu wszystkich wydatków . . . . .	13
4.1. Wygląd bazy danych. . . . .	18
4.2. Wygląd bazy danych cd. . . . .	18
4.3. Ikoniczny przycisk biometryki . . . . .	23
6.1. Strona logowania . . . . .	30
6.2. Strona rejestracji . . . . .	31
6.3. Strona logowania biometrycznego . . . . .	32
6.4. Strona główna aplikacji . . . . .	33
6.5. Strona formularza dodawania wydatków . . . . .	34
6.6. Menu aplikacji . . . . .	35
6.7. Strona listy wydatków . . . . .	37
6.8. Strona szczegółów wydatku . . . . .	38
6.9. Strona ustawień aplikacji . . . . .	39
6.10. Strona statystyk . . . . .	40

## Spis tabel

5.1. Tabela testowania . . . . .	28
----------------------------------	----

## Spis listingów

1.	Firebase Auth naciśnięcie przycisku rejestracji . . . . .	14
2.	Firebase Auth - Rejestracja . . . . .	15
3.	Sprawdza zalogowanie . . . . .	15
4.	Listing przycisku register . . . . .	16
5.	System logowania . . . . .	16
6.	night/colors.xml . . . . .	17
7.	Xml DarkMode . . . . .	17
8.	DarkMode Java Button obsługa . . . . .	19
9.	DarkMode sprawdzanie motywu . . . . .	19
10.	Tworzenie menu nawigacji . . . . .	20
11.	Tworzenie NavigationView . . . . .	20
12.	menu/main menu.xml . . . . .	20
13.	Wysuwanie się nawigacji . . . . .	21
14.	Wywoływanie pozycji nawigacja . . . . .	21
15.	Tworzenie poruszanie się między stronami . . . . .	21
16.	Implementacja biblioteki biometric . . . . .	22
17.	Biometryka w pliku fppage.java . . . . .	23
18.	TextSpinner XAML . . . . .	25
19.	Array kategorie . . . . .	25