AKADEMIA NAUK STOSOWANYCH
W NOWYM SĄCZU

Wydział Nauk Inżynieryjnych
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

1. Ogólne określenie wymagań

Aplikacja ma za zadanie podumowywać wydatki użytkownika. Klient będzie mógł wprowadzać dane dotyczące danego wydatku oraz podporządkować je katerogii (np. spożywcze/transport/podatki). Ma isnieć 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

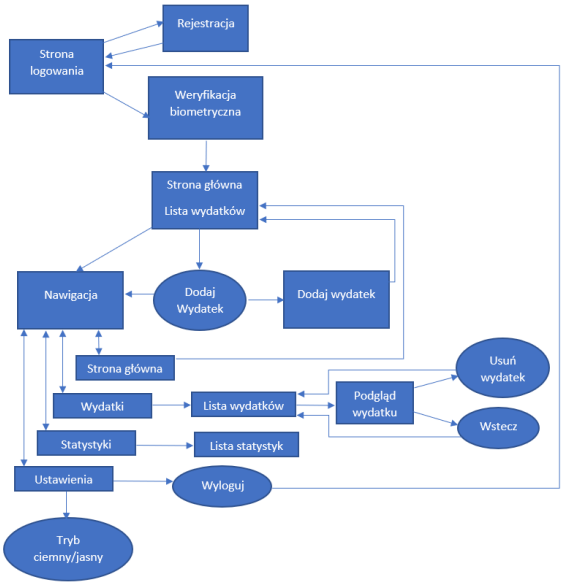
2. Określenie wymagań szczegółowych

Aplikacja będzie pisana w środowisku Android Studio jęzkykiem 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 włączeniiu 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 formalarza dodawania wydatku. Użytkownik będzie podawać tytuł wydatku, kwote 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 statystki. 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 użytkoników przy użyciu internetu.

Podkategoria statystki: suma wydatków, dochód, suma wydatków podporządkowana

kategoriom.
Podkategoria wydatki: wszystkie wydatki, możliwość usunięcia dowolnych wydatków.
Podkategoria ustawienia: przycisk switch dzięki któremy będziemy mieć możliwość przechodzenie pomiędzy trybem jasnym i ciemnym aplikacji, dane użytkownika oraz przycisk wyloguj.

2.1. Schemat



Rys. 2.1. Schemat aplikacji

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 przcyisk "Dodaj wydatek", dzięki niemu można przejść do aktywności 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 sie 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 przecisk "Wyloguj", który wylogowuje nas z aplikacji i przenosi do strony logowania.

3. Projektowanie

Aplikacja wymagała przygotowania narzędzi do rozpoczęcia programowania, rozpoczęliśmy od instalacji IDE Android Studio, który umożliwa wygodne tworzenie layoutów każdej aktywności oraz kodowanie w różnych językach programowania. Na potrzeby projektowe wybraliśmy język JAVA, również jednym z ważniejszych narzędzi które wykorzystywujemy 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 umożliwajaca połączenie z systemem FireBase Auth, odpowiada za cały system logowania oraz rejestrowania nowych użytkoników. Rozwijając myśl tej biblioteki, pobieramy z formularzy id zmiennych(e-mail, hasło) oraz inicjalizujemy bazę danych. Strona logowania (pierwsza po uruchomieniu aplikacji) wpierw sprawdza czy użytkownik jest zalogowany, jeżeli jest to przekierowuje go na stronę główna aplikacji, jeżli nie to zostaje na stronie do logowania, gdzie po wpisaniu danych sprawdza je z danymi serwera i loguje się gdy są poprawne dane poprzez przcisk "zaloguj", jeżli dane sa 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.

Kontunując, jeżeli logowanie przejdzie pomyślnie zanim uzyskamy dostęp do strony głównej aplikacja poprosi nas o zweryfikowanie właściciela poprzez dane biometryczne, dzięki bibilotece "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-

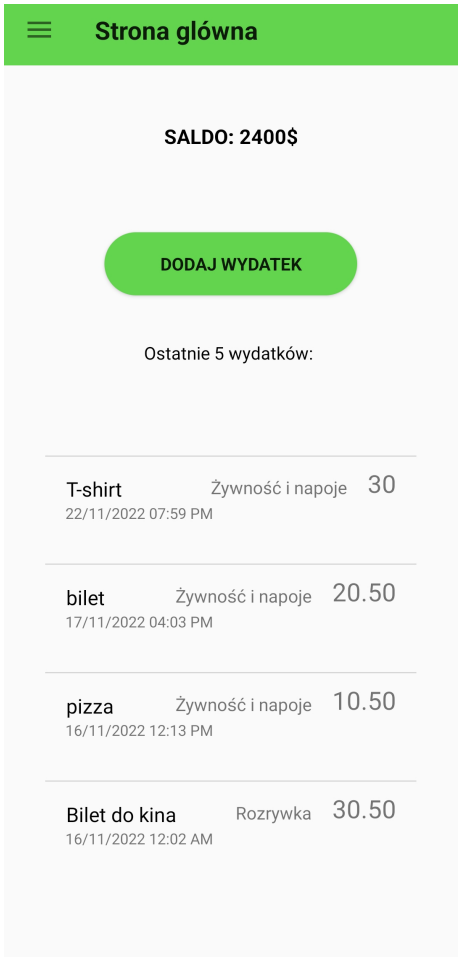
korzystujemy ją jako system do zapisywania wydatków w bazie danych dla poszczególnych użytkowników oraz do wyświetlania ich.
com.google.firebase.firestore.Query - biblioteka, którą wykorzystujemy do obsługi zapytań.
FirebaseUI ułatwia powiązanie danych z Cloud Firestore z interfejsem aplikacji.
com.google.firebase.firestore.DocumentReference - 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 wydaatku.

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 ustytyuwać 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.

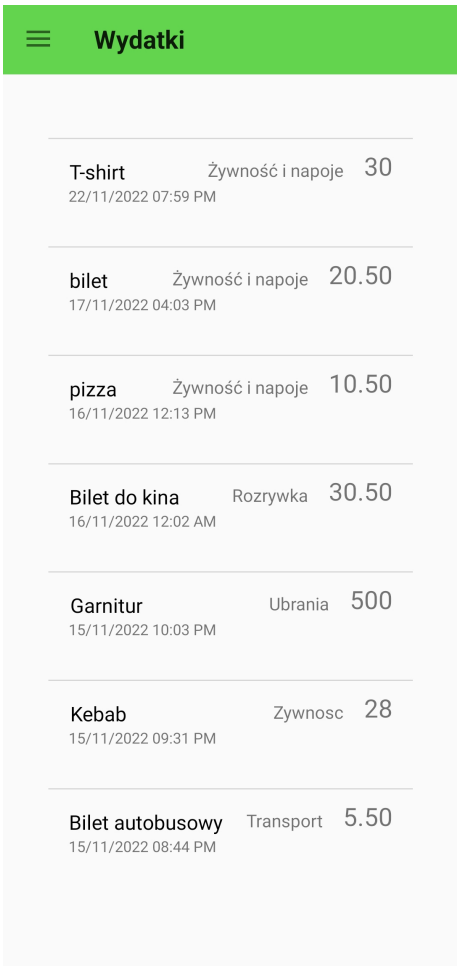
Elementy, które wykorzystujemy:
Button - wywołuje przycisk.
ImageView - tutaj wyświetlamy obraz.
EditText - tworzy pole do wpisania tekstu.
TextView - wyświetla tekst.
Spinner - lista rozwijana
androidx.recyclerview.widget.RecyclerView - specjalna lista do wyświetlania wydatków z bazy danych.
androidx.appcompat.widget.Toolbar - górna belka.
com.google.android.material.navigation.NavigationView - nawigacja aplikacji.
LinearLayout - kolekcja która uporządkowuje elementy w cały layout.

Rysunek 3.1 (s. 8) przedstawia stronę główną aplikacji po zalogowaniu i potwierdzeniu danych biometrycznych. Górna część jest odpowiedzialna za nawigację po aplikacji, poniżej znajduję sie saldo naszego konta, przycisk dodaj wydatek oraz



Rys. 3.1. Strona główna aplikacji

ostatnie 5 wydatków. Rysunek 3.2 (s. 10) przedstawia aktywność, gdzie pobrane są wszystkie wydatki użytkownika.



Rys. 3.2. Strona podglądu wszystkich wydatków

4. Implementacja

4.1. Firebase Auth

RegisterPage:

```
regBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String email = regMail.getText().toString().trim();
                String password = regPassword.getText().toString().
   trim();
                String login = regLogin.getText().toString();
                if(TextUtils.isEmpty(email)){
                    regMail.setError("Email id Required");
                    return;
                }

                if (TextUtils.isEmpty(password)) {
                    regPassword.setError("Password is Required");
                    return;
                }

                if (password.length() < 6) {
                    regPassword.setError("Password must be >=6
   characters");
                    return;
                }
```

Listing 1. Firebase Auth naciśnięcie przycisku rejestracji

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.

Metoda Firebase Auth tworząca użytkownika za pomocą emaila i hasła. Funckja onComplete sprawdza czy rejestarcja się powiodła. Jeśli tak, z Firebase Auth pobrane zostaje ID użytkownika i zapisane jako zmienna String. Następnie zostaje utworzona Hashmapa do której zapisane zostają login oraz email pobrany z formularza rejestarcji. Hashmapa zostaje dodana do bazy danych do kolekcji "users"w dokumencie o id poszczególnego użytkownika. Następnie przenosi do strony głównej aplikacji.

```
fAuth.createUserWithEmailAndPassword(email, password).
 addOnCompleteListener(new OnCompleteListener<AuthResult>() {
  @Override
   public void onComplete(@NonNull Task<AuthResult> task) {
   if (task.isSuccessful()) {
   Toast.makeText(registerPage.this, "Utworzono uzytkownika.",
 Toast.LENGTH_SHORT).show();
   userID = fAuth.getCurrentUser().getUid();
  DocumentReference documentReference = fStore.collection("users"
 ).document(userID);
    Map<String,Object> user = new HashMap<>();
  user.put("login",login);
   user.put("email",email);
   documentReference.set(user).addOnSuccessListener(new
 OnSuccessListener<Void>() {
   @Override
  public void onSuccess(Void unused) {
   Log.d(TAG,"onSuccess: Konto zostalo utworzone dla "+ userID);
 } });
  startActivity(new Intent(getApplicationContext(),startPage.
 class));
  }else {\\
  Toast.makeText(registerPage.this, "Blad! " + task.getException
 ().getMessage(), Toast.LENGTH_SHORT).show();
  } } });\\\\

```

Listing 2. Firebase Auth - Rejestracja

```
  if (fAuth.getCurrentUser() != null) {
          startActivity(new Intent(getApplicationContext(),
 startPage.class));
          finish();
      }

```

Listing 3. Sprawdza zalogowanie

Instrukcja warunkowa sprawdza czy jakiś użytkownik nie jest już zalogowany. Jeśli użytkownik jest zalogowany to przenosi go do strony głównej.

Login page:

```
RegisterBtn.setOnClickListener(new View.OnClickListener() {
              @Override
              public void onClick(View view) {
```

```
                    startActivity(new Intent(loginPage.this,
    registerPage.class));
                }
            });
```

Listing 4. Listner przycisku register

Metoda nasłuchuje, czy zostanie naciśnięty przycisk rejestracji, a jeśli zostanie przenosi użytkownika na stronę z formularzem rejestracji.

```
  LoginBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        String email = Login.getText().toString().trim();
        String password = Password.getText().toString().trim();

        if(email.isEmpty() || password.isEmpty()){
            Toast.makeText(loginPage.this, "Wprowadz dane logowania
    !", Toast.LENGTH_SHORT).show();
            startActivity(new Intent(loginPage.this, loginPage.
    class));
        }


        fAuth.signInWithEmailAndPassword(email,password).
    addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task)
    {
                if(task.isSuccessful()){
                    Toast.makeText(loginPage.this, "Zalogowano",
    Toast.LENGTH_SHORT).show();
                    startActivity(new Intent(getApplicationContext
    (),fpPage.class));
                }else {
                    Toast.makeText(loginPage.this, "Blad logowania!
     " + task.getException().getMessage(), Toast.LENGTH_SHORT).show
    ();
                }
            }
        });
    }
});
```

Listing 5. System logowania

Po naciśnięciu przycisku logowania metoda 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 haśle - 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. 15) 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", ponieżej również widać dane użytkownika - jego email oraz login.

Rysunek 4.2 (s. 15) to ciąg dalszy bazy danych, zatem po kliknięciu w kolekcję "wydatki"ukazują nam się dokumenty, które również jak użytkownicy nosza nazwę swojego ID - po kliknięciu w poszczególny dokument otrzymujemy dane wydatku, takie jak: data, kategoria, kwota oraz nazwa.

4.3. Tryb ciemny

Tworznie trybu ciemnego w aplikacji - wpier trzeba utworzyć plik w folderze values, który będzie odpowiadał za kolory, plik do trybu jasne już został wygenerowany automatyczne, 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

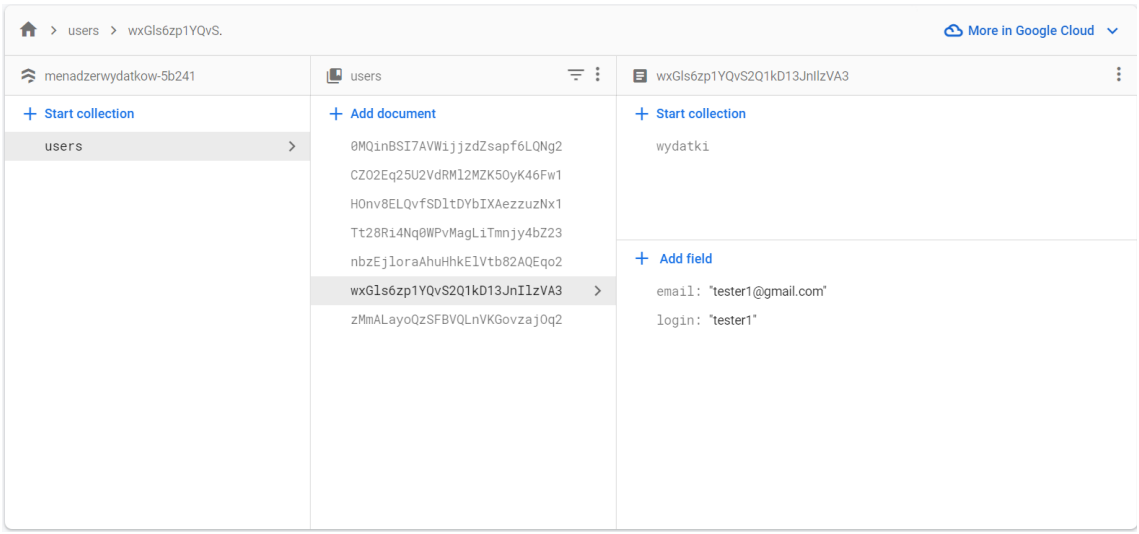
Następnie w pliku xml danej aktywności do elementu odnosimy sie do wyżej określonych kolorów .

```
1  android:background="@color/backgroundColor"
2
```

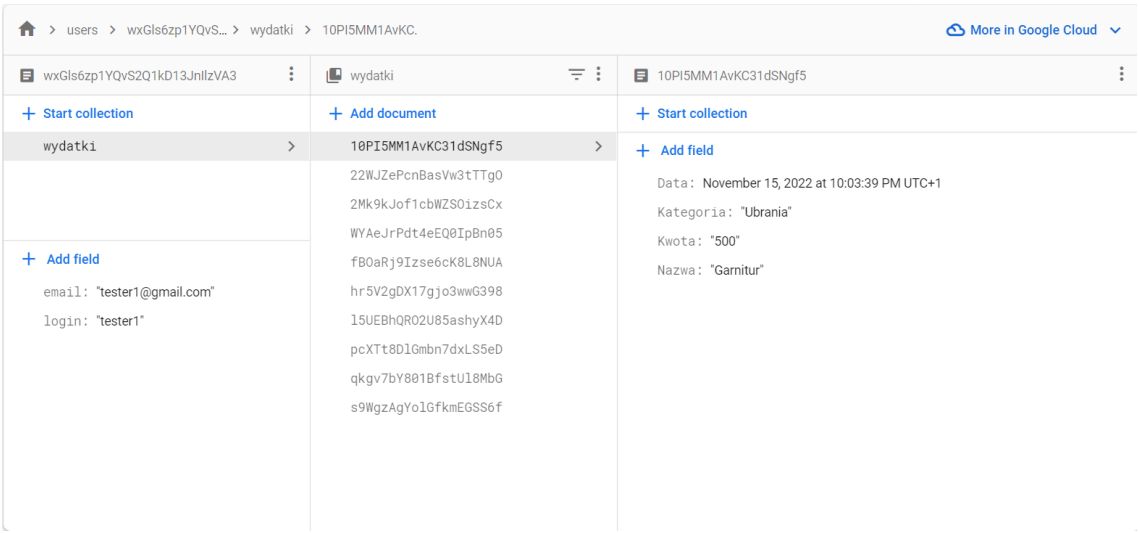
Listing 7. Xml DarkMode

Pobieramy przycisk oraz obsługujemy go.

```
1 btnlightdark = findViewById(R.id.btndarklight);
```



Rys. 4.1. Wygląd bazy danych.



Rys. 4.2. Wygląd bazy danych cd.

```
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             AppCompatDelegate.setDefaultNightMode(AppCompatDelegate
       .MODE_NIGHT_NO);
15             btnlightdark.setText("On Dark Mode");
16             isDarkModeOn=false;
17         }else {
18             AppCompatDelegate.setDefaultNightMode(AppCompatDelegate
       .MODE_NIGHT_YES);
19             btnlightdark.setText("On Light Mode");
20             isDarkModeOn=true;
21         }
22     }
23 });
```

Listing 8. DarkMode Java Button obsługa

Oraz sprawdzamy jaki jest statu trybu.

```
1  private boolean getDarkModeStatus(){
2    int nightModeFlags =
3            ustawieniaPage.this.getResources().getConfiguration().
      uiMode & Configuration.UI_MODE_NIGHT_MASK;
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 – Najpierw w layoucie strony trzeba zaimplementować widget, który umożliwia wysuwanie się szuflad z krawędzi okna:

```
<androidx.drawerlayout.widget.DrawerLayout>
```

Listing 10. Tworzenie menu nawigacji

Następnie trzeba dodać bibliotekę NavigationView aby można było zaimplementować zawartość wszystkich typów szuflad nawigacyjnych:

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

Listing 11. Tworzenie NavigationView

Tworzymy plik main menu.xml, który będzie odpowiadał za nasze menu:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">

  <group android:checkableBehavior="single">
<item
              android:id="@+id/nav_start"
                android:icon="@drawable/ic_start"
                android:title="Strona glowna" />
<item
          android:id="@+id/nav_wydatki"
              android:icon="@drawable/ic_wydatki"
              android:title="Wydatki" />
           <item
              android:id="@+id/nav_statystyki"
              android:icon="@drawable/ic_statystyki"
              android:title="Statystyki" />
    <item
             android:id="@+id/nav_ustawienia"
              android:icon="@drawable/ic_ustawienia"
              android:title="Ustawienia" />
  </group>

</menu>
```

Listing 12. menu/main menu.xml

Z kolei tworzymy:

wysuwanie się, oraz chowanie menu nawigacji:

```
  navigationView.bringToFront();
ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this,
   drawerLayout, toolbar, R.string.navigation_drawer_open, R.string
   .navigation_drawer_close);
```

```
drawerLayout.addDrawerListener(toggle);
toggle.syncState();
```

Listing 13. Wysuwanie się nawigacji

Wywoływanie wybranej pozycji w menu nawigacji:

```
  navigationView.setNavigationItemSelectedListener(this);
```

Listing 14. Wywoływanie pozycji nawigacja

Tworzymy metodę odpowiedzialną za poruszanie się między stronami. onNavigationItemSelected odpowiada za to, że metoda zostaję wywołana po wybraniu elementu z menu nawigacji:

(Przykład na podstawię poruszania się ze strony startowej)

```
  Override
public boolean onNavigationItemSelected(@NonNull MenuItem menuItem)
    {

    switch (menuItem.getItemId()) {

        case R.id.nav_wydatki:
            Intent a = new Intent(startPage.this, wydatkiPage.class
   );
            startActivity(a);
            break;

        case R.id.nav_statystyki:
            Intent b = new Intent(startPage.this, statystykiPage.
   class);
            startActivity(b);
            break;

        case R.id.nav_ustawienia:
            Intent c = new Intent(startPage.this, ustawieniaPage.
   class);
            startActivity(c);
            break;

        case R.id.nav_start:
            Intent d = new Intent(startPage.this, startPage.class);
            startActivity(d);
            break;
    }

```

```
27      drawerLayout.closeDrawer(GravityCompat.START);
28
29      return true;
30  }
```

Listing 15. Tworzenie poruszanie się między stronami

5. Testowanie

6. Podręcznik użytkownika

Spis rysunków

Spis tabel

Spis listingów