AKADEMIA NAUK STOSOWANYCH
W NOWYM SĄCZU

Wydział Nauk Inżynieryjnych
Katedra Informatyki

DOKUMENTACJA PROJEKTOWA

PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

Menadżer Wydatków

Autor:
Marek Pichniarczyk
Karol Wolski
Mikołaj Kwiatek

Prowadzący:
mgr inż. Dawid Kotlarski

Nowy Sącz 2022

Spis treści

1. Ogólne określenie wymagań

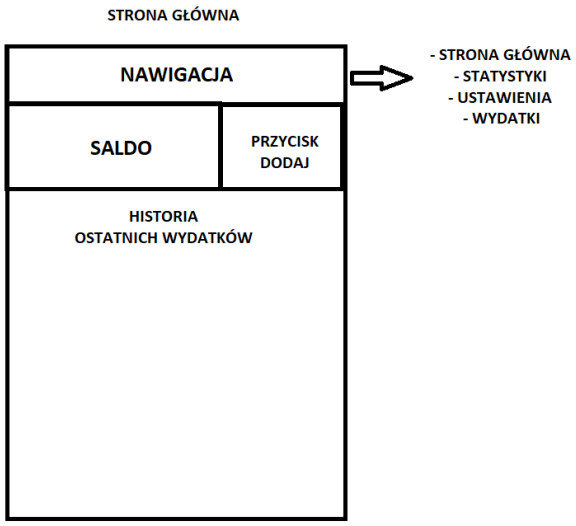
Aplikacja ma za zadanie podumowywać wydatki użytkownika. Klient będzie mógłwprowadzać dane dotyczące danego wydatku oraz podporządkować je katerogii (np. spożywcze/transport/podatki). Ma isnieć możliwość dodania także miesięcznych przychodów z których odejmowane będą wydatki. Do każdej pozycji będzie 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 odciska palca.

Rys. 1.1. Logo

2. Określenie wymagań szczegółowych

Aplikacja będzie pisana w środowisku Android Studio językykiem 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 strony głównej

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".

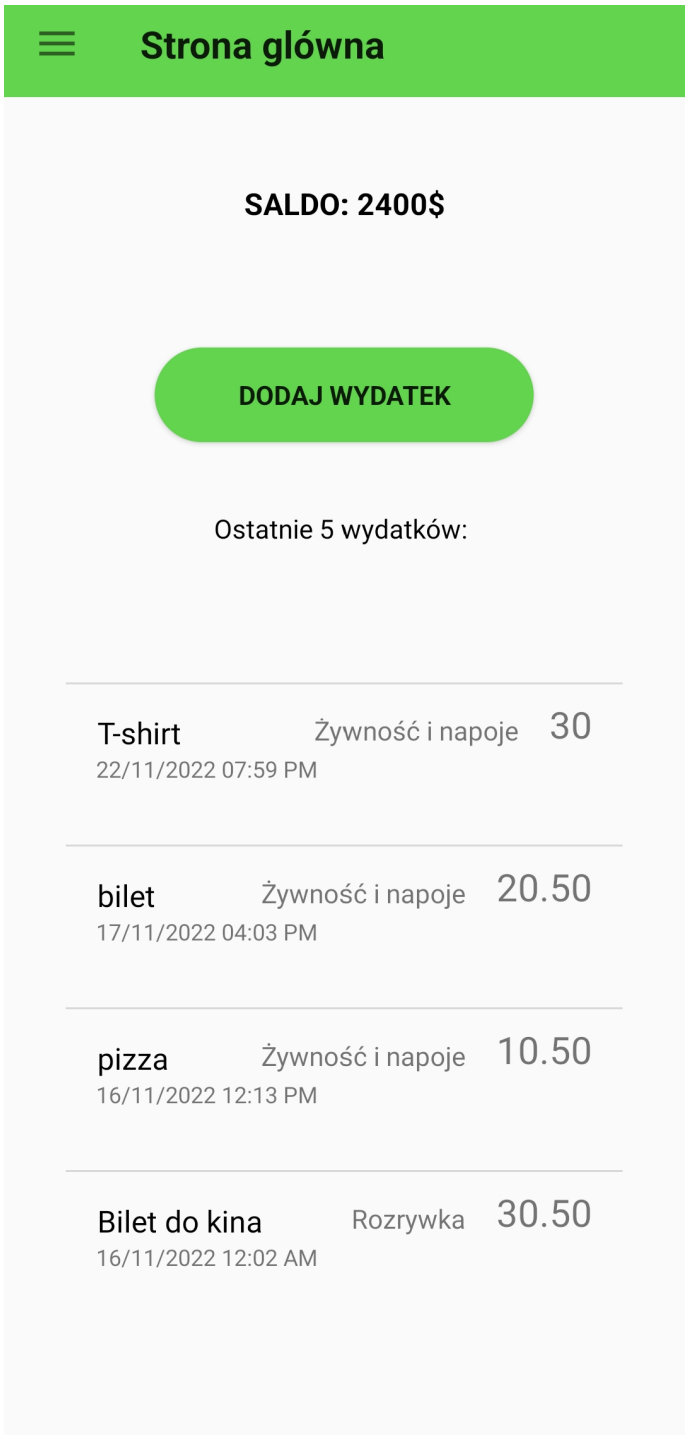
com.google.firebase.firestore.FirebaseFirestore - biblioteka umożliwiająca obsługę bazy danych znajdującej się w Firebase, odpowiada za system przechowywania, odczytu i zapisu danych poszczególnych użytkowników. Cloud Firestore jest elastyczną, skalowalną, hierarchiczną bazą danych NoSQL w chmurze służącą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

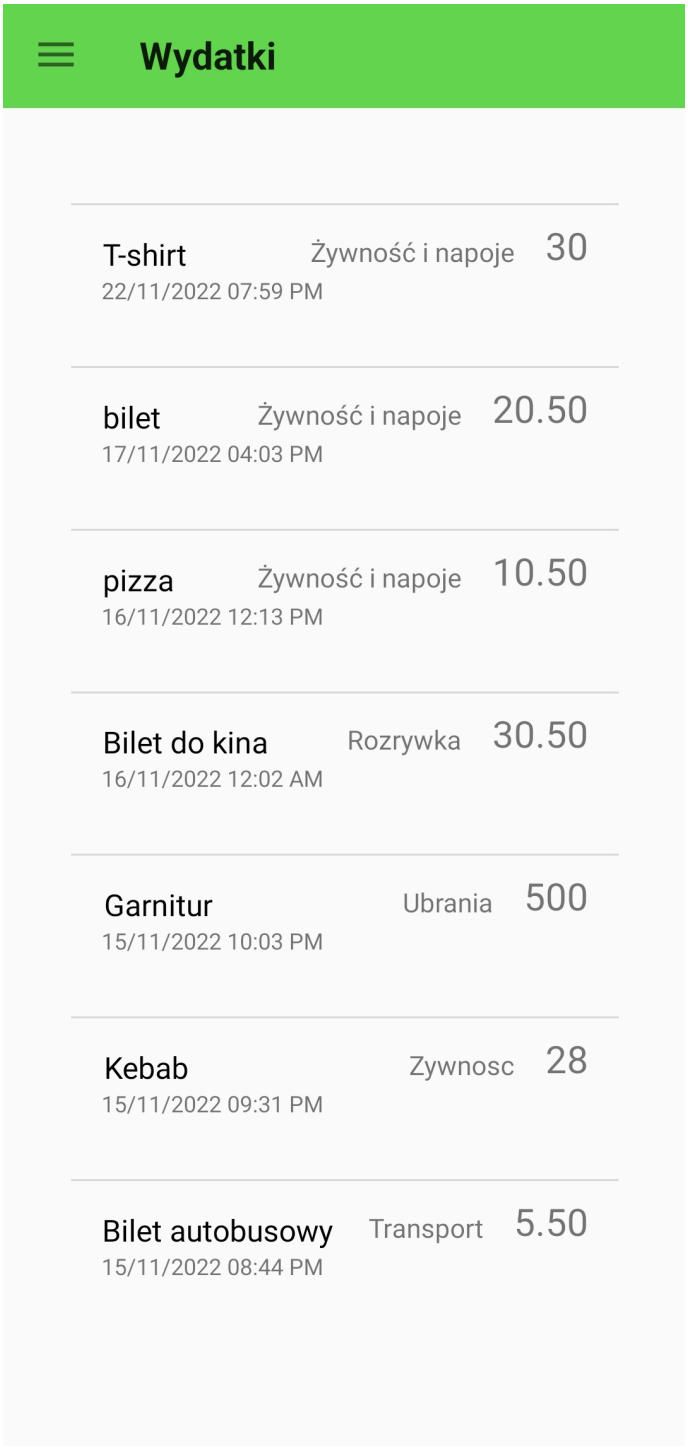
Frontend - XAML
Wygląd naszej aplikacji jest obsługiwany przez język XAML, dzięki niemu jesteśmy w stanie ustytyuwać różne elementy aplikacji w wybranym przez nas miejscu. Oczywiście każdy element możemy edytować w wybrany przez nas sposób, odpowiadają

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.



Rys. 3.1. Strona główna aplikacji



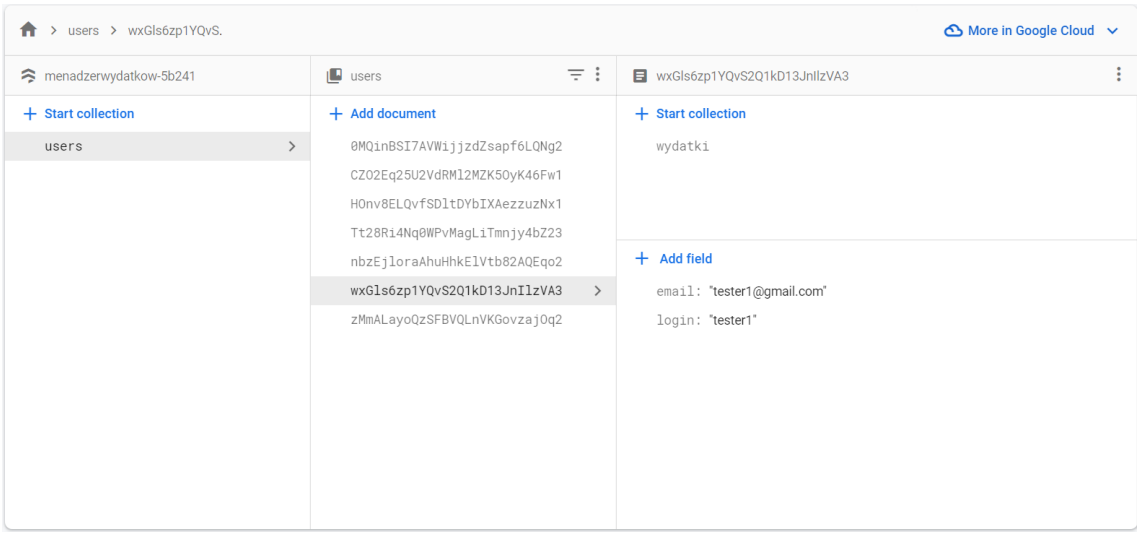
Rys. 3.2. Strona podglądu wszystkich wydatków

4. Implementacja

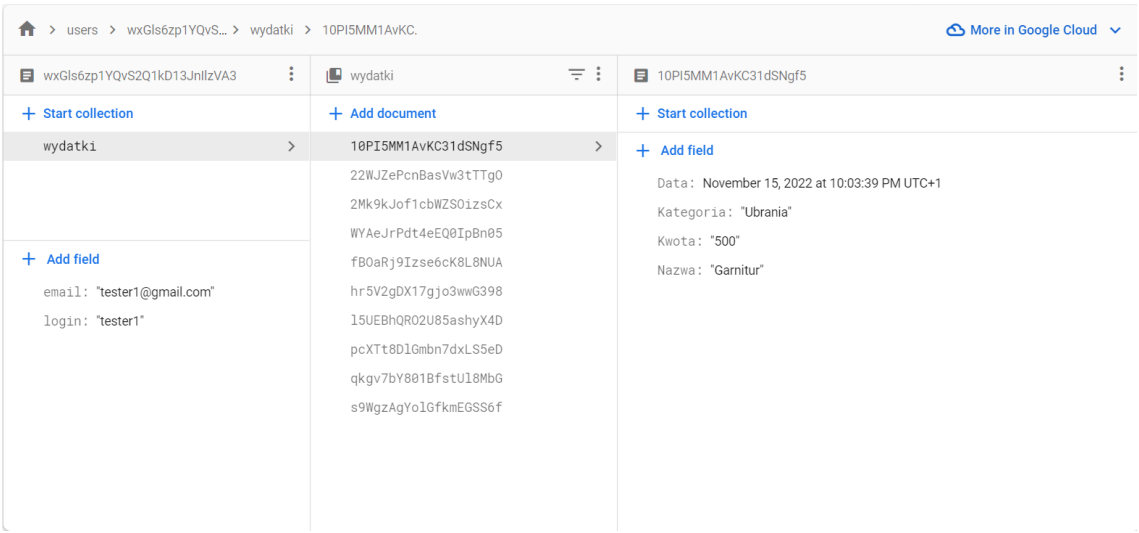
Metoda Firebase Auth tworząca użytkownika za pomocą emaila i hasła.
Funckja onComplete sprawdza czy rejestarcja się powiodła.
Jeśli tak, z Firebase Auth pobrane zostaje ID użytkownika i zapisane jako zmienna String.
Następnie zostaje utworzona Hashmapa do której zapisane zostają login oraz email pobrany 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\\ OnC
@Override
 public void onComplete(@NonNull Task<AuthResult> task) {
 if (task.isSuccessful()) {
 Toast.makeText(registerPage.this, "Utworzono użytkownika.", Toast.LENGTH SHORT)\\.sho
 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 zostało utworzone dla "+ userID); } });
startActivity(new Intent(getApplicationContext(),startPage.class));
}else {\\
Toast.makeText(registerPage.this, "Błąd! " + task.getException().getMessage(),\\ Toast
} } });\\\\
```

Wygląd bazy danych:



Rys. 4.1. Wygląd bazy danych.



Rys. 4.2. Wygląd bazy danych cd.

5. Testowanie

6. Podręcznik użytkownika

Spis rysunków

Spis tabel

Spis listingów