|  | **Wydział Finansów i Zarządzania Kierunek: Informatyka** |
| --- | --- |

**Jan Wieprow**

(numer albumu: 81597)

**Aplikacja mobilna pozwalająca na zapisanie informacji o nowo poznanych ludzi w celu przypomnienia sobie o nich przy ponownym spotkaniu**

Inżynierska praca projektowa

**Opiekun merytoryczny:**

**stopień naukowy, imię nazwisko opiekuna**

Wrocław 2025

1. Wstęp
   1. Wprowadzenie, cel, zakres pracy

W aktualnym świecie wraz z rozwojem technologii oraz trendami, życie ludzkie zaczyna płynąć coraz szybciej. Rzadko skupiamy się na obecnej chwili przez co coraz trudniej zapamiętać konkretne wydarzenia czy ludzi. Szczególnie trudne może dla niektórych dokładne zapamiętanie czyjegoś imienia czy miejsce poznania, przy jednoczesnej świadomości, że skądś kojarzy się tą osobę. Taka sytuacja często jest niezwykle frustrująca czy nawet zawstydzająca. W celu rozwiązania właśnie takich problemów powstało Humanon. Jej celem jest przełożenie potrzeby zapamiętania niezliczonej ilości osób właśnie na tą aplikację. Pozwoli ona na zapisanie informacji o nowo poznanych ludziach takich jak np. imię, miejsce spotkania, charakter, wygląd czy zdjęcie. Daje to możliwość dyskretnego sprawdzenia spotkanej osoby czy już wcześniej się ją poznało. Istnieje również biznesowe zastosowanie dla Humanon. Możliwe jest zapisywanie nowo poznanych partnerów biznesowych w celu późniejszego skontaktowania się lub po prostu żeby ich zapamiętać.

* 1. Analiza konkurencyjności
  2. Analiza SWOT

Analiza SWOT jest to narzędzie analizy strategicznej przedsiębiorstwa. Jest to najpopularniejsza heurystyczna technika analityczna służąca do porządkowania informacji. SWOT ma za zadanie pomóc osobie analizującej poukładać oraz uporządkować myślenie tak, aby informacje były przedstawione w sposób czytelny oraz w łatwy sposób można było wyciągać z nich wnioski.

Nazwa SWOT jest akronimem słów ,, Strenghts (mocne strony przedsiębiorstwa), Weakness (słabe strony przedsiębiorstwa), Opportunities (szanse w otoczeniu przedsiębiorstwa), Threats(zagrożenia w otoczeniu przedsiębiorstwa)”. Będą one szczegółowo opisane w podpunktach poniżej. Dzięki tej analizie jako przedsiębiorstwo można odpowiedzieć na pytania, jaka jest jego pozycja na rynku, jakie są jego cele czy za pomocą jakich narzędzi będą je realizować w przyszłości.[[1]](#footnote-0)

* 1. Strengths (Mocne strony)

Mocne strony przedsiębiorstwa to czynniki, które wyróżniają jednostkę od konkurencyjnych firm, które działają w podobnej branży czy obszarze.

W mojej aplikacji takimi czynnikami mogą być:

* Prostota użytkowania
* Możliwe łatwe dodanie nowych funkcjonalności
* Niski koszt produkcji
* Nacisk na ładny design
* Pomoc w zapamiętaniu nowo poznanych osób
  1. Weaknesses (słabe strony)

Słabe strony przedsiębiorstwa to takie czynniki, które wynikają z niewystarczających umiejętności i ograniczonych zasobów.

W mojej aplikacji takimi czynnikami mogą być:

* Mała ilość funkcjonalności
* Brak jasno wytyczonej strategii
* Brak badań rynku
* Brak marketingu
* Trudność w dotarciu do nowych klientów
  1. Opportunities (szanse)

Szanse to wszystkie czynniki zewnętrzne jeżeli dobrze wykorzystane to staną się bodźcem rozwoju przedsiębiorstwa.

W mojej aplikacji takimi czynnikami mogą być:

* Utrata zainteresowania konkurencyjnymi platformami
* Tendencje społeczne do krótkiego utrzymania uwagi
* Ciągły wzrost użycia telefonów nad komputerami
* Wzięcie pod uwagę ludzi, dla których taka aplikacja byłaby szczególnie użyteczna
  1. Threats

Zagrożenia przedsiębiorstwa to takie czynniki, które mogą pojawić się w otoczeniu organizacji i są postrzegane jak utrudnienia, niebezpieczeństwa oraz bariery w działalności organizacji.

W mojej aplikacji takimi czynnikami mogą być:

* Istnienie podobnych w założeniu aplikacji
* Brak siły przebicia
* Niewystarczająca atrakcyjność aplikacji
* Niewystarczająca ilość testów
* Niewystarczające zauważenie przez ilość wypuszczanych aplikacji

1. Założenia projektowe
   1. Wymagania funkcjonalne według zasady MoSCoW
   2. Wymagania niefunkcjonalne według zasady MoSCoW
   3. Diagramy UML
   4. ERD bazy danych
   5. UX - prototypowanie
2. Technologie opisać też do czego służyły w mojej aplikacji
   1. Android
   2. Java jest to obiektowy język programowania. Może być używany do tworzenia niezależnych aplikacji a także działających na serwerze WWW.[[2]](#footnote-1)
   3. Android Studio
   4. Firebase - napisać o bazie danych oraz logowaniu

Firebase jest platformą w posiadania google, kótra

* 1. GitHub

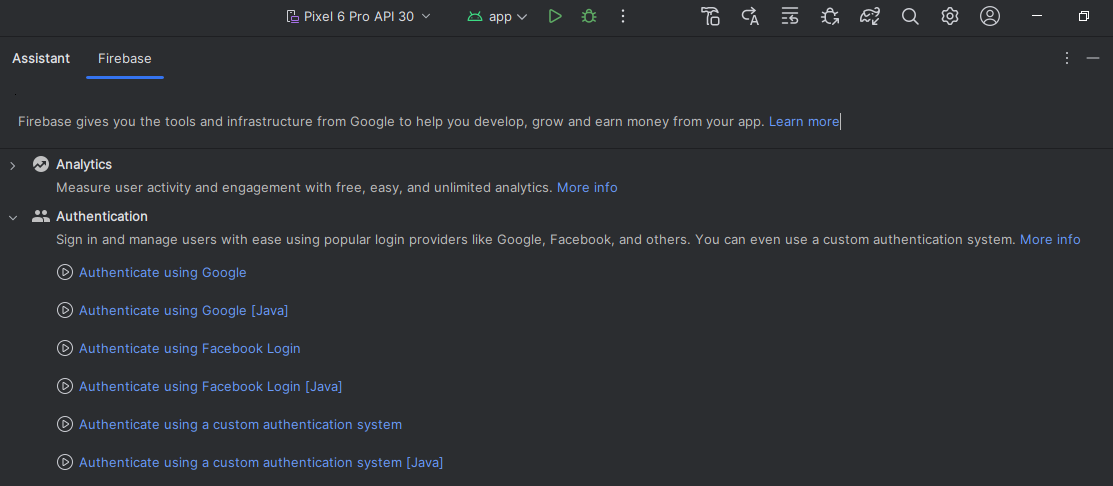
GitHub jest to platforma w chmurze, która używa open-source rozwiązania kontroli wersji Git. Pozwala ona na przechowywanie kodu dzięki czemu możliwe jest śledzić zmiany jakie nastąpiły w projekcie, wydzielać zadania, recenzować, udostępniać czy pracować wspólnie nad kodem. Jest to niezwykle przydatna strona internetowa, która niezwykle często przez programistów. Właśnie przy jej użyciu pisana jest aplikacja opisywana w tej pracy inżynierskiej.[[3]](#footnote-2)

* 1. Git

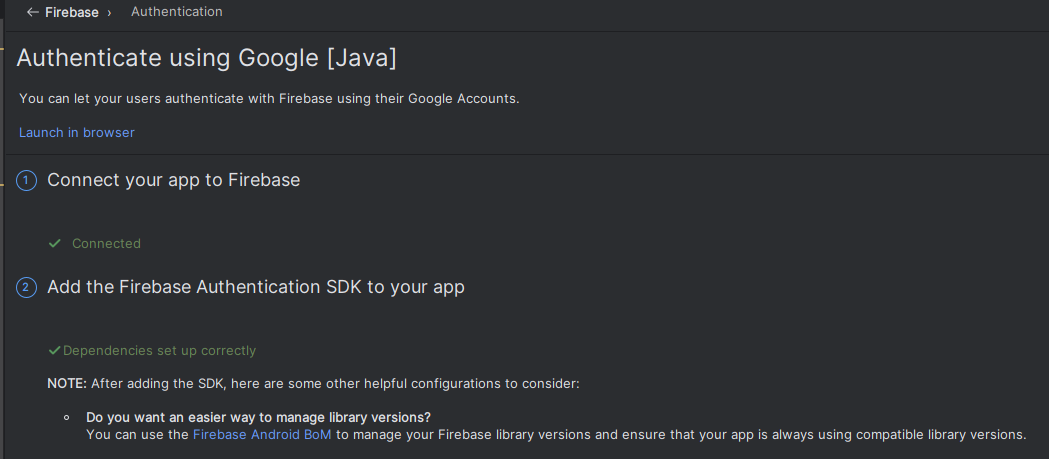
Git jest to rozwiązanie open-source pozwalające na śledzenie zmian w plikach czyli kontrole wersji albo VCS. Jest to niezwykle użyteczna technologia przy pracy wielu osób nad jednym plikiem lub plikami. Dzięki temu deweloper może pracować na dowolnej wersji projektu oraz odzyskać poprzednią wersje plików. Może on również przeglądać kod w celu sprawdzeniu kiedy i jakie zmiany zostały dokonane, kto ich dokonał oraz przy dobrych praktykach dlaczego zostały one dokonane. [[4]](#footnote-3)

1. Implementacja i wdrożenie
   1. Opis projektu
   2. Ikona aplikacji
   3. Logowanie do aplikacji

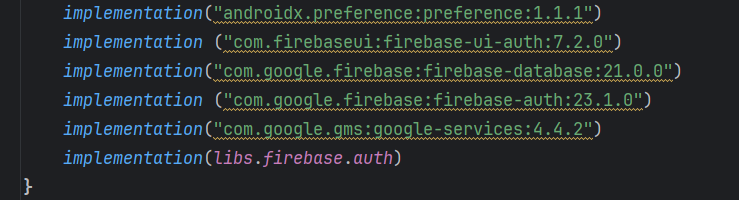
Żeby używać FireBase w aplikacji należy najpierw połączyć Android Studio ze stroną Fire Base na której znajduje się nasz projekt.



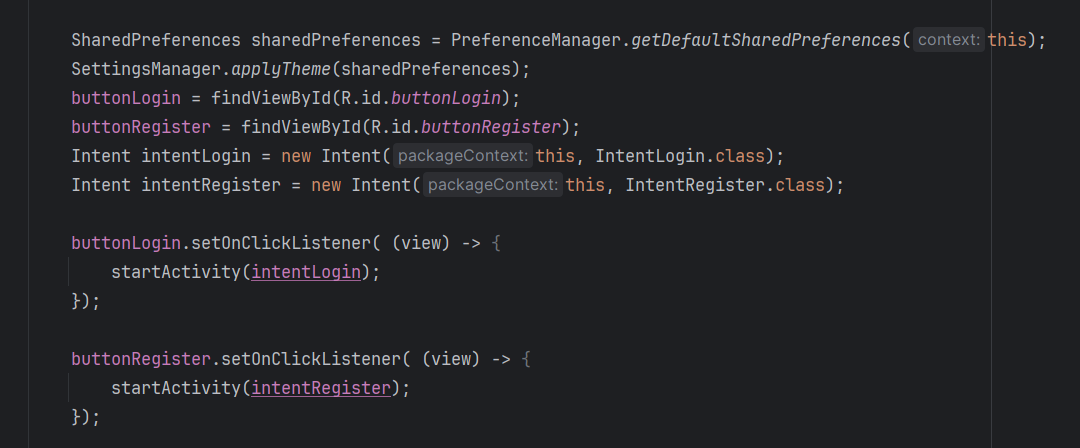
Na załączonym obrazku przedstawione jest okno FireBase w Android Studio. W celu w połączenia projektu ze stroną Fire Base i umożliwić logowanie przez Google należy wybrać Authenticate using Google [Java].



Na obrazku widać, że projekt został już połączony z Fire Base. Przed połączeniem, pod Connect your app to Firebase jest możliwość kliknięcia przycisku, który przeniesie użytkownika na stronę Fire Base, na której będzie możliwe połączenie projektu z Fire Base. Następnie należy kliknąć przycisk, który normalnie pojawia się pod Add Firebase Authentication SDK to your app. Powinny zostać dodane odpowiednie zależności, które umożliwią odwoływanie się do instancji Fire Base.

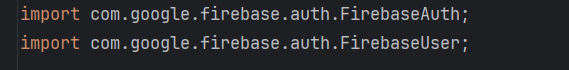


Na obrazku powyżej są przedstawione niezbędne zależności.

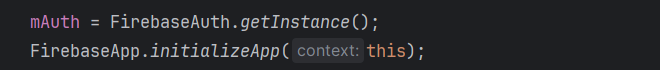


Ze względu na to, że użytkownik musi wybrać czy chce się zalogować czy zarejestrować jest możliwość kliknięcia guzików, które przenoszą do odpowiednich intencji, które pozwalają kolejno się zalogować lub zarejestrować.

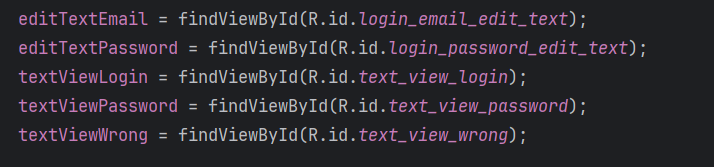
Po przeniesieniu użytkownika do intencji z logowaniem, udostępnione są mu dwa pola, w których może kolejno wpisać email oraz hasło następnie przycisk, który zatwierdza logowanie.



W celu poprawnego działania zaimportowano moduły FirebaseAuth oraz FirebaseUser.



Zainicjalizowano usługę Fire Base w tej intencji co pozwala na dostęp do usługi autoryzującej logowanie.



Zainicjalizowano elementy widoku. Pierwsze dwa pozwalają na wpisanie loginu oraz hasła przez użytkownika a następnie sczytanie ich. Następnie trzy pozwalają na powiadomienie użytkownika kolejno o tym, że brakuje loginu, hasła i dane nie są poprawne.



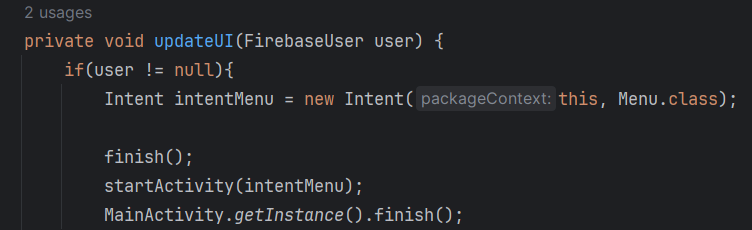
Zainicjalizowano element widoku, który pozwala po wciśnięciu na pobranie danych logowania z odpowiednich pól a następnie przesłanie ich do instancji Fire Base Authorizantion co pozwoli je zweryfikować.



Zainicjalizowano funkcję przycisku, która pozwala na wywołanie kodu po wciśnięciu go. Przedstawiony kod pobiera kolejno wartości dla loginu i hasła po czym zapisuje je jako zmienne. Następnie sprawdza czy, któreś pole nie jest puste a jeśli jest to wyświetla informacje o jego braku co jest wykonywane przez logikę podaną poniżej na obrazku.



Jeśli pola nie są puste zmienne przechowujące login i hasło przesyłane są do instancji mAuth, które z kolei przesyła dane na serwer w celu ich weryfikacji i zwraca informację czy logowanie się powiodło. W przypadku niepowodzenia logowania użytkownik jest informowany o nieprawidłowym loginie lub haśle. W celach bezpieczeństwa nie informuje się użytkownika czy ktoś o takim emailu już ma konto. W sytuacji gdy mAuth zwróci informację o poprawnym zalogowaniu tworzona jest instancja user, która aktualizuje intencje i przenosi do głównego menu.



Przedstawiona funkcja, jeżeli udało się przypisać użytkownika do instancji user, kończy prace aktualnej intencji i aktywuje nową z głównym menu.

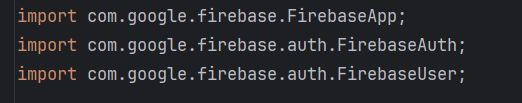
W głównym menu również zainicjalizowano odpowiednie importu.



Tworzona jest instancja Fire Base, która pozwala korzystać jej funkcji. Pobierany jest aktualnie zalogowany użytkownik. Jeżeli logowanie powiodło się pomyślnie to nie wywoła się dodatkowa logika ale w sytuacji przeciwnej intencja zostanie zamknięta. To samo stanie się jeśli użytkownik wyjdzie z aplikacji i wróci do niej. Dodatkowo jeżeli, w którymś momencie użytkownik zostanie wylogowany to główne menu zostanie zamknięte.

* 1. Rejestracja do aplikacji

Tak jak w sytuacji, która została przedstawiona w poprzednim podpunkcie gdy użytkownik może wybrać czy chce się zarejestrować czy zalogować, wybierzecie rejestracje zostanie przeniesiony do odpowiedniej intencji, która obsługuje to działanie.



Tak samo jak przy logowaniu zainicjalizowaniu odpowiednie importy.



Zainicjalizowano instancje FirebaseAuth oraz zainicjalizowano samo Fire Base.



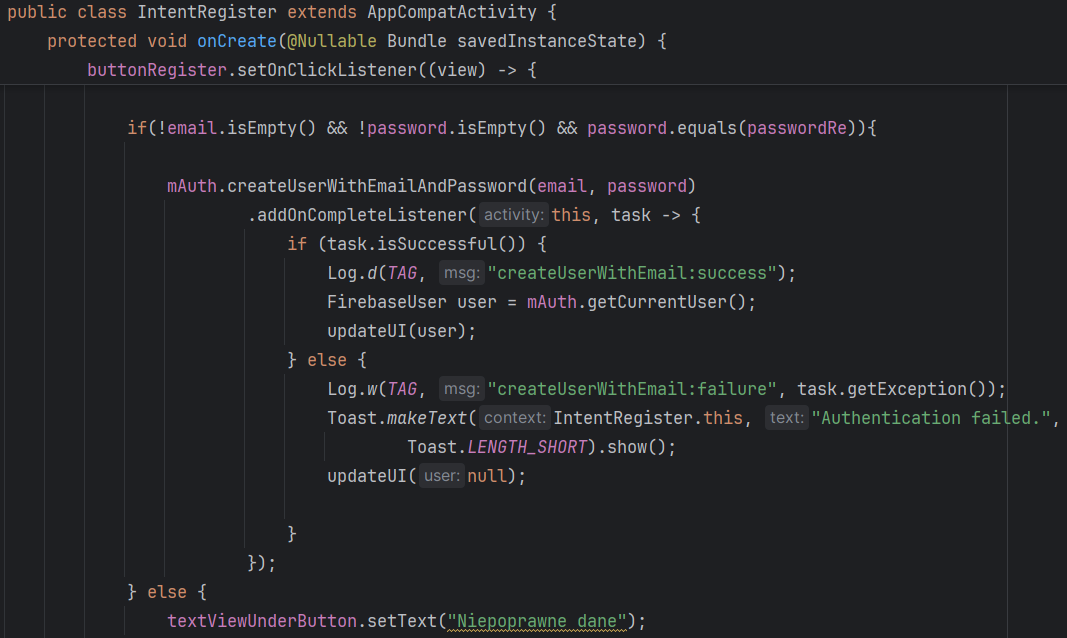
Inicjalizowanie widoków pozwalające na kliknięcie przycisku, wpisanie przez użytkownika emaila, hasła, powtórzonego hasła, poinformowanie użytkownika o niepoprawnych danych, mailu, haśle lub powtórzonym haśle.



Po wciśnięciu przycisku aktywowana jest logika, która pobiera dane z odpowiednich pól i zapisuje je jako zmienne.

Jeśli hasła się nie zgadzają użytkownik zostanie o tym poinformowany poprzez pokazanie TextView z odpowiednią informacją.

W przypadku jeśli hasła są takie same to tekst w polach na dane zostanie ustawiony na pusty.



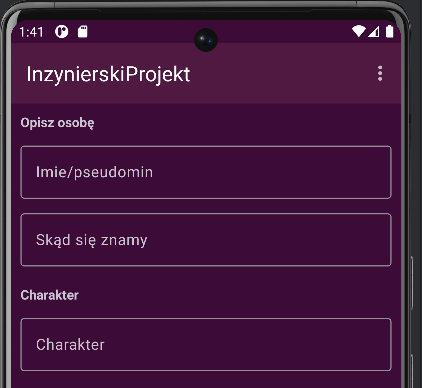
Po weryfikacji hasła jeśli są one takie same a login nie jest pusty to te dane zostaną przekazane do instancji FireBase, która prześle je dalej na serwer Fire Base. Jeśli hasła są puste, nie są takie same lub login jest pusty to wyświetli się komunikat w TextView o niepoprawnych danych.

W sytuacji uzupełnienia danych ale niespełnienia wymogów odpowiedniej długości znaków, wystarczające ilości cyfr i znaków specjalnych lub email nie będzie poprawny to użytkownik nie zostanie zarejestrowany.

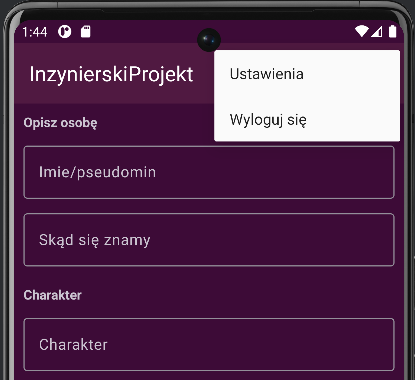
W sytuacji uzupełnienia poprawnych danych użytkownik zostanie zarejestrowany i przeniesiony jako zalogowany użytkownik do głównego menu.

* 1. Wylogowywanie z aplikacji

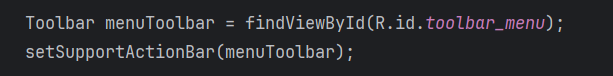
Wylogowanie z aplikacji jest możliwe tylko z intencji głównego menu po kliknięciu w 3 kropki w prawym górnych rogu co otworzy małe okienko.



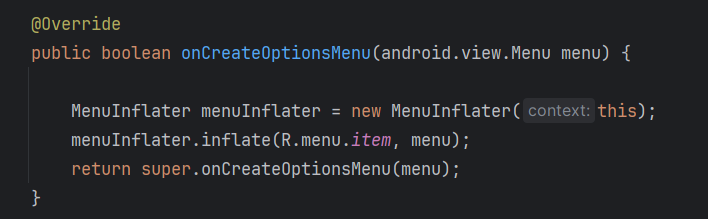
Będzie tam pozycja Wyloguje się. Po jej kliknięciu użytkownik zostanie wylogowany, główne menu zostanie zamknięte a użytkownik wróci do wyboru czy chce się zalogować czy założyć nowe konto.



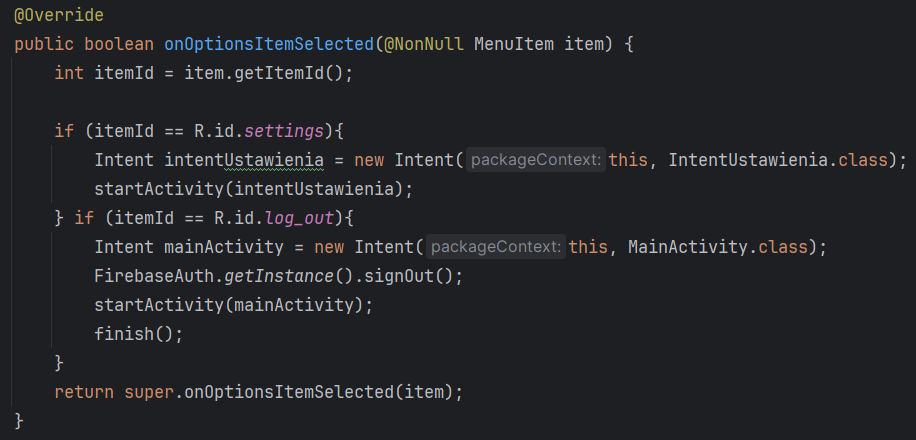
Logikę menu w której pojawiają trzy kropki opcji oraz okienko z opcjami obsługuje widok toolbar menu.



Przypisuje on zmiennej menuToolbar widok, który wywołuje właściwy widok co z kolei wywołuje funkcję onCreateOptionMenu.



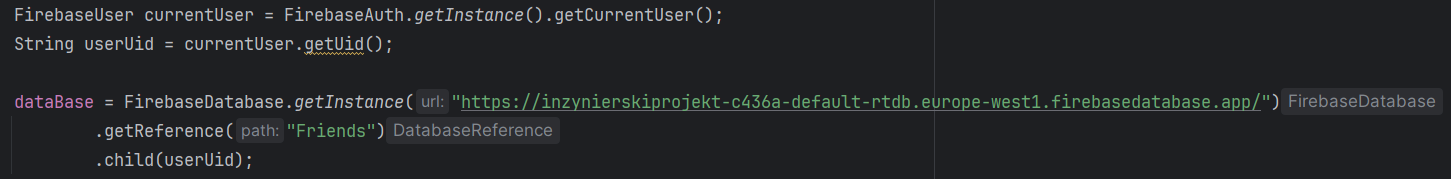
Funkcja ta tworzy item z wcześniej przygotowane layout, które to są rzeczonymi opcjami.



Obsługą logiki, która się aktywuje po kliknięciu, którejś z opcji zajmuje się onOptionItemSelected. Po wybraniu pozycji pobiera on jej wartość a następnie porównuje ją z id item. W naszej sytuacji wybrane zostaje log\_out. Wywołuje on funkcje signOut na instancji Fire Base co powoduje wylogowanie się aktualnie zalogowanego użytkownika , zamyka aktualnie otwartą intencje i otwiera tą z wyborem logowania lub rejestracji.

* 1. Tworzenie z opisem nowych znajomych do listy znajomych
  2. Lista znajomych

Lista znajomych generowana jest poprzez pobranie danych z bazy danych firebase dla aktualnie zalogowanego użytkownika. Dzieję się to poprzez poprzez dodanie osobnego rekordu z unikalnym Uid każdego użytkownika. W tym rekordzie zapisują się nowo dodani przyjaciele jako kolejne rekordy. Dzięki temu zapisane dane nie wyświetlą się dla innych użytkowników.

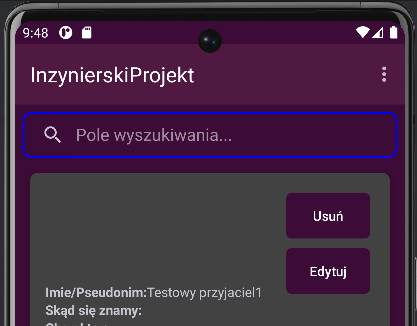


W celu pobrania odpowiednich danych niezbędne jest utworzenie instancji FirebaseAuth, z którego pobierany jest aktualnie zalogowany użytkownik oraz jego Uid.

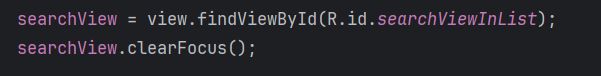
Następnie tworzona jest instancja FirebaseDatabase, do której przekazujemy referencje Friends, która zawiera wszystkich użytkowników, na której wywołujemy uzyskanie rekordu z nazwą zależną od aktualnego Uid.

* 1. Wyszukiwarka w liście przyjaciół

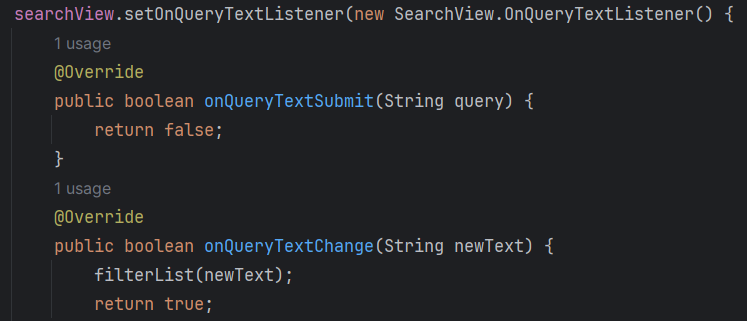
Wyszukiwanie odbywa się poprzez wpisanie interesującej użytkownika frazy do pola wyszukiwania w zakładce z listą przyjaciół.



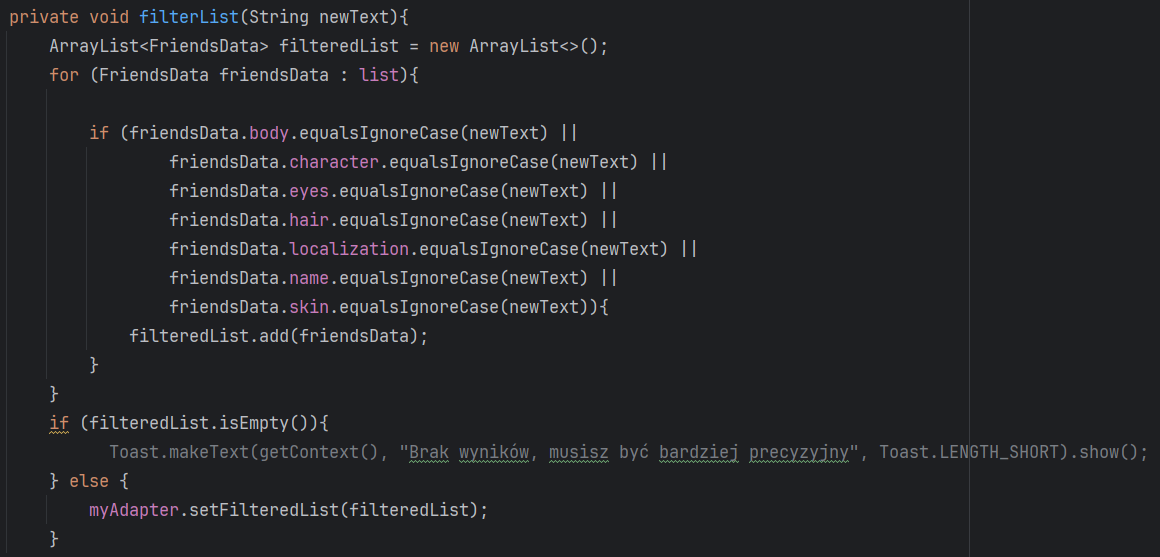
Znalezieniem odpowiedniego widoku zajmuje się searchView.



Jest on przypisywany do odpowiedniego widoku a następnie wywoływana jest na nim funkcja clearFocus, która zamknie klawiaturę jeżeli aplikacji wybierze domyślnie searchView w celu wpisania tam tekstu.



Następnie ustawiany jest nasłuchiwacz, który w momencie wpisania przez użytkownika tekstu do searchView przekaże go do funkcji filterList. Będzie się to działo po wpisaniu lub usunięciu każdej litery dla tego użytkownikowi na bieżąco będą się zmieniały widoki.



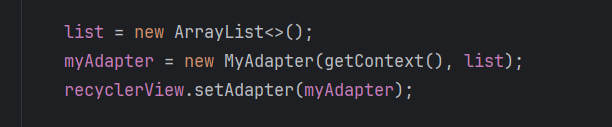
Funkcja filterList tworzy nową listę, do której zostaną przypisane te elementy widoku recycleView, które w polach tekstowy zawierają wyszukiwaną frazę. Dzieje się to w pętli for, która iteruje po aktualnej liście pobranej z bazy danych. Dla każdego obiektu sprawdzane jest czy wyszukiwana fraza odpowiada zapisanym danym. W przypadku gdy frazy są takie same obiekt zostanie przypisany do listy stworzonej na początku.

Na koniec jeżeli lista filteresList jest pusta to nic się nie stanie ale w przeciwnym razie zostanie przekazana do adaptera, gdzie zastąpi aktualnie wywoływaną listę.

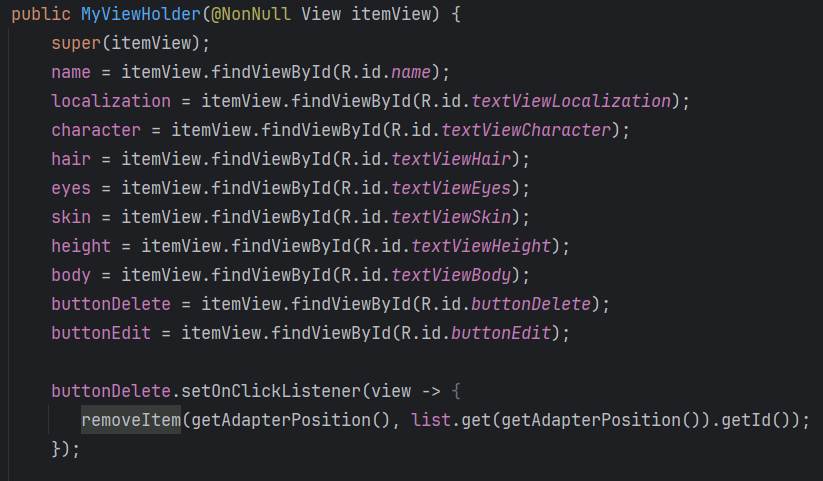
* 1. Edytowanie listy znajomych

* 1. Usuwanie wierszy w liście znajomych

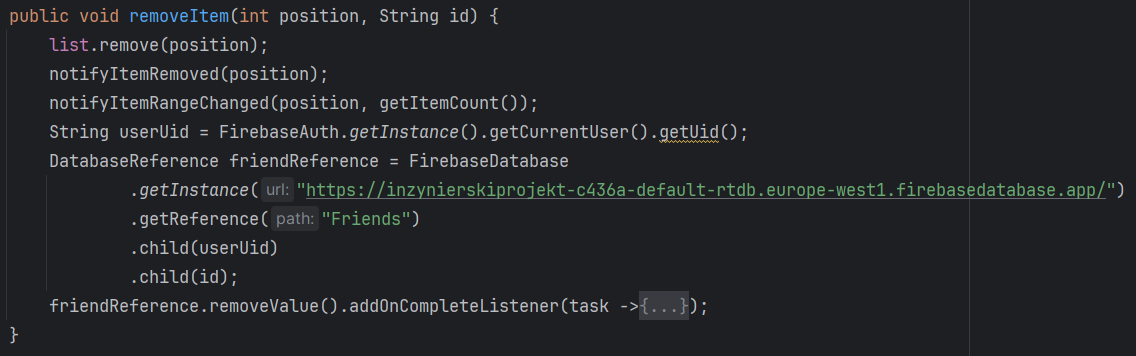
Usuwanie obsługiwane jest przez funkcje removeItem w klasie MyAdapter.



Jak na załączonym obrazku widać, trzeba przypisać obiekt adaptera do obiektu recycleView. Dzięki temu elementy recycleView będą miały dostęp do funkcji adaptera.



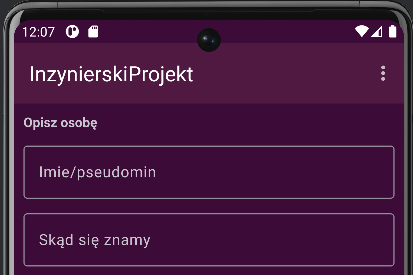
Wpierw przypisany zostaje przycisk, który obsługuję usunięcie pozycji. A następnie przypisanie do niego funkcji, która nasłuchuje moment wciśnięcia. Po kliknięciu wywoływana jest funkcja removeItem, do której przekazana jest pozycja elementu w recycleView oraz id konkretnego przyjaciela w bazie danych.



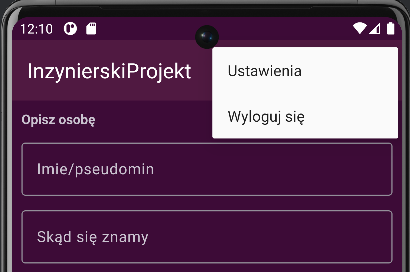
Następnie na liście, która aktualnie przechowuje dane dla recycleView usuwana jest odpowiednia pozycja. Po tym recycle view powiadamiany jest o usunięciu pozycji oraz o zmianie długości listy. Pobierany jest Uid aktualnie zalogowanego użytkownika i tworzony jest obiekt friendReference, który przechowuje jakie dane mają zostać usunięte. Na koniec na tym obiekcie wywoływana jest funkcja usuwająca te dane.

* 1. Ustawienia

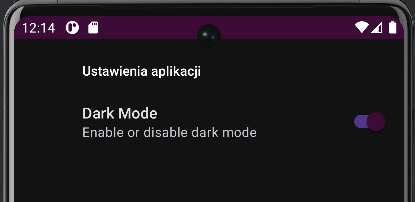
Ustawienia są zarządzane poprzez shared preferences. Przechowują one zbiór wartości w postaci klucz-wartość co pozwala na zapisywanie w nich ustawień, które mają być zastosowane w całej aplikacji.



W celu otworzenia ustawień należy najechać i kliknąć na 3 kropki w prawym górnym rogu aplikacji. Są one dostępne w głównym menu, logowaniu, rejestrowaniu czy wyborze czy użytkownik chce się zalogować czy zarejestrować.



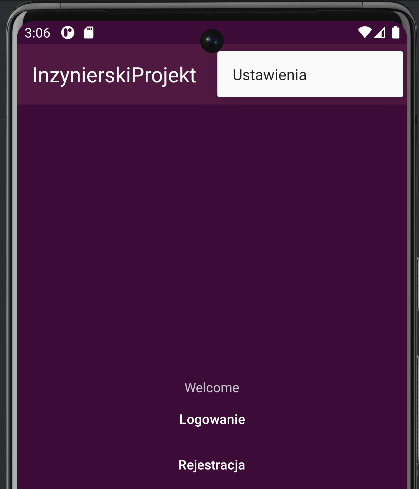
Po kliknięciu w 3 kropki należy wybrać pozycję ustawienia. Przeniesie ona użytkownika do nowej intencji, w której znajdują się rzeczone ustawienia.

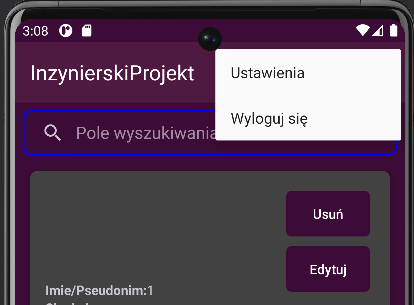


Z tego poziomu można wybrać interesujące nas ustawienie oraz powrócić do poprzedniego ekranu

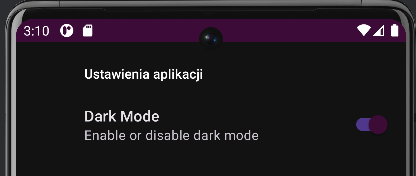
* 1. Zmiana trybu na dzienny lub nocny

Zmiana trybu dziennego na nocny i odwrotnie odbywa się w ustawieniach , do których dostęp jest w każdej aktywności.

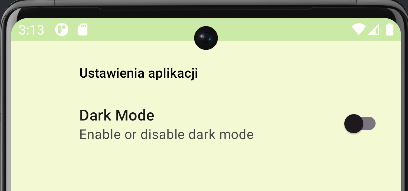




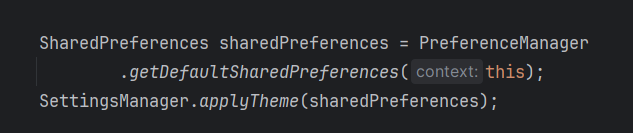
Po kliknięciu w Ustawienia pojawia się nowa intencja, w której mamy dostęp do zmiany trybu.



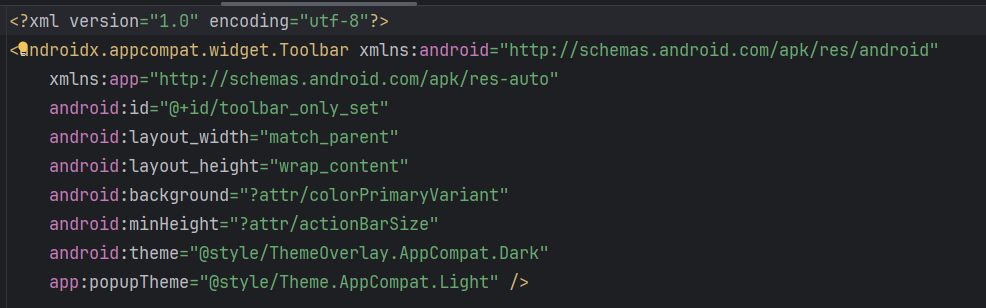
Zmiana trybu następuje po wciśnięciu przełącznika.



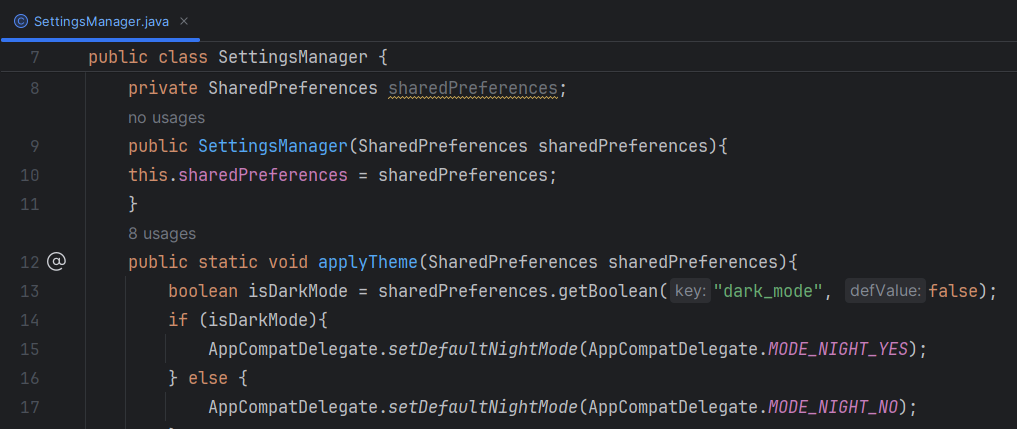
Działa to poprzez zmianę themes za pomocą shared preferences. Pozwalają one zapisać wartości w parach klucz-wartość.



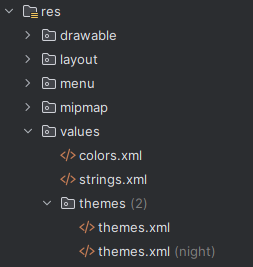
Przedstawiony kod tworzy instancję sharedPreferences, z której pobieramy domyślne ustawienia. Następnie wywołujemy funkcję applyTheme, która systemowo zmienia styl.



Na obrazku powyżej widać użycie colorPrimaryVariant, które znajduje się w obu stylach. Aplikacja przypisuje kolor z odpowiedniego stylu na podstawie, które ustawienia zostały zastosowane.

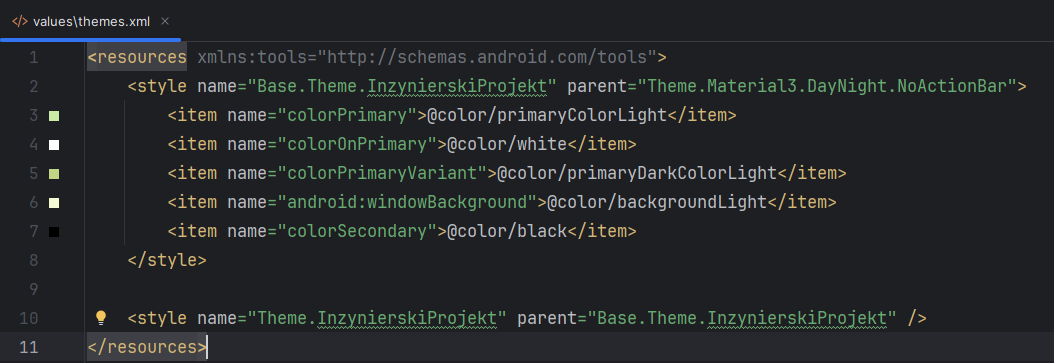


Dzieje się to w klasie SettingsManager. ApplyTheme sprawdza czy aktualnie ustawiony jest tryb nocny. Jeśli nie to aplikacja jest informowana o ustawieniu stylu dziennego i odwrotnie jeśli tak to ustawiany jest styl nocny.

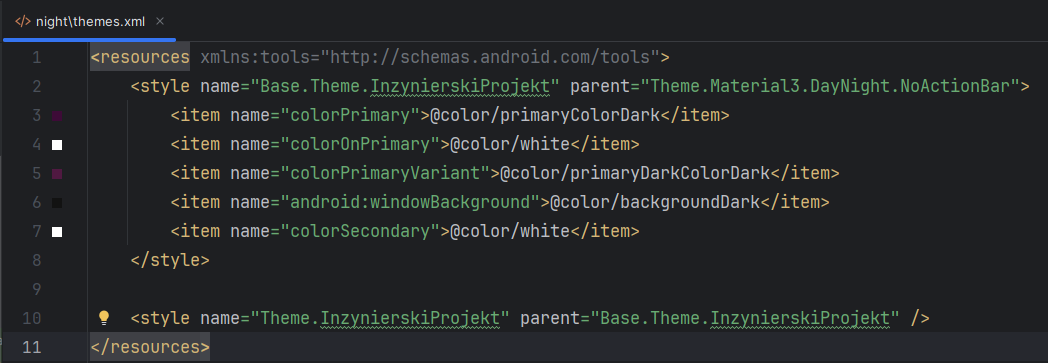


Style deklarowane są w katalogu res/values/themes.

Istnieją tam dwa pliki themes.xml. Jeden przechowuje wartości kolorów dla stylu dziennego a drugi dla nocnego.



W stylu opisane są imiona dla kolorów, których wartości są opisane w res/values/colors.xml. Na obrazku powyżej widać styl zawierający kolory dla trybu jasnego.



Analogicznie inne kolory są dodane w stylu dla trybu nocnego.

1. Testy i wdrożenie
   1. Testy jednostkowe
   2. Testy UI
   3. Bezpieczeństwo i dostępność aplikacji
   4. Metody dystrybucji aplikacji - napisać co najmniej jedną stronę
2. Podsumowanie i wnioski
   1. Wnioski
   2. Przyszłość aplikacji (następne kroki)

Następnymi krokami w rozwoju aplikacji jest dalszy rozwój polegający na:

* dodanie nowych funkcjonalności
* poprawa istniejących już funkcjonalności
* poprawienie ogólnego wyglądu aplikacji
* analiza dostępności aplikacji i wprowadzenie potrzebnych zmian w tym zakresie
* stworzenie wersji na system IOS

Do dodatkowych funkcjonalności planowane jest dodanie gry lub lobby, w którym można odwiedzić zapisane osoby, innej gry, która pozwala ćwiczyć imiona poznanych osób, opcji wybrania czy chce się zapisać nowego przyjaciela czy partnera biznesowego oraz możliwość stworzenia awatara w sytuacji kiedy nie posiadamy czyjegoś zdjęcia.

W istniejących funkcjonalność kiedy zapisywany jest nowy przyjaciel powinny być pytania pomocnicze, które mogłyby opisać daną osobę. Po zapisaniu danych puste TextView powinny się chować. Lista przyjaciół powinna mieć różne podgrupy na przyjaciół. Ekran wyboru logowania lub rejestracji jest niepotrzebny i powinien być zastąpiony przez ekran logowania z możliwością przejścia do rejestracji.

Ustawienia powinny zawierać więcej opcji takich jak zmiana stylu czcionki, zmiana języka czy zarządzanie powiadomieniami.

Shared Preferences, które przechowuje ustawienia będzie zamieniona na Data Store, które jest nowszą wersją w pakiecie Android Jetpack.[[5]](#footnote-4)

Przy wyglądzie aplikacji na pewno trzeba zastanowić się nad kolorami czy są zachęcające i przyjemne dla oka oraz czy nie sprawiają ludziom problemów z czytelnością. Zastanowienia wymaga wygląd elementów widoków, kształt i odległości między nimi. Wzorem tego na pewno powinien być design material m3.

Bardzo ważne by aplikacje były dostosowane dla ludzi, którzy różne problemy motoryczne czy poznawcze. Dlatego by nikogo nie wykluczać należy dokładnie przeanalizować aplikację a następnie wprowadzić potrzebne zmiany. Dzięki takim zmianom aplikacja sama w sobie może stać się czytelniejsza, czy przyjaźniejsza dla każdego użytkownika.

Ze względu na cel dotarcia do jak największej grupy odbiorczej nie można zapomnieć o użytkownikach iphona. Są oni dużą częścią rynku przez co naturalne jest to, że powstanie wersja aplikacji na IOS.

Bibliografia

Źródła z literatury:

Źródła cyfrowe:

Do napisania może:

Napisać o archiwizacji też od razu można o apk napisać

Dokumentacja prawna

Opisać uprawnienia np do internetu czy do zdjęć

Użyć ProGuard może

Może użyć dex2jar

MobFs może użyć - chyba sprawdza jak bezpieczna jest aplikacja

1. Sylwia Gostkowska-Dźwig, Ewa Kempa, Magdalena Mrozik, Ryszard Królik, Wybrane metody analizy w procesie zarządzania strategicznego, Wydawca Politechnika Częstochowska [↑](#footnote-ref-0)
2. Tony Gaddis, *Java dla zupełnie początkujących. Owoce programowania. Wydanie VII*, Wydawnictwo Helion [↑](#footnote-ref-1)
3. <https://docs.github.com/en/get-started/start-your-journey/about-github-and-git> 21.11.2024 [↑](#footnote-ref-2)
4. <https://docs.github.com/en/get-started/using-git/about-git> 24.11.2024 [↑](#footnote-ref-3)
5. <https://developer.android.com/topic/libraries/architecture/datastore?hl=pl> 24.11.2024 [↑](#footnote-ref-4)