**Sprawozdanie z aplikacji Cocktails**

Cały kod jest też dostępny na repozytorium: [Prorokslawek/MobileAplication-Cocktails](https://github.com/Prorokslawek/MobileAplication-Cocktails/tree/main)

**1. Cel i funkcjonalność aplikacji**

Aplikacja **Cocktails** to narzędzie mobilne służące do przeglądania przepisów na koktajle alkoholowe i bezalkoholowe. Strona główna odpowiada za przywitanie użytkownika oraz krótki opis zakładek. Zawiera też przycisk służący do przełączania się pomiędzy jasnym/ciemnym motywem. Aplikacja zawiera szufladę nawigacyjną i umożliwia poruszanie się między zakładkami za pomocą gestu przeciągnięcia palca.

Główne funkcjonalności:

* **Przeglądanie koktajli** w trzech kategoriach: strona główna, drinki alkoholowe, drinki bezalkoholowe.
* **Szczegóły koktajlu**: wyświetlanie pełnego opisu, składników i obrazu drinka.
* **Motyw kolorystyczny**: przełączanie między trybem jasnym a ciemnym.
* **Nawigacja**:
  + Za pomocą **zakładek (TabRow)** i gestów przesuwania palcem (**HorizontalPager**).
  + **Szuflada nawigacyjna** z bezpośrednim dostępem do wszystkich sekcji.
* **Zegar**: odliczanie czasu przygotowania drinka (integracja **z TimerViewModel**).
* **Urządzenia**: Aplikacja działa poprawnie zarówno dla telefonów jak i tabletów. Na każdej stronie możliwa jest zmiana orientacji urządzenia.
* **Baza drinków**: Lista drinków wraz z opisami przygotowania, zdjęciami oraz składnikami pobierana jest z API: [Free Cocktail API - TheCocktailDB.com](https://www.thecocktaildb.com/api.php) i odpowiednio przerabiana według potrzeb.

**2. Architektura techniczna**

Aplikacja oparta jest na:

* **Jetpack Compose** – nowoczesny framework UI dla Android.
* **Biblioteki Accompanist** – wsparcie dla pagera i animacji.
* **Coil** – ładowanie obrazów z sieci.

**3. Główne komponenty i ich odpowiedzialności**

**Pliki kodu źródłowego**

1. **MainActivity.kt**
   * **Rola**: Główna aktywność aplikacji, inicjalizuje UI i motyw.
   * **Kluczowe elementy**:
     + setContent – konfiguruje kompozycję UI.
     + installSplashScreen – obsługa animowanego ekranu startowego.
     + Integracja z ThemeViewModel i TimerViewModel.
2. **AppNavigation.kt**
   * **Rola**: Zarządza nawigacją między ekranami i szufladą nawigacyjną.
   * **Kluczowe elementy**:
     + ModalNavigationDrawer – implementacja szuflady nawigacyjnej.
     + NavHost – kontener dla ścieżek nawigacyjnych.
     + NavController – steruje przejściami między ekranami.
     + Integracja z CocktailListViewModel do synchronizacji zakładek.
3. **CocktailListScreen.kt**
   * **Rola**: Wyświetla listę koktajli w formie zakładek.
   * **Kluczowe elementy**:
     + HorizontalPager – umożliwia przewijanie między zakładkami.
     + TabRow – pasek z nazwami zakładek.
     + LazyVerticalGrid / LazyColumn – lista koktajli (różne układy dla telefonów i tabletów).
     + CocktailCard – karta z podglądem koktajlu.
4. **CocktailDetailScreen.kt**
   * **Rola**: Pokazuje szczegóły wybranego koktajlu.
   * **Kluczowe elementy**:
     + TimerViewModel – obsługa odliczania czasu przygotowania.
     + AsyncImage – wyświetlanie obrazu koktajlu.
5. **ThemeViewModel.kt**
   * **Rola**: Zarządza trybem ciemnym/jasnym.
   * **Kluczowe elementy**:
     + MutableStateFlow<Boolean> – przechowuje stan motywu.
     + toggleTheme() – przełącza między trybami.
6. **CocktailListViewModel.kt**
   * **Rola**: Synchronizuje stan zakładek między szufladą nawigacyjną a HorizontalPager.
   * **Kluczowe elementy**:
     + MutableStateFlow<Int> – przechowuje indeks aktualnej zakładki.
7. **CocktailApiService.kt**

Plik CocktailApi.kt to część aplikacji odpowiedzialna za komunikację z zewnętrznym serwerem API (w tym przypadku serwisem TheCocktailDB). Dzięki temu plikowi aplikacja może pobierać dane o koktajlach z internetu. Ten plik wykorzystuje bibliotekę Retrofit, która jest popularnym narzędziem w Androidzie do obsługi żądań HTTP.

Dodaje konwerter JSON (GsonConverterFactory) – automatycznie zamienia dane JSON na obiekty Kotlin i odwrotnie.

**4. Nawigacja i przepływ danych (Jetpack Compose Navigation)**

Fragmenty zostały całkowicie zastąpione przez funkcje Composable oraz mechanizm Jetpack Compose Navigation:

* Funkcje takie jak CocktailDetailScreen pełnią rolę widoków (ekranów).
* Cyklem życia zarządza Compose za pomocą efektów side-effect (LaunchedEffect, remember).
* Przejścia między ekranami są realizowane za pomocą NavHost i NavController.

To podejście upraszcza architekturę aplikacji, zwiększa jej wydajność i elastyczność oraz pozwala na łatwiejsze dostosowanie interfejsu do różnych urządzeń!

* **Synchronizacja stanów**:
  + CocktailListViewModel łączy szufladę nawigacyjną z HorizontalPager.
  + Zmiana zakładki w szufladzie aktualizuje selectedTabIndex w ViewModel, co wymusza przejście w HorizontalPager (i na odwrót).
  + LaunchedEffect w CocktailListScreen zapewnia synchronizację między pagerState a selectedTabIndex.

**5. Obsługa motywu kolorystycznego**

* **Theme.kt**: Definiuje kolory, typografię i kształty dla trybu jasnego/ciemnego.
* **ThemeViewModel**:
  + Przechowuje stan motywu w isDarkTheme: StateFlow<Boolean>.

**6. Wyzwania i rozwiązania**

1. **Problem**: Niesynchronizowane przejścia między szufladą a zakładkami.
   * **Rozwiązanie**: Dwukierunkowa synchronizacja stanów za pomocą LaunchedEffect i collectAsStateWithLifecycle.
2. **Problem**: Biały ekran po animacji splash screen.
   * **Rozwiązanie**: Dodanie setKeepOnScreenCondition w MainActivity, które opóźnia ukrycie splash screen do momentu załadowania danych.

**7. Zegar**

* Umieszczenie na przyciskach ikon zamiast napisów na przyciskach zostały umieszczone odpowiedni ikony(start,stop,reset)
* Dodatkowe funkcje – został dodany przycisk rozszerzający stoper na cały ekran.
* Możliwość ustawienia odliczanego czasu z dokładnością do jednej sekundy

**8. FAB (floating action button)**

* Dla każdego drinka został dodany dynamiczny przycisk akcji, pozwalający na wysłanie SMS’em składników wybranego drinka.

**9. Animacja**

* Przy uruchamianiu aplikacji pojawia się krótka animacja loga. Obraz podnosi się do góry następnie lekko zmienia kąt i zmniejszając się zanika na środku ekranu w końcu uruchamiając aplikację.

**10. Zmiana orientacji**

Aplikacja samodzielnie obsługuje obrót ekranu (nie restartuje się przy zmianie orientacji):

android:configChanges="orientation|screenSize|screenLayout|keyboardHidden" ,  
override fun onConfigurationChanged(newConfig: Configuration) {  
 super.onConfigurationChanged(newConfig)  
}