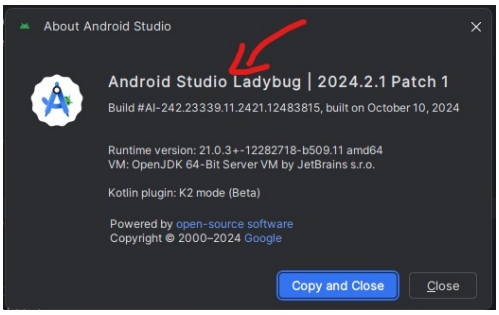
Übung & Leistungsnachweis:

Coroutinen und Services

In diesem Arbeitsblatt lernst du die grundlegenden Konzepte der Coroutinen und Services in der Android-Entwicklung kennen. Mit einer einfachen Wetter-App wirst du untersuchen, wie man asynchrone Aufgaben ausführt und Daten effizient abruft, ohne die Benutzeroberfläche zu blockieren. Ziel ist es, die Funktionsweise von Coroutinen und Services praktisch zu erfahren und die App um diese Funktionen zu erweitern.

1. **Schritt 1: Projekt klonen & konfigurieren (0 Punkte)**



A) Öffne Android Studio. Zur Vermeidung möglicher Probleme empfehlen wir die Verwendung von Android Studio Ladybug. Überprüfe deine Version, indem du zu Help > About gehst.

B) Klicke auf File > New > Project from Version Control.

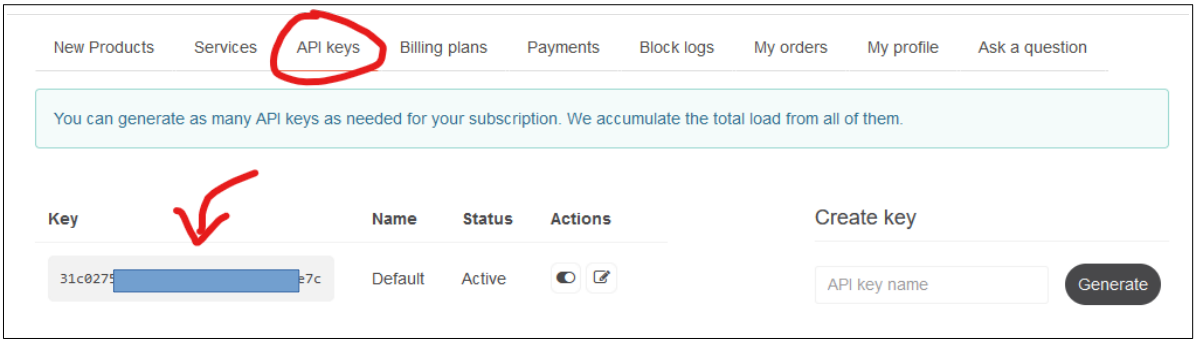
Gib die folgende URL ein, um das Projekt zu klonen:
https://github.com/julianertle/CoroutinesAndServices
Klicke auf Clone und warte, bis das Projekt heruntergeladen wurde.

C) Öffne das geklonte Projekt in Android Studio und starte die App auf einem Emulator oder einem echten Gerät.

D) Öffne das Menü in der App und gehe zu den Einstellungen. Definiere eine Beispielstadt und speichere diese.

E) Nach dem Speichern der Stadt wird eine Fehlermeldung angezeigt, da der API-Token für OpenWeather noch nicht konfiguriert wurde.

F) API-Token einfügen:
Registriere dich auf https://home.openweathermap.org/api_keys, um deinen persönlichen API-Token zu erhalten.



G) Kopiere den API-Token und füge ihn in den Einstellungen der App ein, um die Verbindung zu OpenWeather herzustellen.

H) Teste die App, indem du die verschiedenen Funktionen verwendest. Überprüfe, ob die Stadtinformationen korrekt angezeigt werden und ob die App nach der Konfiguration des API-Tokens ordnungsgemäß funktioniert.

---

2. **Implementierung der Methode fetchForecast** **(4 Punkte)**

   Öffne die Datei **WeatherApiService.kt** In dieser Datei befindet sich bereits die Methode fetchWeather, die aktuelle Wetterdaten abruft.

   A) Implementiere die Methode `fetchForecast`, die die Vorhersagedaten abruft und zurückgibt, ähnlich wie die Methode `fetchWeather`. Verwende das `suspend`-Schlüsselwort, um die Methode asynchron auszuführen, und wähle einen Dispatcher, der besser geeignet ist als `Dispatchers.Default`.

---

3. **Implementierung der Methode fetchForecastData** **(7 Punkte)**

   Öffne die Datei **WeatherViewModel.kt**

   A) Implementiere die Methode fetchForecastData mit den folgenden Parametern:
   - city: Der Name der Stadt
   - apiKey: Der API-Schlüssel, der für den Zugriff auf OpenWeather

   B) Verwende eine Coroutine, um den API-Aufruf asynchron auszuführen, damit die Benutzeroberfläche während des Abrufs nicht blockiert wird.

   C) Wenn der Abruf der Vorhersagedaten erfolgreich ist, speichere die Vorhersagedaten in der _forecast-Variable und setze die Fehlermeldung auf null.

   D) Wenn der Abruf fehlschlägt oder eine Exception auftritt, setze eine entsprechende Fehlermeldung in _errorMessage.

   Beispiel:
   - Speichere die erhaltenen Daten mit der Zeile:

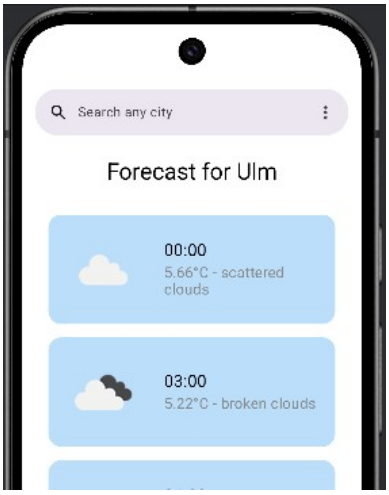
   _forecast.value = forecastResponse.list.

   - Wenn eine Ausnahme auftritt, setze in _errorMessage eine Fehlermeldung, die dem Benutzer angezeigt wird. Andernfalls setze

   _errorMessage.value = null.

---

4. **Anzeige der Wettervorhersage (3 Punkte)**



Öffne die Datei **ForecastWeatherView.kt** und füge eine Liste hinzu, die die abgefragten Wettervorhersagedaten anzeigt.

Nutze dafür die **WeatherCard** Komponente.

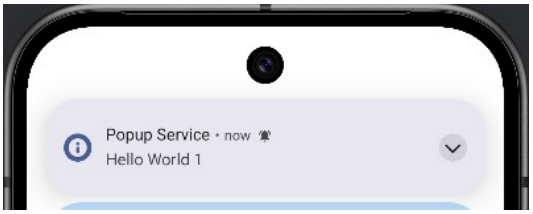
Hinweis: Die Wetterdaten werden der Klasse bereits als Übergabeparameter mitgegeben.

---

5. **Implementierung eines Foreground Services (4 Punkte)**

Gehe zu MainActivity.kt und aktiviere den PopupService im Code.

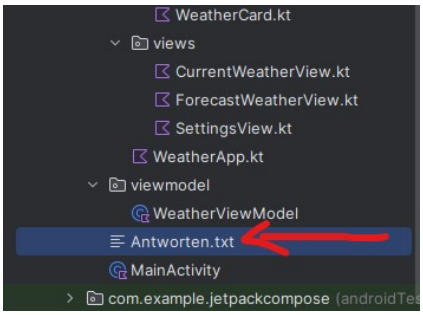
A) Implementiere nun die Methode **onCreate()** in der Datei **PopupService.kt** vollständig. Der PopupService soll als Foreground-Service gestartet werden. Nutze dafür eine zusätzliche Methode in derselben Klasse.



Wähle in den App-Einstellungen die Timer Option 10s aus, nun solltest du alle 10 Sekunden eine Notification erhalten

---

6. **Fragen: (8 Punkte)**



Schreibe deine Antworten aus den folgenden Fragen in die **Antworten.txt** Datei in deinem Projekt.

A) Welche Vorteile bieten Coroutinen in der Android-Entwicklung und wie tragen sie dazu bei, asynchrone Aufgaben effizienter und benutzerfreundlicher zu gestalten?

B) Erkläre warum du dich für einen bestimmten Dispatcher in Aufgabe 2A) entschieden hast.

C) Nenne zwei praxisrelevante Anwendungsfälle, bei denen der Einsatz von Services sinnvoll ist. Erkläre kurz warum.

D) Welche Vorteile bietet die Kombination von Services und Coroutinen? Kannst du ein Beispiel aus dem Code der Wetter-App nennen, in dem beide miteinander kombiniert genutzt werden?

7. **Allgemeine Kriterien:** **(4 Punkte)**

   A) Codequalität:

      1. Achte darauf, dass der Code sauber, gut strukturiert und gut lesbar ist. Verwende sinnvolle Variablennamen, Kommentare und halte dich an gängige Best Practices.

   B) Dokumentation:

      1. Dokumentiere den Code, insbesondere komplexe Funktionen und Logik. Beschreibe, was die Funktionen tun, welche Parameter erwartet werden und was zurückgegeben wird.

   C) Fehlerbehandlung:

      1. Stelle sicher, dass Fehler und Ausnahmen korrekt behandelt werden. Verwende Try-Catch-Blöcke, prüfe Eingabewerte und stelle sicher, dass der Code nicht unerwartet abstürzt, sondern eine hilfreiche Fehlermeldung oder Alternative bietet.

   D) Kompilierbarkeit:

      1. Stelle sicher, dass die App fehlerfrei kompiliert und beim Start keine Abstürze auftreten. Achte darauf, dass alle Abhängigkeiten korrekt konfiguriert sind und keine fehlenden oder inkonsistenten Code-Teile vorliegen, die zu einem Fehler beim Kompilieren oder Ausführen führen könnten.

Abgabe bis zum 02. Januar. 2025 um 23:55. Bitte den Link zu deinem Repository mit dem Commit-Hash im Briefkasten ablegen.

Unter https://github.com/julianertle/CoroutinesAndServices/commit findest du deine Commits. Eine URL in der Form https://github.com/<owner>/<project>/commit/<hash> zeigt die Änderungen, die mit diesem Commit eingeführt wurden.

Bsp.:
https://github.com/julianertle/CoroutinesAndServices/commit/5e2ad317a9c3cae2bd2cf79bcc0530853a8a844b