Mendapatkan data dari Internet dengan retrofit

Berikut adalah tutorial untuk membuat aplikasi sederhana di Jetpack Compose yang akan mengambil data dari internet. Dalam contoh ini, kita akan membuat aplikasi yang mendapatkan data JSON dari sebuah API menggunakan Retrofit. Contoh ini menggunakan **ViewModel** untuk memisahkan logika pengambilan data dan menambahkan tampilan **Card** untuk setiap item posting.

Langkah 1: Tambahkan Dependensi Retrofit dan Gson

Di `build.gradle (Module :app)`, tambahkan dependensi untuk Retrofit dan Gson:

```
dependencies {
    . . .
    // Retrofit dan Gson
    implementation ("com.squareup.retrofit2:retrofit:2.9.0")
    implementation ("com.squareup.retrofit2:converter-gson:2.9.0")
}
```

Langkah 2: Tambahkan Dependensi ViewModel

Pastikan Anda memiliki dependensi berikut di `build.gradle` untuk menggunakan ViewModel:

```
dependencies {
   . . .
   // ViewModel
   implementation ("androidx.lifecycle:lifecycle-viewmodel-compose:2.8.7")
}
```

Klik **Sync**

Langkah 3: Tambahkan ViewModel untuk Pengambilan Data

Kita akan membuat `MainViewModel.kt` untuk memisahkan logika pengambilan data dari UI.

```
import androidx.compose.runtime.State
import androidx.compose.runtime.mutableStateOf
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import kotlinx.coroutines.launch

class MainViewModel : ViewModel() {
    private val _posts = mutableStateOf<List<Post>>(emptyList())
    val posts: State<List<Post>> = _posts

    init {
```

```
        fetchPosts()
    }

    private fun fetchPosts() {
        viewModelScope.launch {
            try {
                // Mengambil data dari API
                val postList = RetrofitInstance.api.getPosts()
                _posts.value = postList
            } catch (e: Exception) {
                // Tangani kesalahan (misalnya dengan menampilkan pesan
error)
                _posts.value = emptyList()
            }
        }
    }
}
```

- `MainViewModel` mengelola pengambilan data menggunakan `viewModelScope` untuk menjalankan coroutine.
- `fetchPosts` secara otomatis memuat data saat `MainViewModel` diinisialisasi.

Langkah 4: Tambahkan Izin Internet

Tambahkan izin internet di `AndroidManifest.xml` tepat di atas tag `<application>`:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Langkah 5: Membuat Data Model

Misalnya kita akan menggunakan JSON Placeholder untuk mengambil daftar posting. Berikut adalah model data yang sesuai dengan struktur JSON.

Buat file `Post.kt`:

```
// Post.kt
data class Post(
    val userId: Int,
    val id: Int,
    val title: String,
    val body: String
)
```

Langkah 6: Siapkan Retrofit

Buat file bernama `ApiService.kt` untuk mendefinisikan endpoint API dan mengatur instance Retrofit.

```
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
import retrofit2.http.GET

// 1. Definisikan endpoint API
interface ApiService {
    @GET("posts")
    suspend fun getPosts(): List<Post>
}

// 2. Inisialisasi Retrofit
object RetrofitInstance {
    private const val BASE_URL = "https://jsonplaceholder.typicode.com/"

    val api: ApiService by lazy {
        Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
            .create(ApiService::class.java)
    }
}
```

- Di sini, `ApiService` memiliki satu endpoint `getPosts()` untuk mendapatkan daftar posting.
- `RetrofitInstance` mengatur instance Retrofit yang dapat digunakan di seluruh aplikasi.

Langkah 5: Menampilkan Data di MainActivity dengan ViewModel dan Tampilan Card

Selanjutnya, gunakan ViewModel ini di `MainActivity.kt` dan tampilkan data dengan **Card** di Compose.

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.PaddingValues
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
```

```
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material3.Card
import androidx.compose.material3.CardDefaults
import androidx.compose.material3.CircularProgressIndicator
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.material3.TopAppBar
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import androidx.lifecycle.viewmodel.compose.viewModel
import com.example.internet2.ui.theme.Internet2Theme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Internet2Theme {
                Surface(modifier = Modifier.fillMaxSize()) {
                    MyApp()
                }
            }
        }
    }
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun MyApp(mainViewModel: MainViewModel = viewModel()) {
    val posts by mainViewModel.posts

    Scaffold(
        topBar = { TopAppBar(title = { Text("Posts") }) },
        content = {
            if (posts.isEmpty()) {
                CircularProgressIndicator(modifier =
Modifier.fillMaxSize().padding(16.dp))
            } else {
                PostList(posts = posts, contentPadding = it)
            }
        }
    )
}

@Composable
fun PostList(posts: List<Post>, contentPadding: PaddingValues =
PaddingValues(16.dp)) {
```

```
    LazyColumn(modifier = Modifier.fillMaxSize().padding(top = 80.dp)) {
        items(posts){ post ->
            PostItem(post = post)
            Spacer(modifier = Modifier.height(8.dp))
        }
    }
}

@Composable
fun PostItem(post: Post) {
    Card(
        modifier = Modifier.fillMaxWidth().padding(8.dp),
        elevation = CardDefaults.cardElevation(4.dp)
    ) {
        Column(modifier = Modifier.padding(16.dp)) {
            Text(text = post.title, style =
MaterialTheme.typography.headlineMedium)
            Spacer(modifier = Modifier.height(4.dp))
            Text(text = post.body, style =
MaterialTheme.typography.bodyLarge)
        }
    }
}
```

Penjelasan Kode

1. **ViewModel Integration**: `MyApp` sekarang menerima `mainViewModel` sebagai parameter. Data `posts` akan otomatis di-update ketika ViewModel mengambil data.
2. **Loading Indicator**: Menampilkan `CircularProgressIndicator` ketika data masih kosong.
3. **Card Layout**: `PostItem` sekarang menampilkan setiap `post` dalam `Card` dengan padding dan jarak antar item.