Kotlin

MOBILE PROGRAMMING

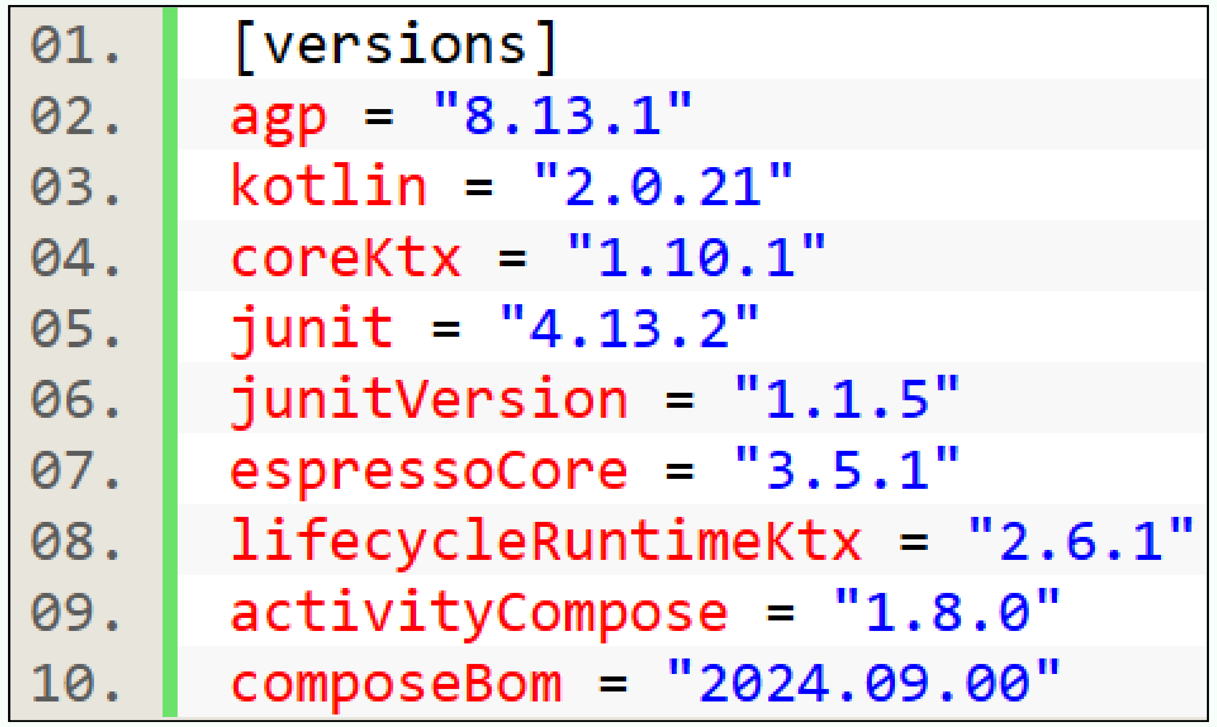
KOTLIN SERIALIZATION AND RETROFIT

ROBBY TAN

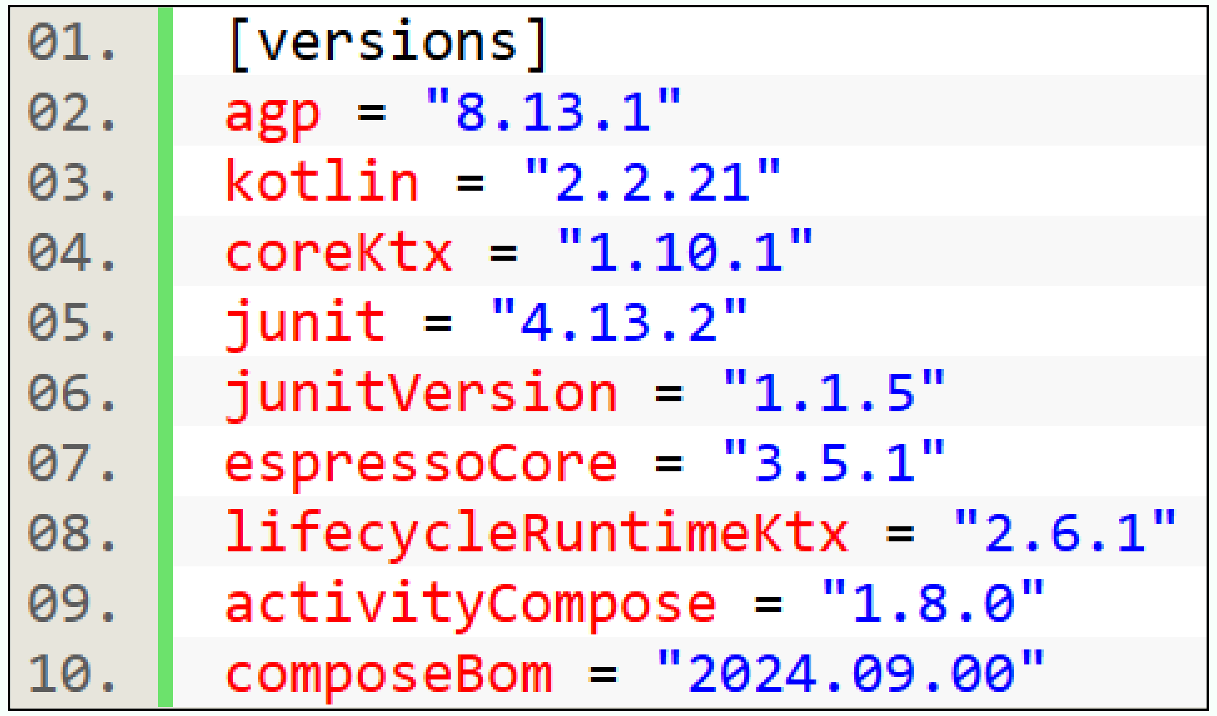
Kotlin Serialization

Update #1

- Buka **libs.versions.toml**
- Kode di bawah berdasarkan versi *default* yang dibuat dengan menggunakan Android Studio Otter | 2025.2.1 Patch 1.
- Ubah versi **kotlin** dari **2.0.21** menjadi **2.2.21**.
- Lakukan **Gradle sync**.



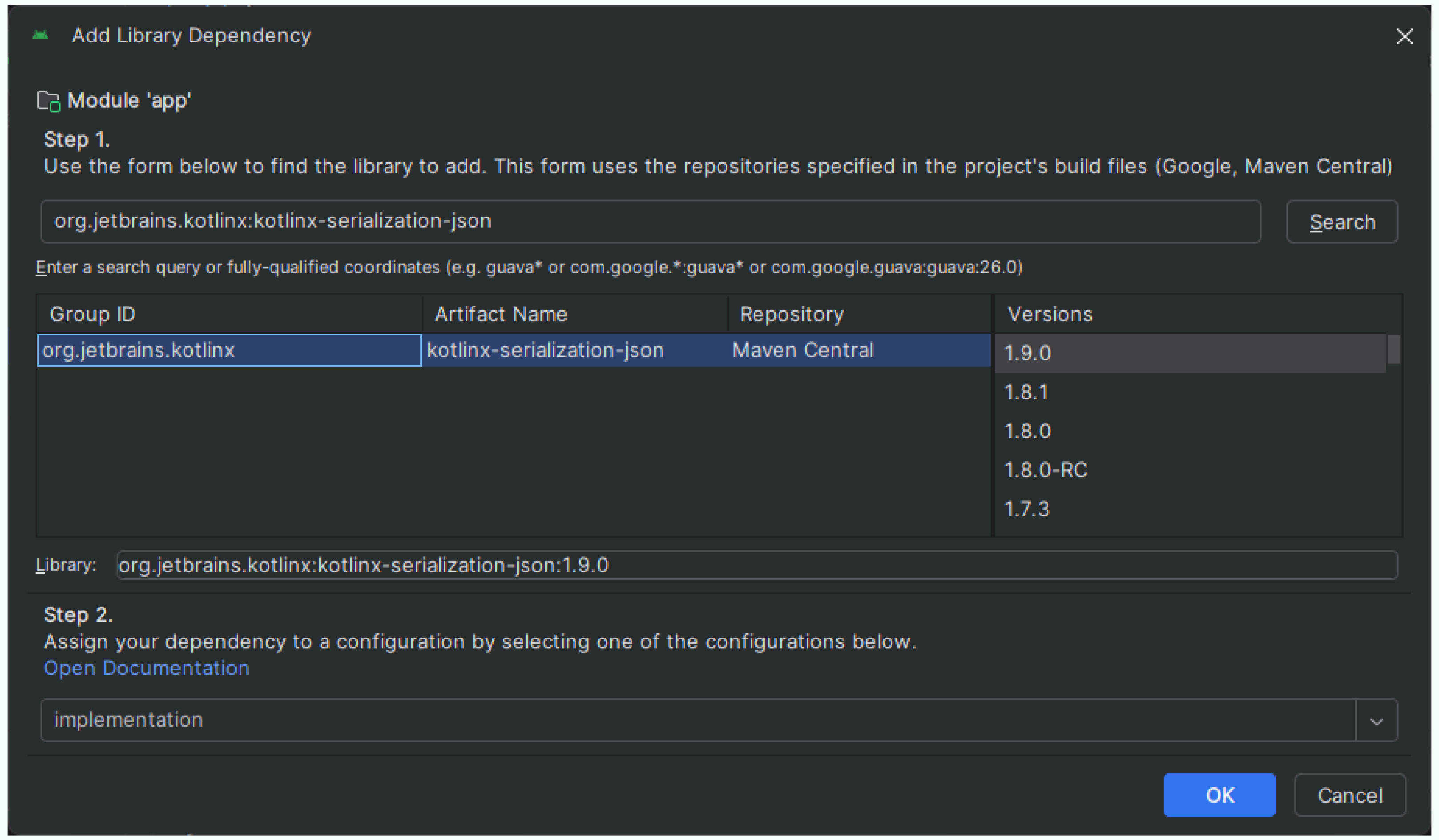
```toml
[versions]
agp = "8.13.1"
kotlin = "2.0.21"
coreKtx = "1.10.1"
junit = "4.13.2"
junitVersion = "1.1.5"
espressoCore = "3.5.1"
lifecycleRuntimeKtx = "2.6.1"
activityCompose = "1.8.0"
composeBom = "2024.09.00"
```



```toml
[versions]
agp = "8.13.1"
kotlin = "2.2.21"
coreKtx = "1.10.1"
junit = "4.13.2"
junitVersion = "1.1.5"
espressoCore = "3.5.1"
lifecycleRuntimeKtx = "2.6.1"
activityCompose = "1.8.0"
composeBom = "2024.09.00"
```

Add Dependencies

- Library: **org.jetbrains.kotlinx:kotlinx-serialization-json**
- Versi yang digunakan adalah versi **1.9.0**.
- Jika **tidak melakukan *update*** versi Kotlin, maka versi yang digunakan adalah versi **1.7.3**.
- Bandingkan versi Kotlin dan Kotlin serialization
  - https://mvnrepository.com/artifact/org.jetbrains.kotlinx/kotlinx-serialization-json
  - https://kotlinlang.org/docs/releases.html#release-details


Add Library Dependency

Module 'app'

Step 1.
Use the form below to find the library to add. This form uses the repositories specified in the project's build files (Google, Maven Central)

org.jetbrains.kotlinx:kotlinx-serialization-json

Enter a search query or fully-qualified coordinates (e.g. guava* or com.google.*:guava* or com.google.guava:guava:26.0)

Group ID	Artifact Name	Repository	Versions
org.jetbrains.kotlinx	kotlinx-serialization-json	Maven Central	1.9.0
			1.8.1
			1.8.0
			1.8.0-RC
			1.7.3

Library: org.jetbrains.kotlinx:kotlinx-serialization-json:1.9.0

Step 2.
Assign your dependency to a configuration by selecting one of the configurations below.
Open Documentation

implementation

Update #2

libs.versions.toml

```
[plugins]
android-application = { id = "com.android.application", version.ref = "agp" }
kotlin-android = { id = "org.jetbrains.kotlin.android", version.ref = "kotlin" }
kotlin-compose = { id = "org.jetbrains.kotlin.plugin.compose", version.ref = "kotlin" }
kotlin-serialization = { id = "org.jetbrains.kotlin.plugin.serialization", version.ref = "kotlinxSerializationJson"}
```

build.gradle.kts (Module:app)

```
plugins {
  alias(libs.plugins.android.application)
  alias(libs.plugins.kotlin.android)
  alias(libs.plugins.kotlin.compose)
  alias(libs.plugins.kotlin.serialization)
}
```

Update Kode Pertemuan 6 (LazyColumn)

- *Update* **data class** dan fungsi untuk membaca file JSON.

```kotlin
import kotlinx.serialization.SerialName
import kotlinx.serialization.Serializable

@Serializable
data class Movie(
  @SerialName("id") var id: Int,
  @SerialName("cover") var cover: String,
  @SerialName("title") var title: String,
  @SerialName("description") var description: String
)
```

```kotlin
import android.content.Context
import com.robby.demo07kotlinserialization.entity.Movie
import kotlinx.serialization.json.Json

fun loadMovies(context: Context, fileName: String): List<Movie> {
    val jsonString = context.assets.open(fileName).bufferedReader().use { it.readText() }
    val json = Json { ignoreUnknownKeys = true }
    return json.decodeFromString(jsonString)
}
```

Update Source Code

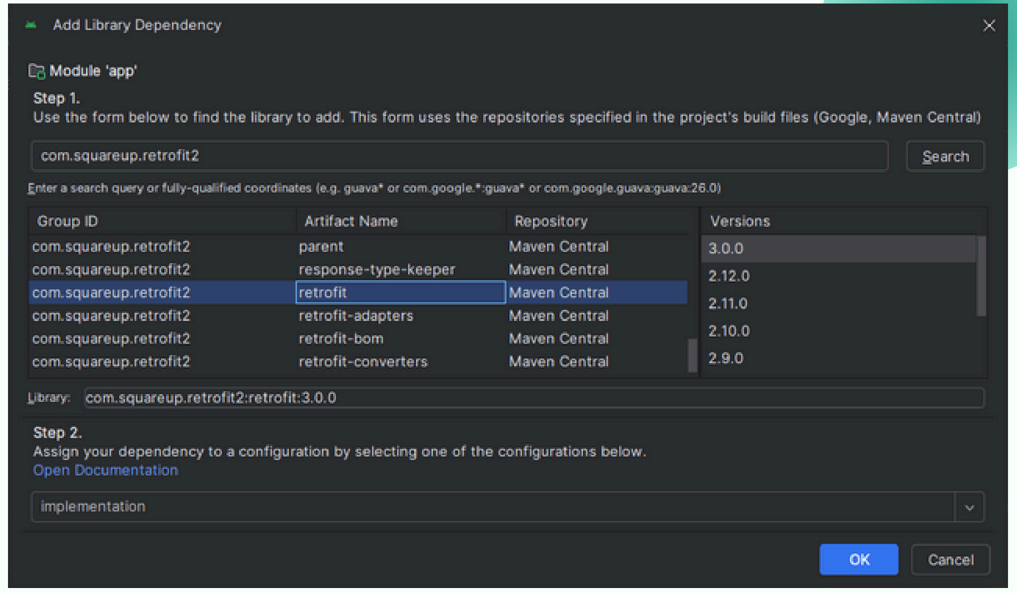
```kotlin
@Composable
fun AppNavigation() {
  val navController = rememberNavController()
  val context = LocalContext.current
  val movies = remember { loadMovies(context, "movie_data.json") }

  NavHost(navController = navController, startDestination = "home") {
    composable("home") {
      MainPage(movies, navController = navController)
    }

    composable(
      "detail/{id}",
      arguments = listOf(navArgument("id") { type = NavType.IntType })
    ) { backStackEntry ->
      val movieId = backStackEntry.arguments?.getInt("id")
      val movie = movies.find { it.id == movieId }
      DetailPage(movie, onBack = { navController.popBackStack() })
    }
  }
}
```

```kotlin
class MainActivity : ComponentActivity() {
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    enableEdgeToEdge()
    setContent {
      Demo07KotlinSerializationTheme {
        AppNavigation()
      }
    }
  }
}

@Preview(showBackground = true)
@Composable
fun KotlinSerializationDemo() {
  Demo07KotlinSerializationTheme {
    AppNavigation()
  }
}
```

Retrofit

Add Dependencies

1. org.jetbrains.kotlinx ➜ Artifact name: kotlinx-serialization-json
2. com.squareup.retrofit2 ➜ Artifact name: retrofit
3. com.squareup.okhttp3 ➜ Artifact name: logging-interceptor
4. com.jakewharton.retrofit ➜ Artifact name: retrofit2-kotlinx-serialization-converter



Module 'app'

Step 1.
Use the form below to find the library to add. This form uses the repositories specified in the project's build files (Google, Maven Central)

com.squareup.retrofit2

Enter a search query or fully-qualified coordinates (e.g. guava* or com.google.*:guava* or com.google.guava:guava:26.0)

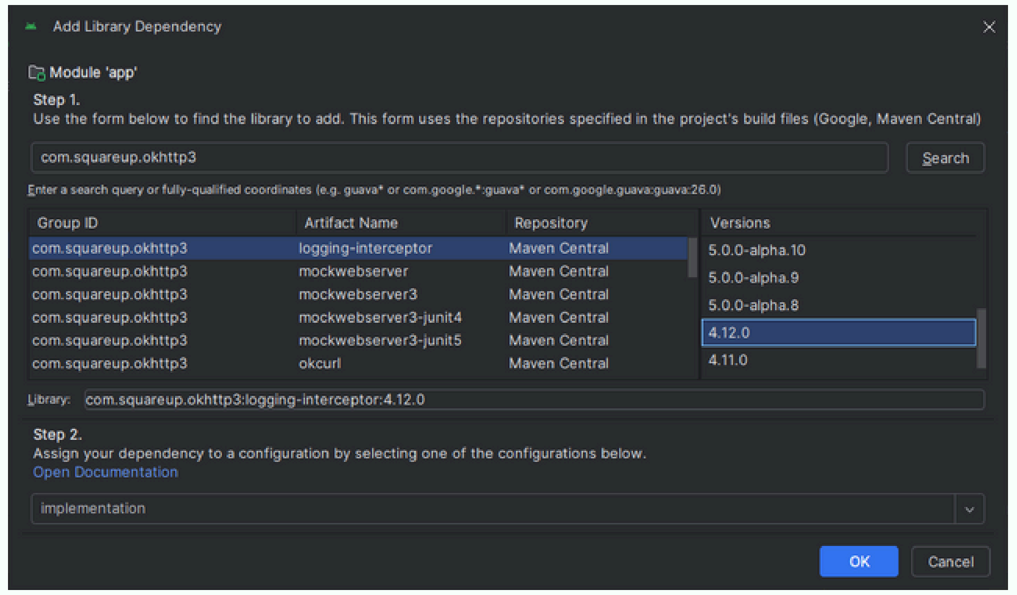
Group ID	Artifact Name	Repository
com.squareup.retrofit2	parent	Maven Central
com.squareup.retrofit2	response-type-keeper	Maven Central
com.squareup.retrofit2	retrofit	Maven Central
com.squareup.retrofit2	retrofit-adapters	Maven Central
com.squareup.retrofit2	retrofit-bom	Maven Central
com.squareup.retrofit2	retrofit-converters	Maven Central

Versions
3.0.0
2.12.0
2.11.0
2.10.0
2.9.0

Library: com.squareup.retrofit2:retrofit:3.0.0

Step 2.
Assign your dependency to a configuration by selecting one of the configurations below.
Open Documentation

implementation



Module 'app'

Step 1.
Use the form below to find the library to add. This form uses the repositories specified in the project's build files (Google, Maven Central)

com.squareup.okhttp3

Enter a search query or fully-qualified coordinates (e.g. guava* or com.google.*:guava* or com.google.guava:guava:26.0)

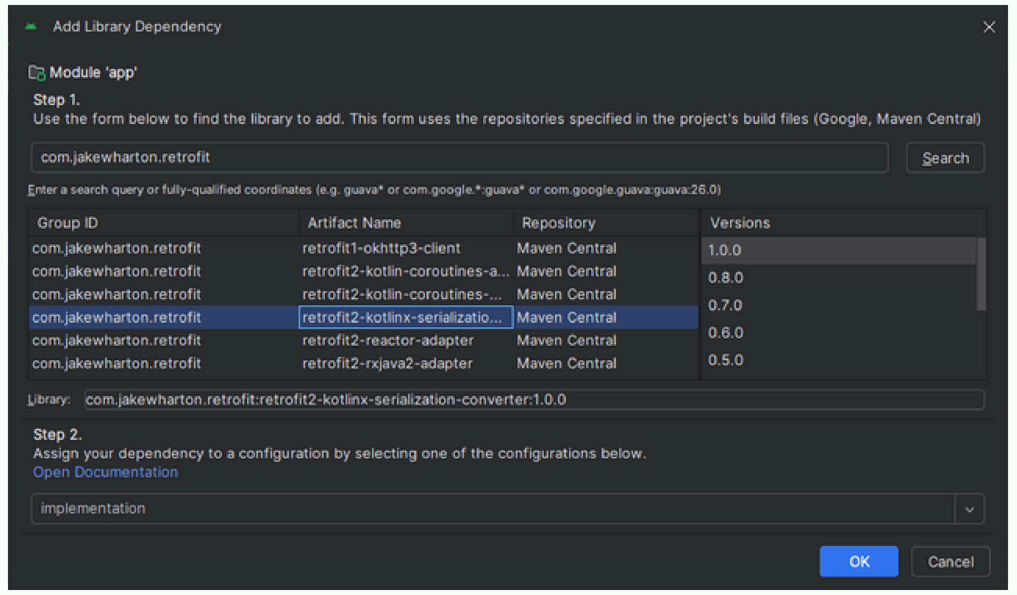
Group ID	Artifact Name	Repository
com.squareup.okhttp3	logging-interceptor	Maven Central
com.squareup.okhttp3	mockwebserver	Maven Central
com.squareup.okhttp3	mockwebserver3	Maven Central
com.squareup.okhttp3	mockwebserver3-junit4	Maven Central
com.squareup.okhttp3	mockwebserver3-junit5	Maven Central
com.squareup.okhttp3	okcurl	Maven Central

Versions
5.0.0-alpha.10
5.0.0-alpha.9
5.0.0-alpha.8
4.12.0
4.11.0

Library: com.squareup.okhttp3:logging-interceptor:4.12.0

Step 2.
Assign your dependency to a configuration by selecting one of the configurations below.
Open Documentation

implementation



Module 'app'

Step 1.
Use the form below to find the library to add. This form uses the repositories specified in the project's build files (Google, Maven Central)

com.jakewharton.retrofit

Enter a search query or fully-qualified coordinates (e.g. guava* or com.google.*:guava* or com.google.guava:guava:26.0)

Group ID	Artifact Name	Repository
com.jakewharton.retrofit	retrofit1-okhttp3-client	Maven Central
com.jakewharton.retrofit	retrofit2-kotlin-coroutines-a...	Maven Central
com.jakewharton.retrofit	retrofit2-kotlin-coroutines-...	Maven Central
com.jakewharton.retrofit	retrofit2-kotlinx-serializatio...	Maven Central
com.jakewharton.retrofit	retrofit2-reactor-adapter	Maven Central
com.jakewharton.retrofit	retrofit2-rxjava2-adapter	Maven Central

Versions
1.0.0
0.8.0
0.7.0
0.6.0
0.5.0

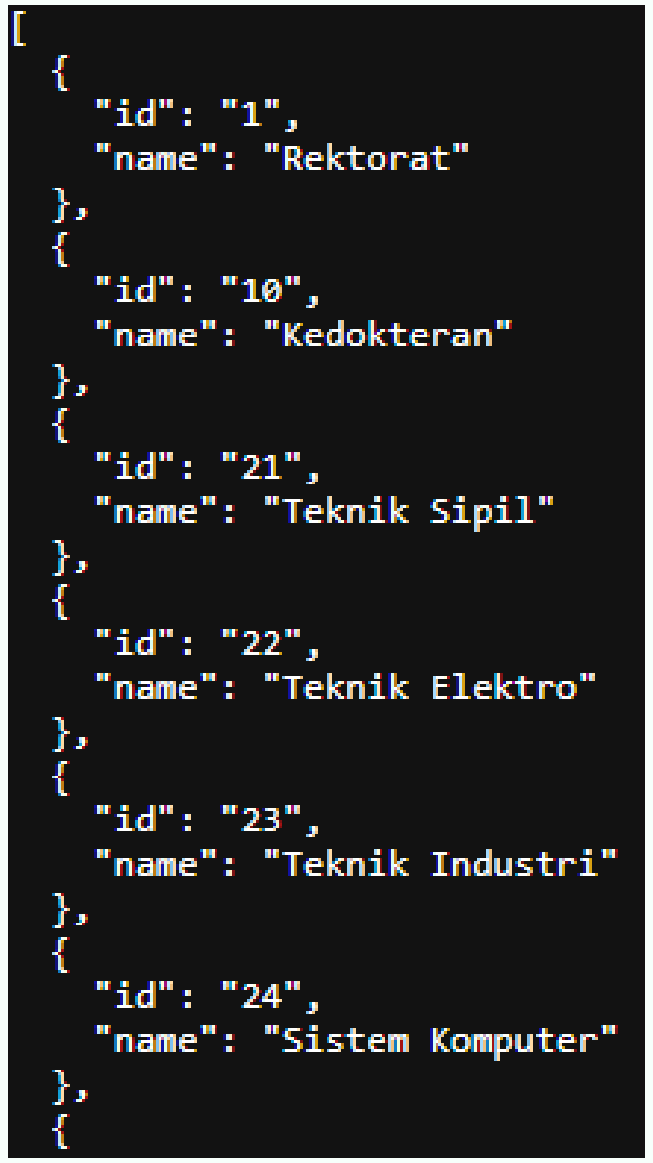
Library: com.jakewharton.retrofit:retrofit2-kotlinx-serialization-converter:1.0.0

Step 2.
Assign your dependency to a configuration by selecting one of the configurations below.
Open Documentation

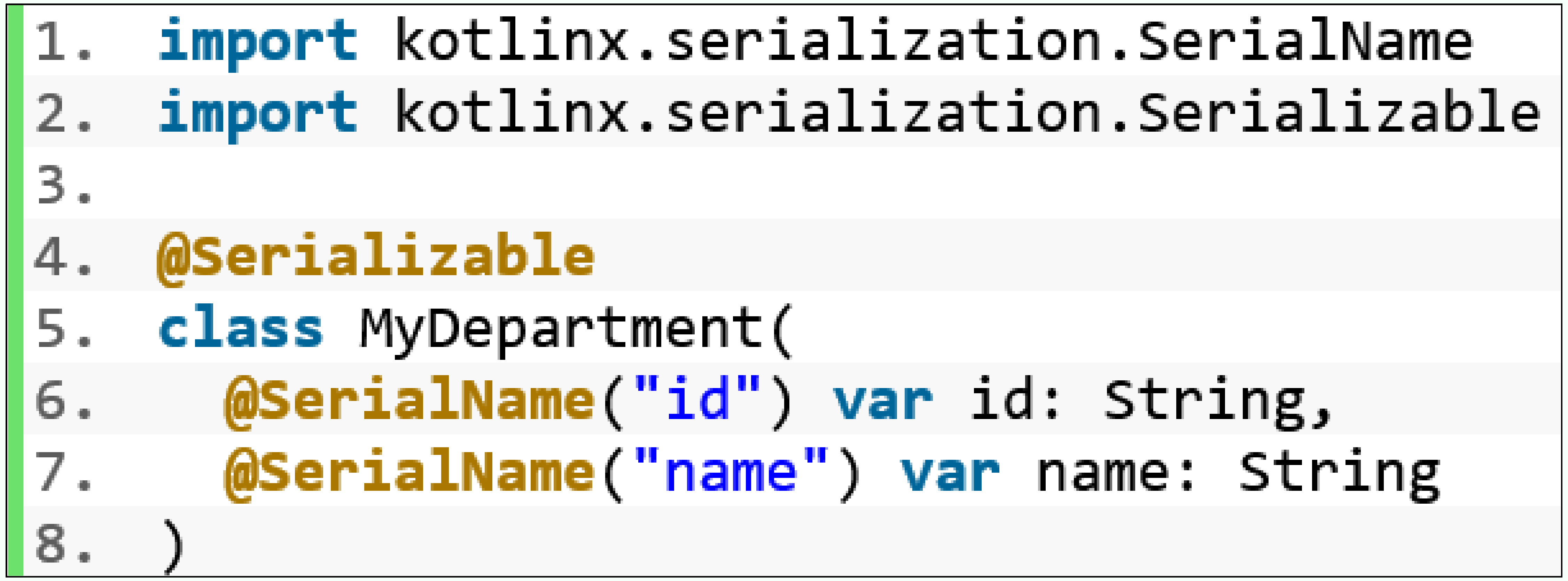
implementation

Entity Class (Data Model)

1.



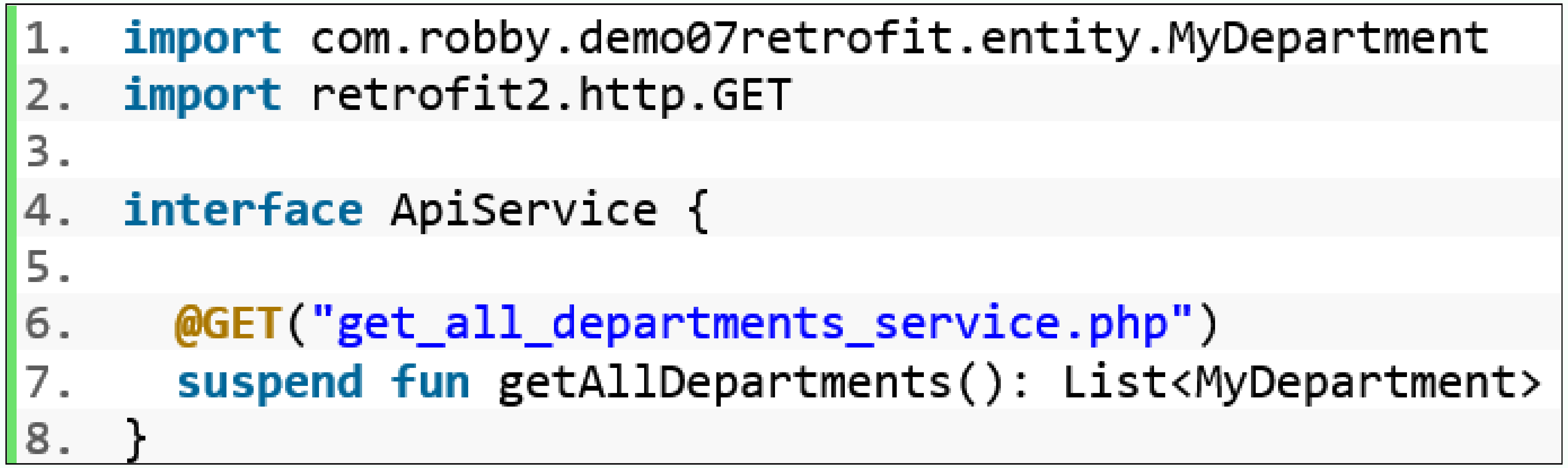
```
[
  {
    "id": "1",
    "name": "Rektorat"
  },
  {
    "id": "10",
    "name": "Kedokteran"
  },
  {
    "id": "21",
    "name": "Teknik Sipil"
  },
  {
    "id": "22",
    "name": "Teknik Elektro"
  },
  {
    "id": "23",
    "name": "Teknik Industri"
  },
  {
    "id": "24",
    "name": "Sistem Komputer"
  },
  {
```



```kotlin
import kotlinx.serialization.SerialName
import kotlinx.serialization.Serializable

@Serializable
class MyDepartment(
  @SerialName("id") var id: String,
  @SerialName("name") var name: String
)
```

API Service

Main URL: http://fittest.itmaranatha.org/me_mobile20172/service/
Service:
- get_all_departments_service.php (**GET**)

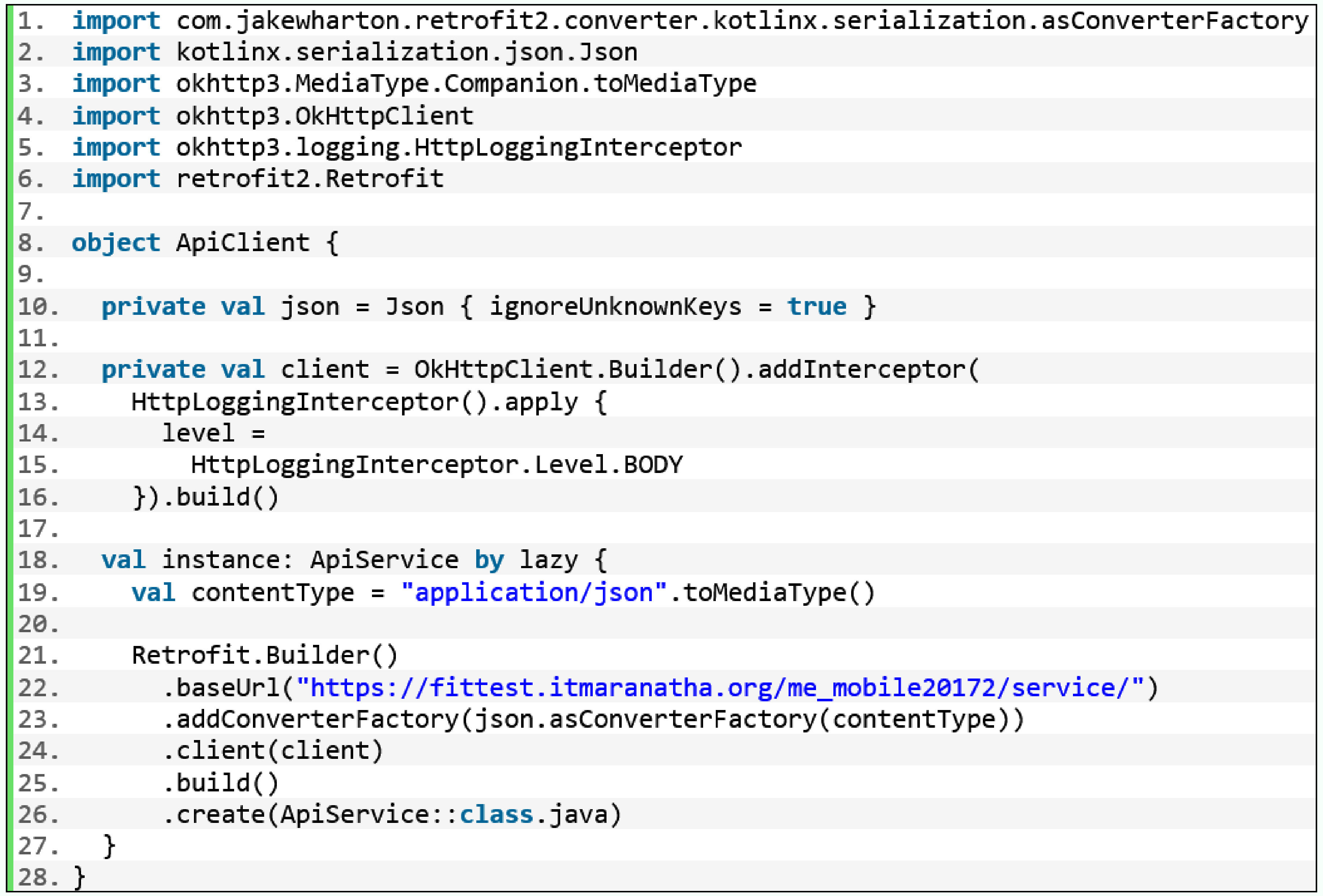


```kotlin
import com.robby.demo07retrofit.entity.MyDepartment
import retrofit2.http.GET

interface ApiService {

  @GET("get_all_departments_service.php")
  suspend fun getAllDepartments(): List<MyDepartment>
}
```

Retrofit API Client

Main URL: http://fittest.itmaranatha.org/me_mobile20172/service/
Service:

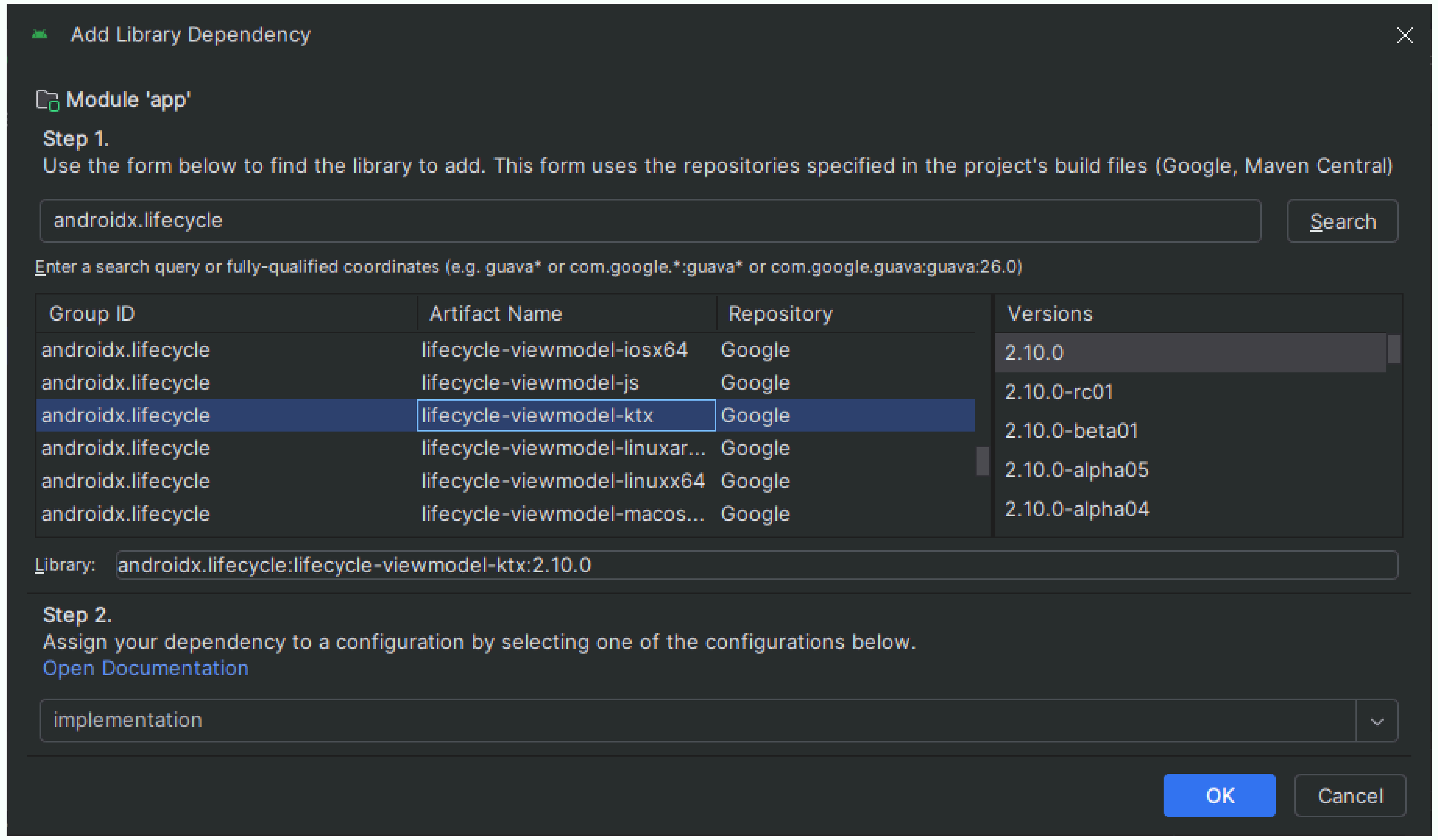
- get_all_departments_service.php (**GET**)



```kotlin
import com.jakewharton.retrofit2.converter.kotlinx.serialization.asConverterFactory
import kotlinx.serialization.json.Json
import okhttp3.MediaType.Companion.toMediaType
import okhttp3.OkHttpClient
import okhttp3.logging.HttpLoggingInterceptor
import retrofit2.Retrofit

object ApiClient {

  private val json = Json { ignoreUnknownKeys = true }

  private val client = OkHttpClient.Builder().addInterceptor(
    HttpLoggingInterceptor().apply {
      level =
        HttpLoggingInterceptor.Level.BODY
    }).build()

  val instance: ApiService by lazy {
    val contentType = "application/json".toMediaType()

    Retrofit.Builder()
      .baseUrl("https://fittest.itmaranatha.org/me_mobile20172/service/")
      .addConverterFactory(json.asConverterFactory(contentType))
      .client(client)
      .build()
      .create(ApiService::class.java)
  }
}
```

ViewModel Concept

- *Layer* yang memisahkan antara data dengan Composable (UI)
  - ViewModel memiliki *life cycle* sendiri yang terpisah dengan UI.
  - Jika data diletakkan pada Composable, data dapat diload berulang karena konsep *composable* (*recomposition*).
- ViewModel akan menghandle *state*.
- Compose akan mengamati (*observe*) *state* kemudian mengubah UI sesuai *state* tersebut.
- Penambahan *dependencies* **androidx.lifecycle** dengan *artifact name* **lifecycle-viewmodel-ktx** dan **lifecycle-viewmodel-compose**.


Add Library Dependency

Module 'app'

Step 1.
Use the form below to find the library to add. This form uses the repositories specified in the project's build files (Google, Maven Central)

androidx.lifecycle

Search

Enter a search query or fully-qualified coordinates (e.g. guava* or com.google.*:guava* or com.google.guava:guava:26.0)

Group ID	Artifact Name	Repository
androidx.lifecycle	lifecycle-viewmodel-iosx64	Google
androidx.lifecycle	lifecycle-viewmodel-js	Google
androidx.lifecycle	lifecycle-viewmodel-ktx	Google
androidx.lifecycle	lifecycle-viewmodel-linuxar...	Google
androidx.lifecycle	lifecycle-viewmodel-linuxx64	Google
androidx.lifecycle	lifecycle-viewmodel-macos...	Google

Versions
2.10.0
2.10.0-rc01
2.10.0-beta01
2.10.0-alpha05
2.10.0-alpha04

Library: androidx.lifecycle:lifecycle-viewmodel-ktx:2.10.0

Step 2.
Assign your dependency to a configuration by selecting one of the configurations below.
Open Documentation

implementation

ViewModel Code

- *Layer* yang memisahkan antara data dengan Composable (UI)
  - ViewModel memiliki *life cycle* sendiri yang terpisah dengan UI.
  - Jika data diletakkan pada Composable, data dapat diload berulang karena konsep *composable* (*recomposition*).
- ViewModel akan menghandle *state*.
- Compose akan mengamati (*observe*) *state* kemudian mengubah UI sesuai *state* tersebut.
- Penambahan *dependencies* **androidx.lifecycle** dengan *artifact name* **lifecycle-viewmodel-ktx**.

```kotlin
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.robby.demo07retrofit.entity.MyDepartment
import com.robby.demo07retrofit.services.ApiClient
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.launch

class DepartmentViewModel: ViewModel() {

  private val _departments = MutableStateFlow<List<MyDepartment>>(emptyList())
  val departments = _departments.asStateFlow()

  init {
    loadDepartments()
  }

  private fun loadDepartments() {
    viewModelScope.launch {
        val response = ApiClient.instance.getAllDepartments()
        _departments.value = response
    }
  }
```

Composable Page and MainActivity

```kotlin
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.collectAsState
import androidx.compose.runtime.getValue
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import androidx.lifecycle.viewmodel.compose.viewModel

@Composable
fun DepartmentPage(departmentViewModel: DepartmentViewModel = viewModel()) {
  val departments by departmentViewModel.departments.collectAsState()

  LazyColumn {
    items(departments) { department ->
      Text(
        text = "${department.id} ${department.name}",
        modifier = Modifier.padding(16.dp)
      )
    }
  }
}
```

```kotlin
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Scaffold
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.robby.demo07retrofit.ui.department.DepartmentPage
import com.robby.demo07retrofit.ui.theme.Demo07RetrofitTheme

class MainActivity : ComponentActivity() {
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    enableEdgeToEdge()
    setContent {
      DepartmentPage()
    }
  }
}
```

- Tambahkan **uses permission** untuk **INTERNET** pada AndroidManifest.xml

Thank You

ROBBY.TAN@IT.MARANATHA.EDU