Local DB (Room)

MOBDEVE – Mobile Development

Room - Definition

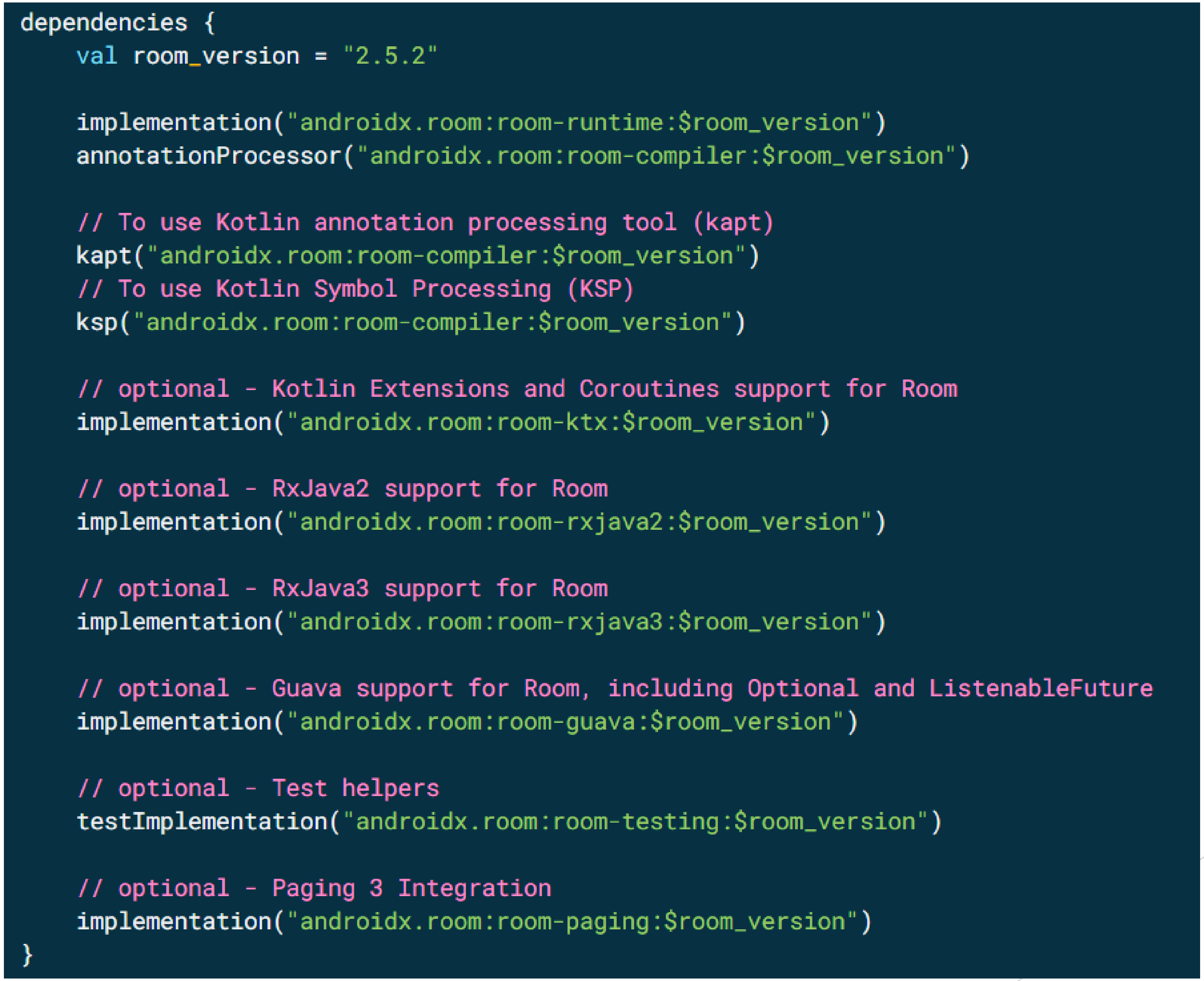
- Room is a database layer on top of an SQLite database.
- Room takes care of mundane tasks that you used to handle with an SQLiteOpenHelper.
- Room uses the DAO to issue queries to its database.
- By default, to avoid poor UI performance, Room doesn't allow you to issue queries on the main thread. When Room queries return Flow, the queries are automatically run asynchronously on a background thread.
- Room provides compile-time checks of SQLite statements.

Room - Advantage

- The Room persistence library provides an abstraction layer over SQLite to allow fluent database access while harnessing the full power of SQLite. In particular, Room provides the following benefits:
- Compile-time verification of SQL queries.
- Convenience annotations that minimize repetitive and error-prone boilerplate code.
- Streamlined database migration paths.

Room – as compared with SQLite

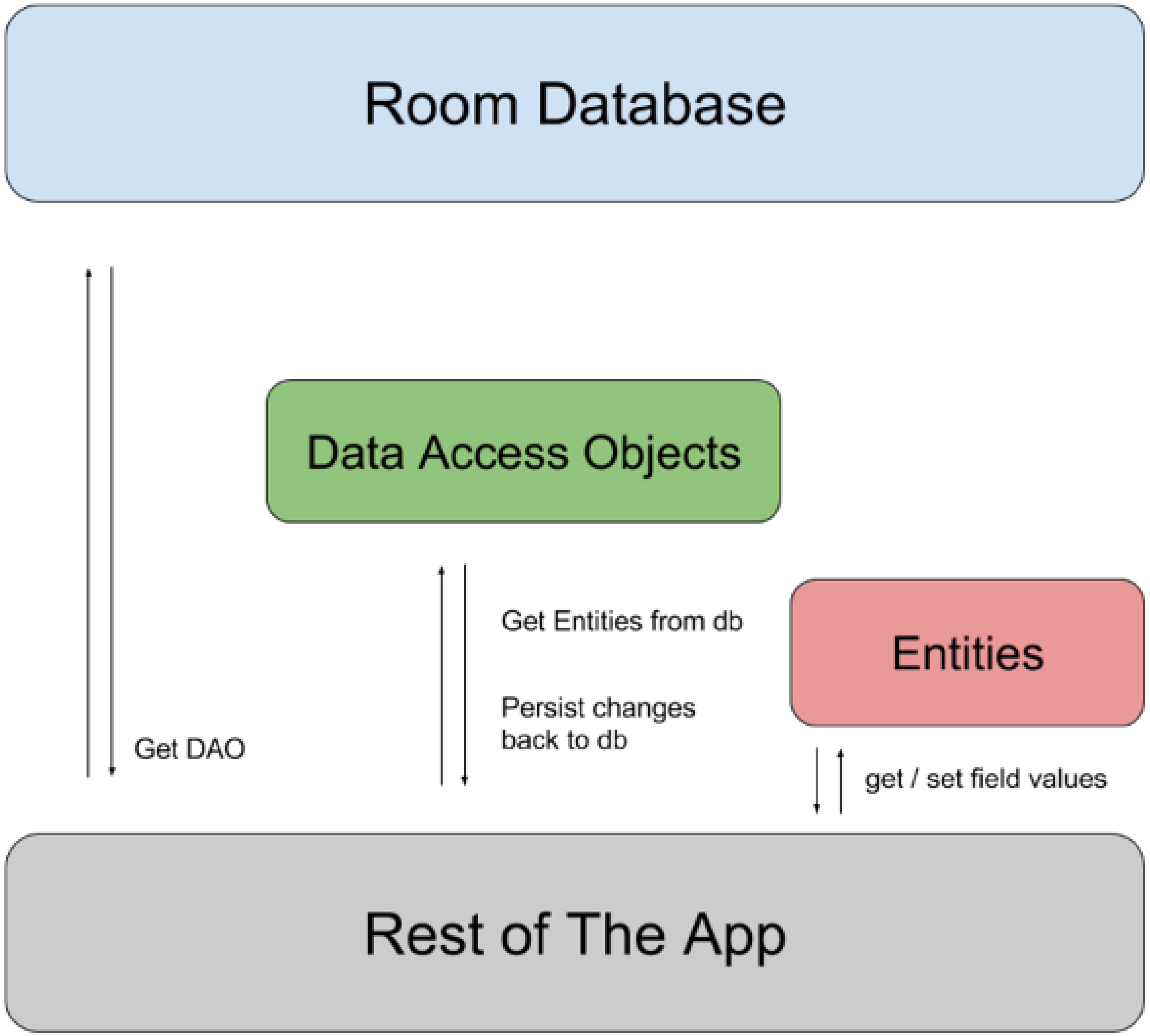
- In case of SQLite, There is no compile time verification of raw SQLite queries. But in Room there is SQL validation at compile time.
- As your schema changes, you need to update the affected SQL queries manually. Room solves this problem.
- You need to use lots of boilerplate code to convert between SQL queries and Java data objects. But, Room maps our database objects to Java Object without boilerplate code.
- Room is built to work with LiveData and RxJava for data observation, while SQLite does not.



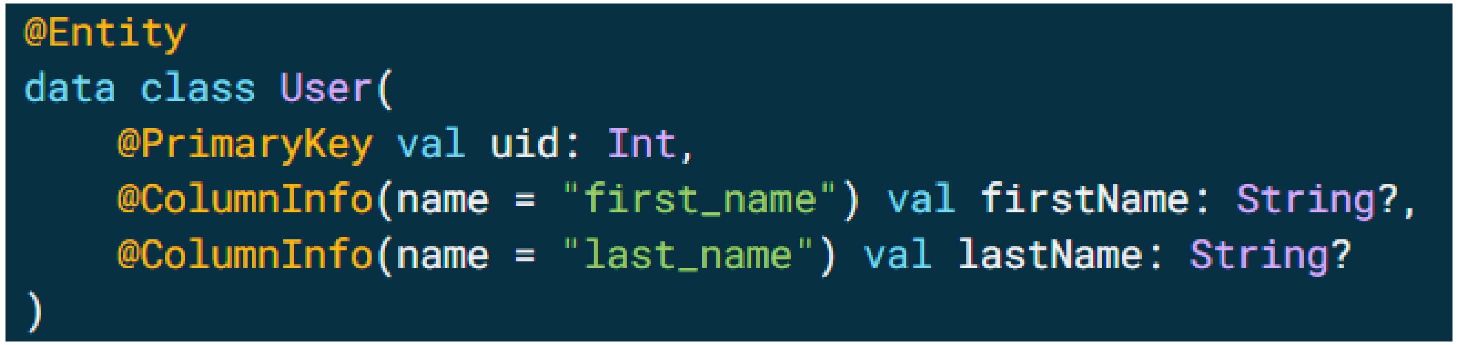
```
dependencies {
    val room_version = "2.5.2"

    implementation("androidx.room:room-runtime:$room_version")
    annotationProcessor("androidx.room:room-compiler:$room_version")

    // To use Kotlin annotation processing tool (kapt)
    kapt("androidx.room:room-compiler:$room_version")
    // To use Kotlin Symbol Processing (KSP)
    ksp("androidx.room:room-compiler:$room_version")

    // optional - Kotlin Extensions and Coroutines support for Room
    implementation("androidx.room:room-ktx:$room_version")

    // optional - RxJava2 support for Room
    implementation("androidx.room:room-rxjava2:$room_version")

    // optional - RxJava3 support for Room
    implementation("androidx.room:room-rxjava3:$room_version")

    // optional - Guava support for Room, including Optional and ListenableFuture
    implementation("androidx.room:room-guava:$room_version")

    // optional - Test helpers
    testImplementation("androidx.room:room-testing:$room_version")

    // optional - Paging 3 Integration
    implementation("androidx.room:room-paging:$room_version")
}
```

Room – Primary Components

- The database class that holds the database and serves as the main access point for the underlying connection to your app's persisted data.
- Data entities that represent tables in your app's database.
- Data access objects (DAOs) that provide methods that your app can use to query, update, insert, and delete data in the database.


Room Database
Data Access Objects
Get DAO
Get Entities from db
Persist changes back to db
Entities
get / set field values
Rest of The App

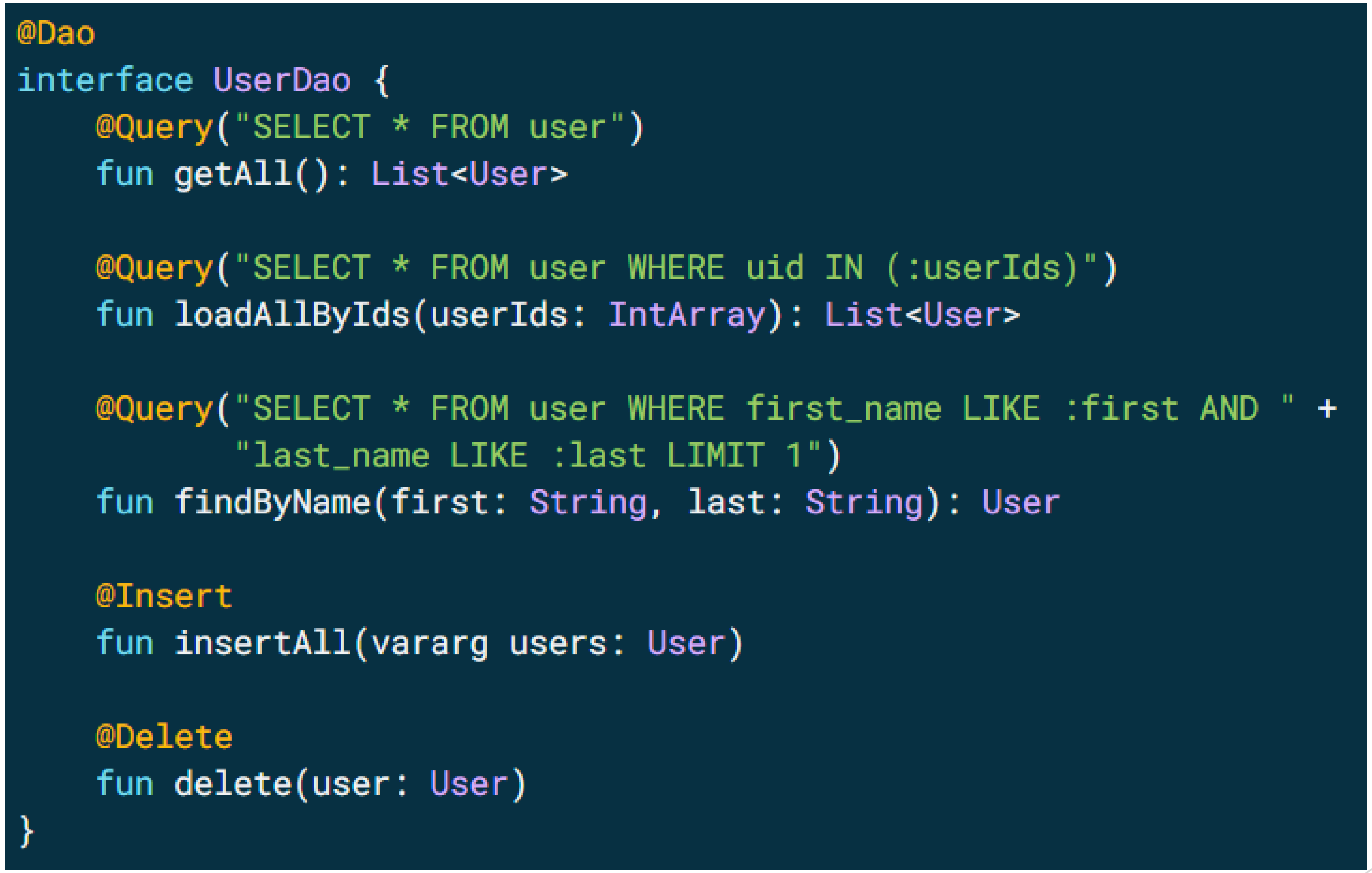
Room – Step 1: Entity



```
@Entity
data class User(
    @PrimaryKey val uid: Int,
    @ColumnInfo(name = "first_name") val firstName: String?,
    @ColumnInfo(name = "last_name") val lastName: String?
)
```

** This can also be used:*
  @PrimaryKey(autoGenerate = true)

Room – Step 2: Data Access Object



```kotlin
@Dao
interface UserDao {
    @Query("SELECT * FROM user")
    fun getAll(): List<User>

    @Query("SELECT * FROM user WHERE uid IN (:userIds)")
    fun loadAllByIds(userIds: IntArray): List<User>

    @Query("SELECT * FROM user WHERE first_name LIKE :first AND " +
           "last_name LIKE :last LIMIT 1")
    fun findByName(first: String, last: String): User

    @Insert
    fun insertAll(vararg users: User)

    @Delete
    fun delete(user: User)
}
```

Room – Step 3: Database

Declaration

```
@Database(entities = [User::class], version = 1)
abstract class AppDatabase : RoomDatabase() {
    abstract fun userDao(): UserDao
}
```

Usage

```
val db = Room.databaseBuilder(
            applicationContext,
            AppDatabase::class.java, "database-name"
        ).build()
```

```
val userDao = db.userDao()
val users: List<User> = userDao.getAll()
```

Room – Hands-on via Codelabs

- Let’s get our hands-on with Room via Codelabs: https://developer.android.com/codelabs/android-room-with-a-view-kotlin
- Submission will be treated as points (FP) in our Exercises module

End of Presentation

Thank you very much! Have a great day ahead! ☺