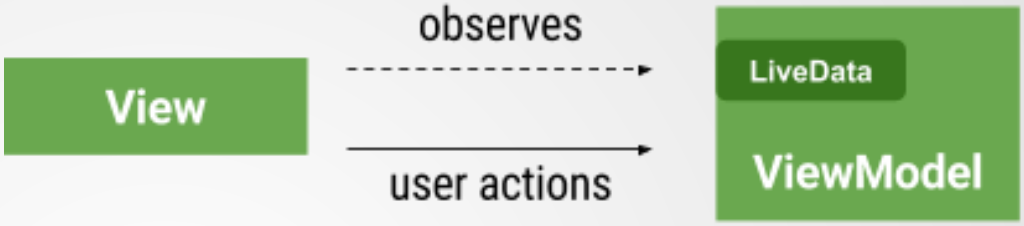
Android

LiveData

LiveData

Erklärung

Observer Pattern

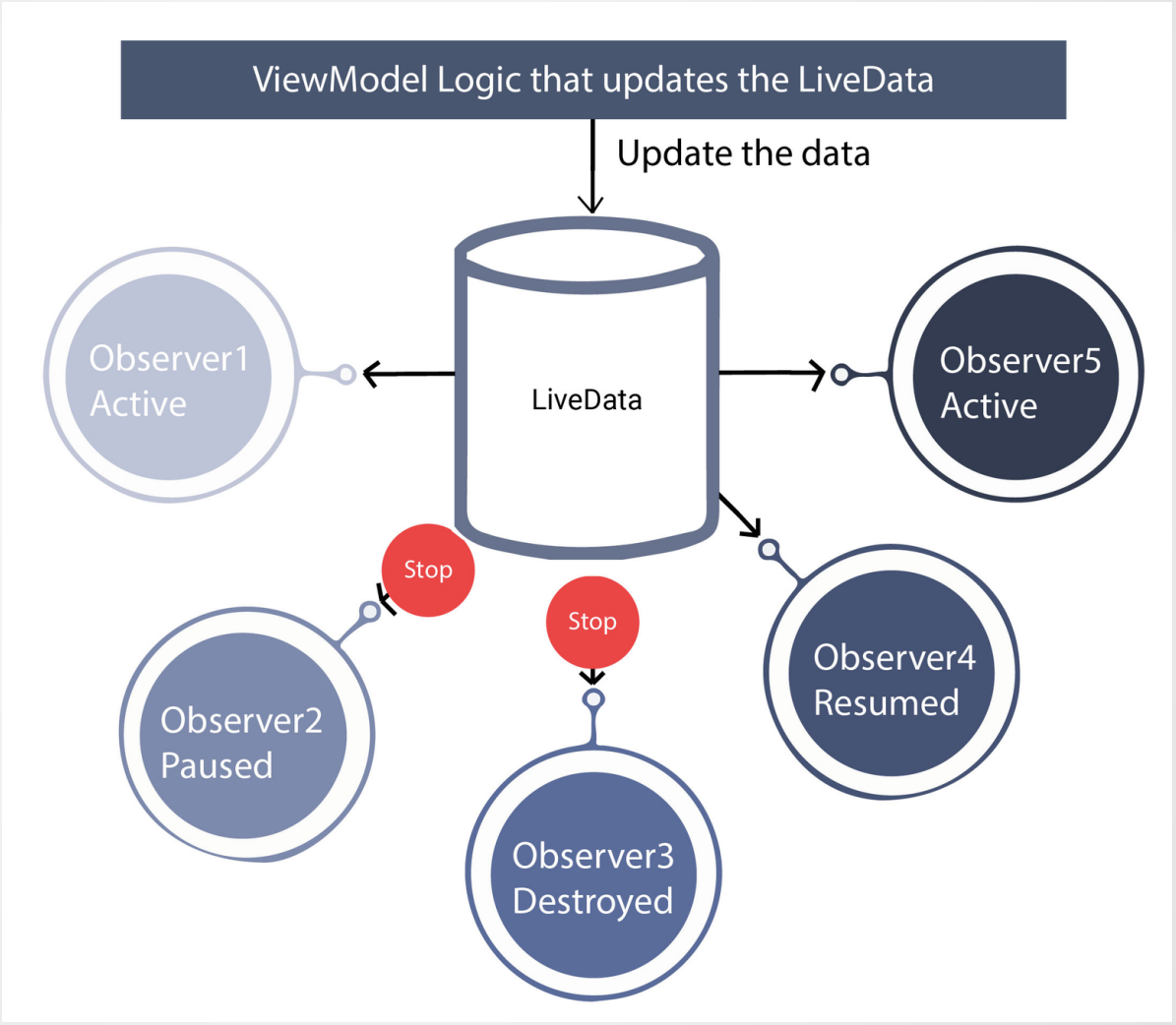


- ViewModels sollten nichts vom View wissen
- Der View trägt sich beim LiveData im ViewModel ein, damit er updates bekommt
- Der View kann Methoden im ViewModel aufrufen, falls ein Benutzer eine Aktion ausführt

Lifecycle Aware

- Schickt nur Updates an aktive Abonnenten (**STARTED** oder **RESUMED**)
- Beim Abonnieren wird ein LifeCylce-Objekt übergeben
  - Wenn dieses Objekt auf **DESTROYED** wechselt, wird das Abonnement automatisch entfernt

Lifecycle Aware



Vorteile

- UI zeigt sicher immer den Data-State an
- Keine Mem-Leaks
- Keine Crashes wegen gestoppter Activities
- Kein manuelles Lifecycle-Handling
- Immer Up-To-Date (auch nach Aufwachen, Config-Änderungen, ...)
- Resourcen teilen (singleton LiveData)

LiveData

Beispiel

Deklaration

```
1 class NameViewModel : ViewModel() {
2
3     // Create a LiveData with a String
4     val currentName: MutableLiveData<String> by lazy {
5         MutableLiveData<String>()
6     }
7
8     // Rest of the ViewModel...
9 }
```

- Wrapper für irgendwelche Datentypen
- Lebt normalerweise im ViewModel
- Wird normal über Getter abgefragt

LiveData Abonnieren

- Normalerweise in der **onCreate()** Methode in einem ViewModel
  - damit nicht unnötige Calls gemacht werden, wie z.B. in **onResume()**
  - damit das Fragment die Daten zum richtigen Zeitpunkt hat (sobald es **STARTED** ist)
- Zu diesem Zeitpunkt muss das LiveData-Objekt gesetzt sein. Ansonsten wird kein Abonnent notifiziert

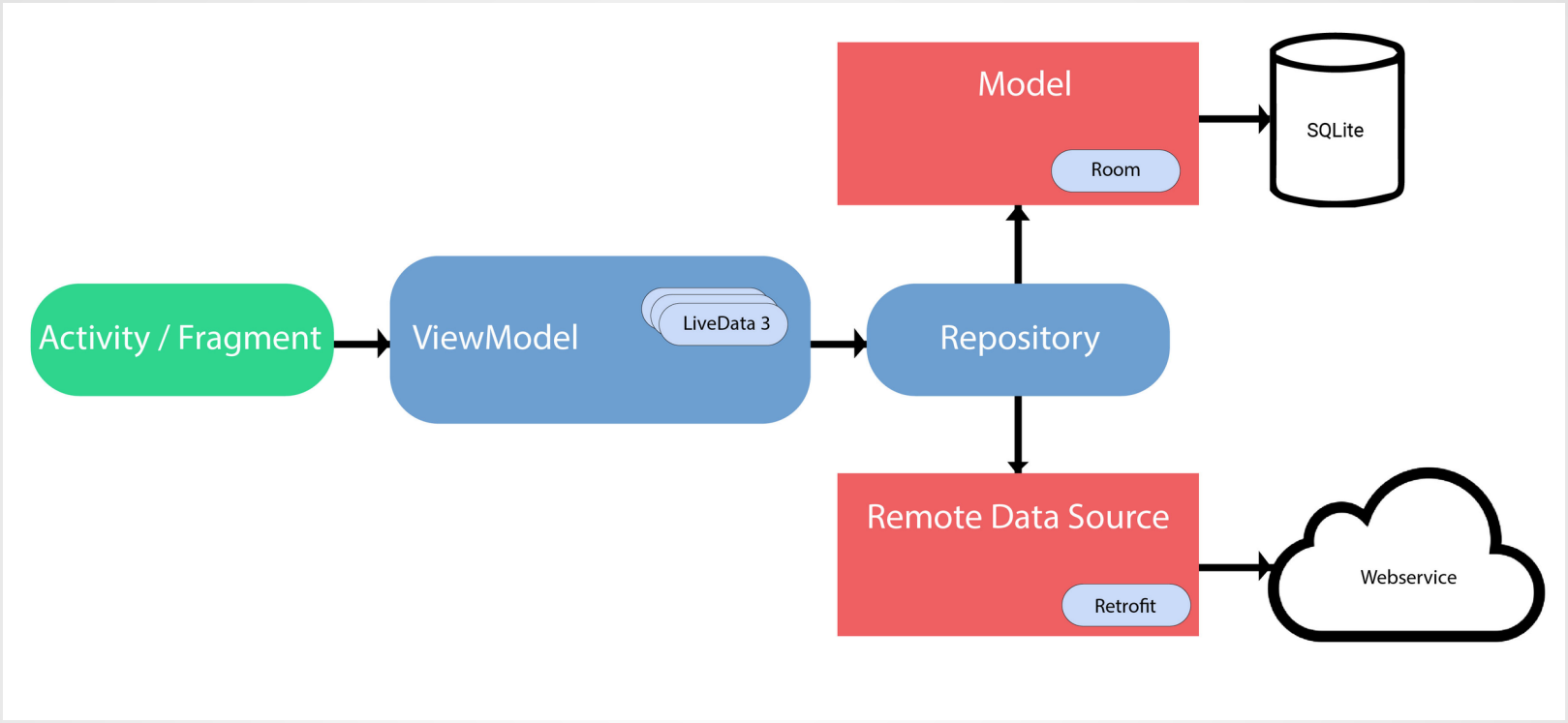
LiveData Abonnieren

```kotlin
class NameActivity : AppCompatActivity() {

    // Use the 'by viewModels()' Kotlin property delegate
    // from the activity-ktx artifact
    private val model: NameViewModel by viewModels()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // Other code to setup the activity...

        // Create the observer which updates the UI.
        val nameObserver = Observer<String> { newName ->
            // Update the UI, in this case, a TextView.
            nameTextView.text = newName
        }

        // Observe the LiveData, passing in this activity as the
        // LifecycleOwner and the observer.
        model.currentName.observe(this, nameObserver)
    }
}
```

- Updates werden geschickt, falls der Wert sich ändert und nur an aktive Abonnenten
- Außer ein Abonnent wechselt von *"inaktiv"* auf *"aktiv"*
  - Updates werden dann nur geschickt, falls sich seitdem Daten geändert hatten

LiveData

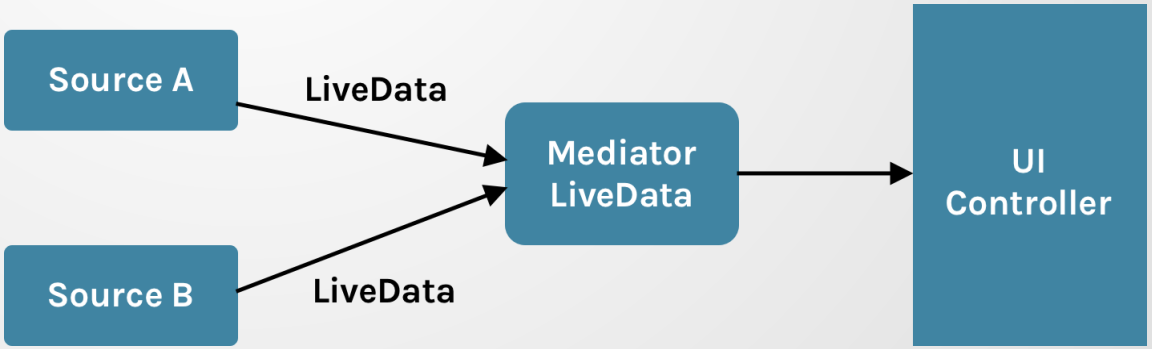
Verwendung


Model
Room
SQLite
Activity / Fragment
ViewModel
LiveData 3
Repository
Remote Data Source
Retrofit
Webservice

- **LiveData** ... immutable (read only)

- **MutableLiveData** ... mutable (getter und setter)

- **MediatorLiveData** ... horcht auf andere **LiveData**s und entscheidet, ob ein Event weiter propagiert wird oder nicht



Referenzen

- https://developer.android.com/topic/libraries/architecture/livedata#kotlin
- https://medium.com/androiddevelopers/viewmodels-and-livedata-patterns-antipatterns-21efaef74a54
- https://www.innominds.com/blog/introduction-to-livedata-in-android