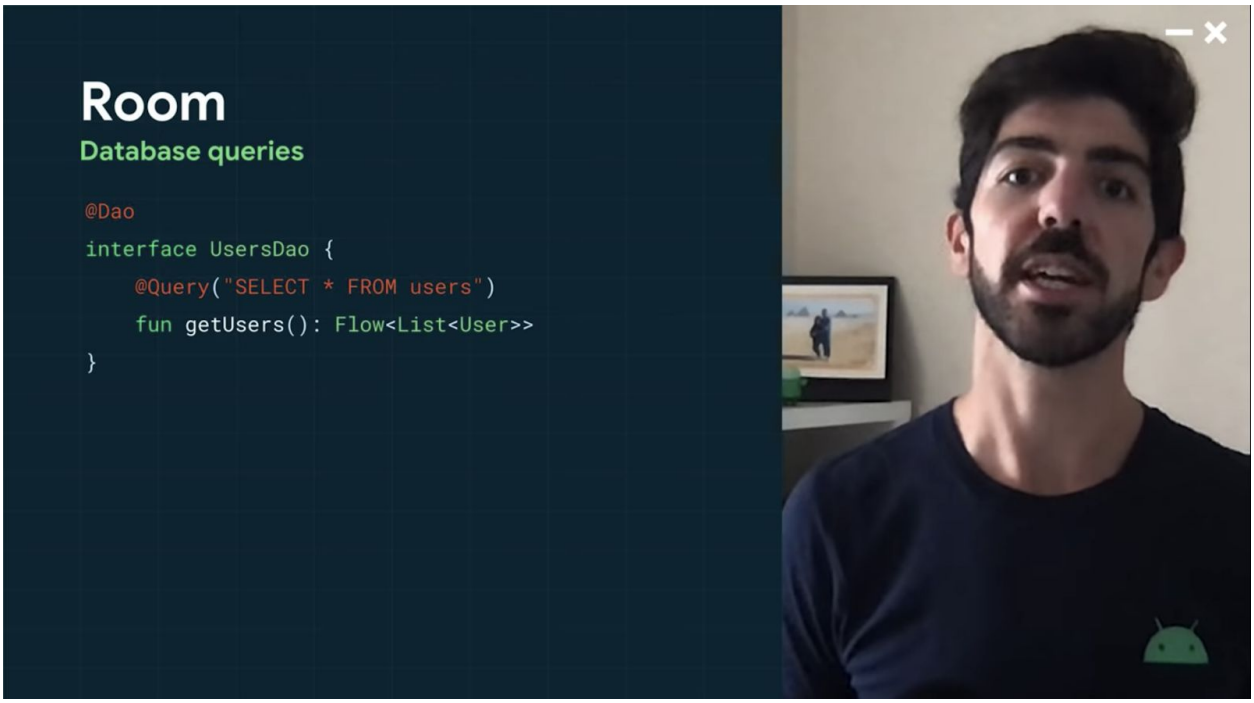
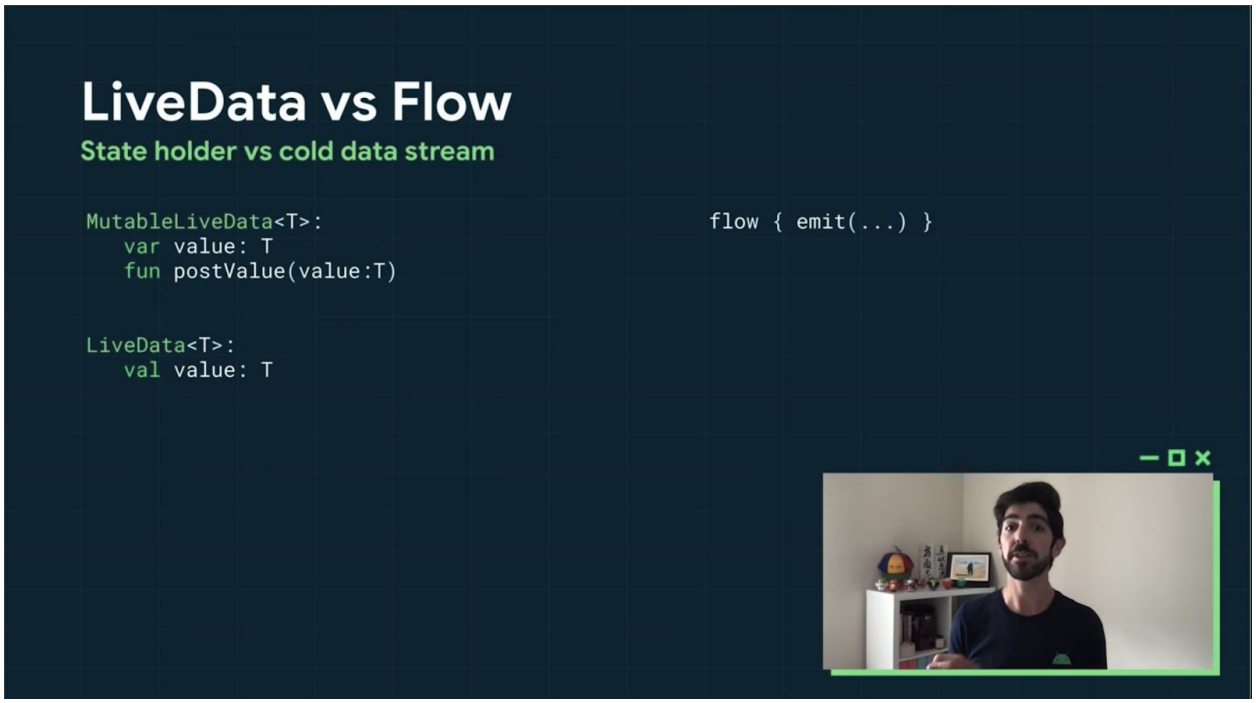
1. Kotlin Flows
    a. Model streams of the data, not the single values.
    b. Built upon the foundation of coroutines and suspend function
        i. Cancellation
        ii. Structure Concurrency
        iii. Exception Transparency
        iv. Natural back pressure handling

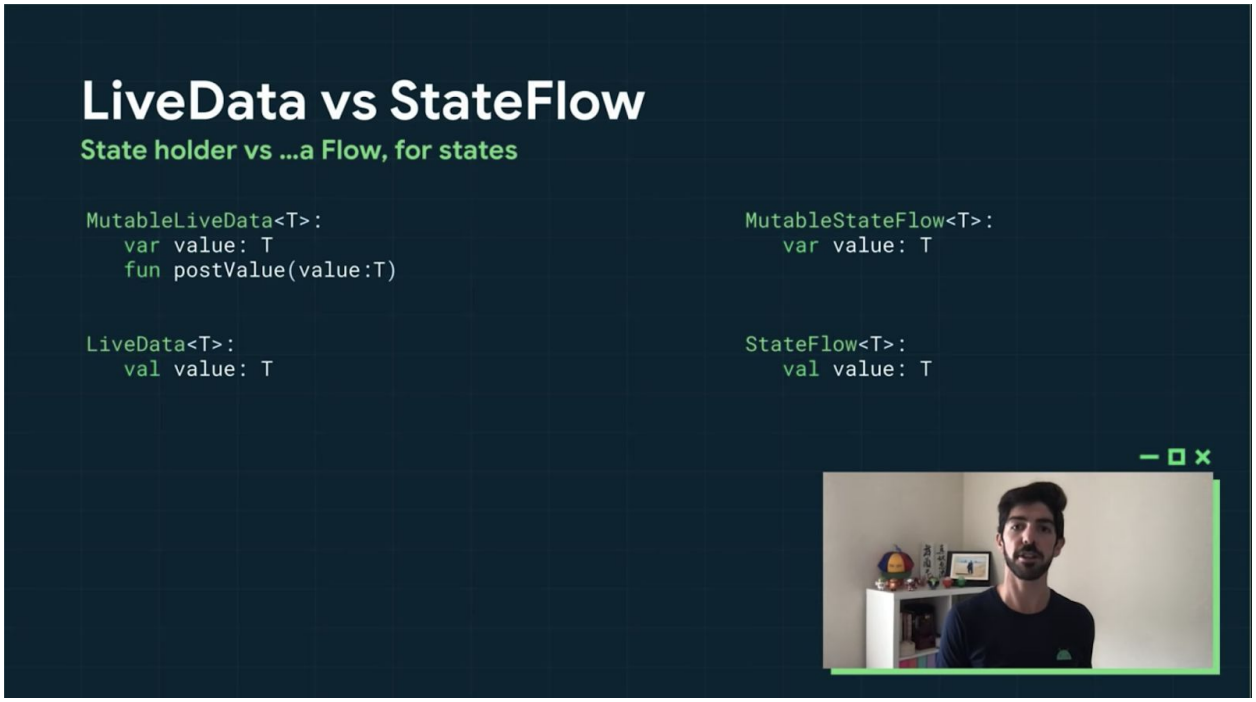



        v. Here, once the database is changed, it automatically updates the UI in a reactive way. (Observing the flow lets **the app react to changes** when the DB emits new data)

2. Comparison to the RxJava
    a. Conceptually it's not different.
3. Comparison to the Livedata
    a. Short Answer: little difference
    b. Full Answer: more complicated

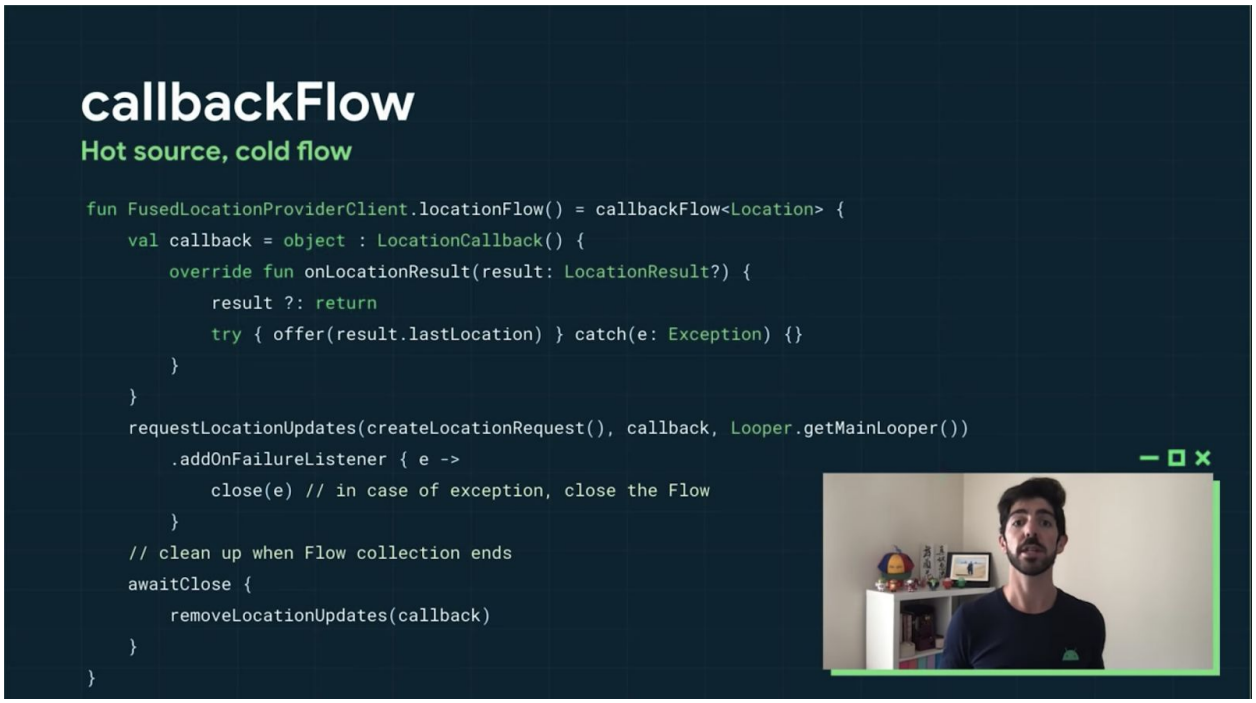



4. Deep Dive into LiveData vs Flow
    a. Conceptually LiveData is a value holder, so the most important thing is to observe the current state.
    b. As UI only sees the latest result, hence it's fine to use LiveData
    c. Flow observes the 'current' of the data, so it's more appropriate for the lower layer of the app

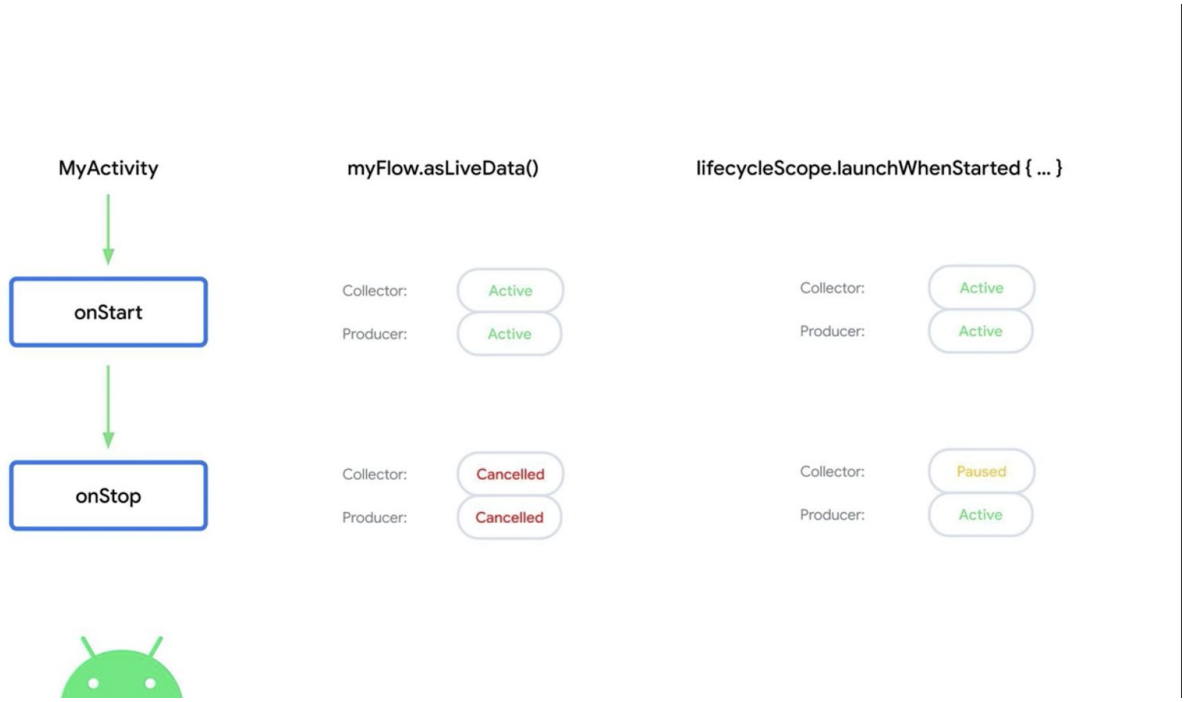


5. StateFlow
    a. Found in the latest coroutine version
    b. It holds the current value like LiveData
    c. But LiveData has some drawbacks like being able to be observed only in the main thread.
    d. Trying converting one over the other, as converting is easy from regular Flow to LiveData or vice versa.

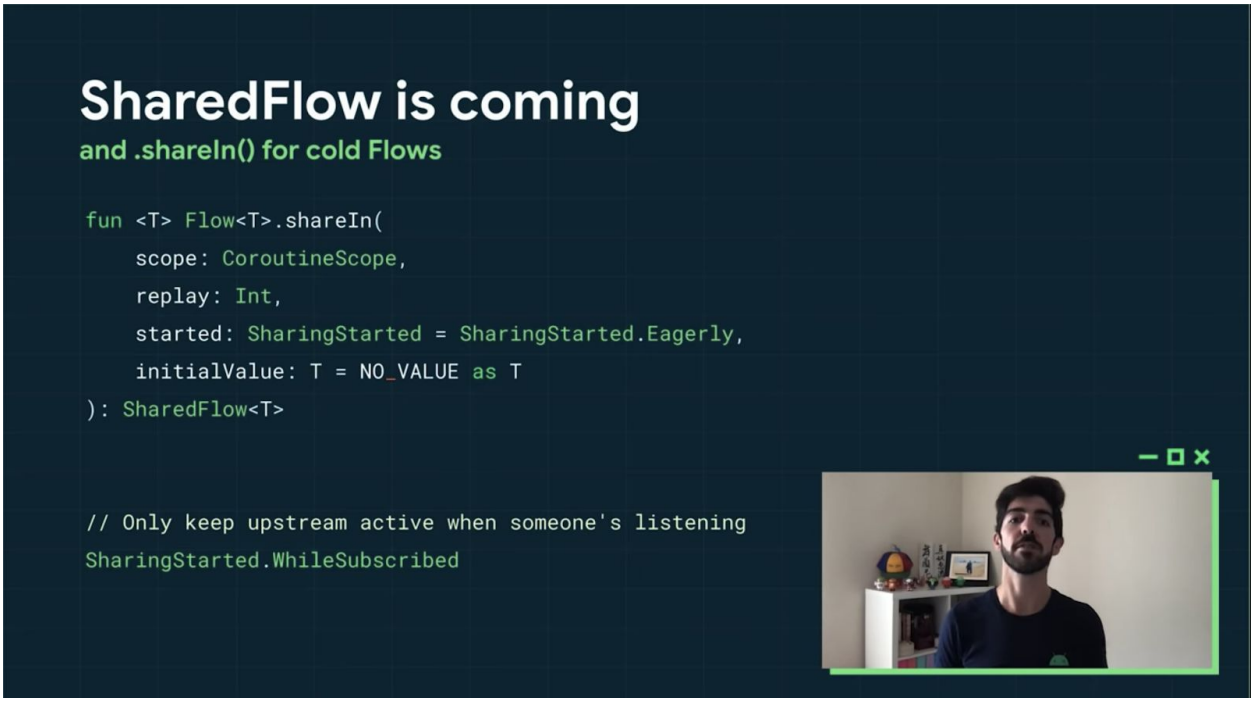

6. ViewModelScope, CoroutineScope and Android
    a. LifecycleScope for Activities and Fragments
        i. When to cancel the livedata and flow?




7. callbackFlow
    a. A builder function that lets us convert callback or listener-based API to Flow.
    b. Hot source: Can it be understood as Hot Observable (?)
    c. Cold flow: the code block will not be executed until the flow is collected.



d. So, even if the activity stays onStop, producers are still alive, which means it causes battery problems.
e. https://developer.android.com/topic/libraries/architecture/coroutines




8. SharedFlow
    a. It can take the flow, and be shared among multiple subscribers.
    b. Useful when the flow is expensive to create.
    c. WhileSubscribed option: free up the upstream producer whenever no subscribers.