droidcon academy

Migrating LiveData to Kotlin Flow with tests in Android

Maia Grotepass

@maiatoday

in/maiagrotepass

Overview

- Explore the starter project and find the LiveData to migrate
- Flow vs LiveData
- **Migrate** the DataSource
- **Migrate** the Repository and add **tests**
- **Migrate** the MainViewModel and add **tests**
- Coding **Challenge**
- **Connect** the MainViewModel to the Composable UI

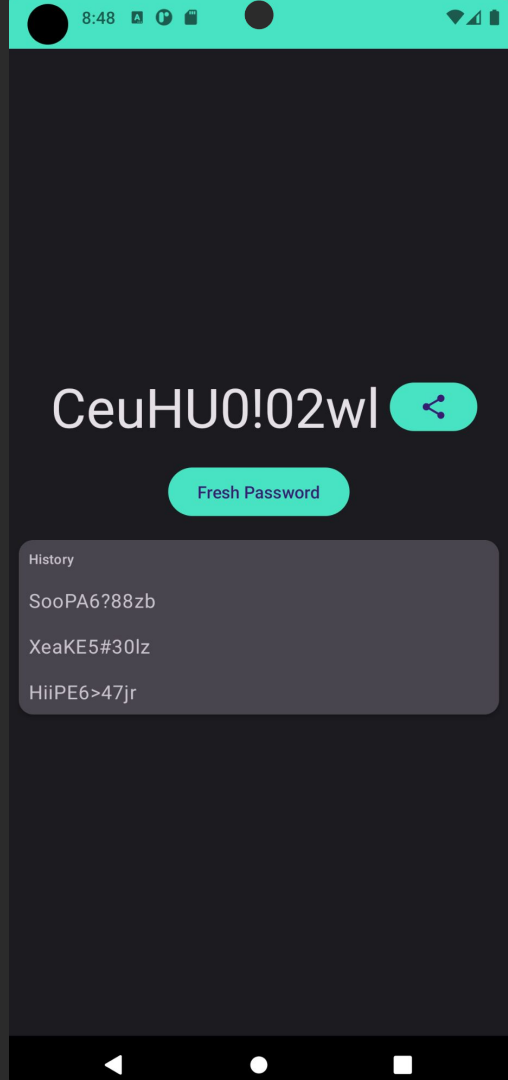
droidcon academy

Course Demo App

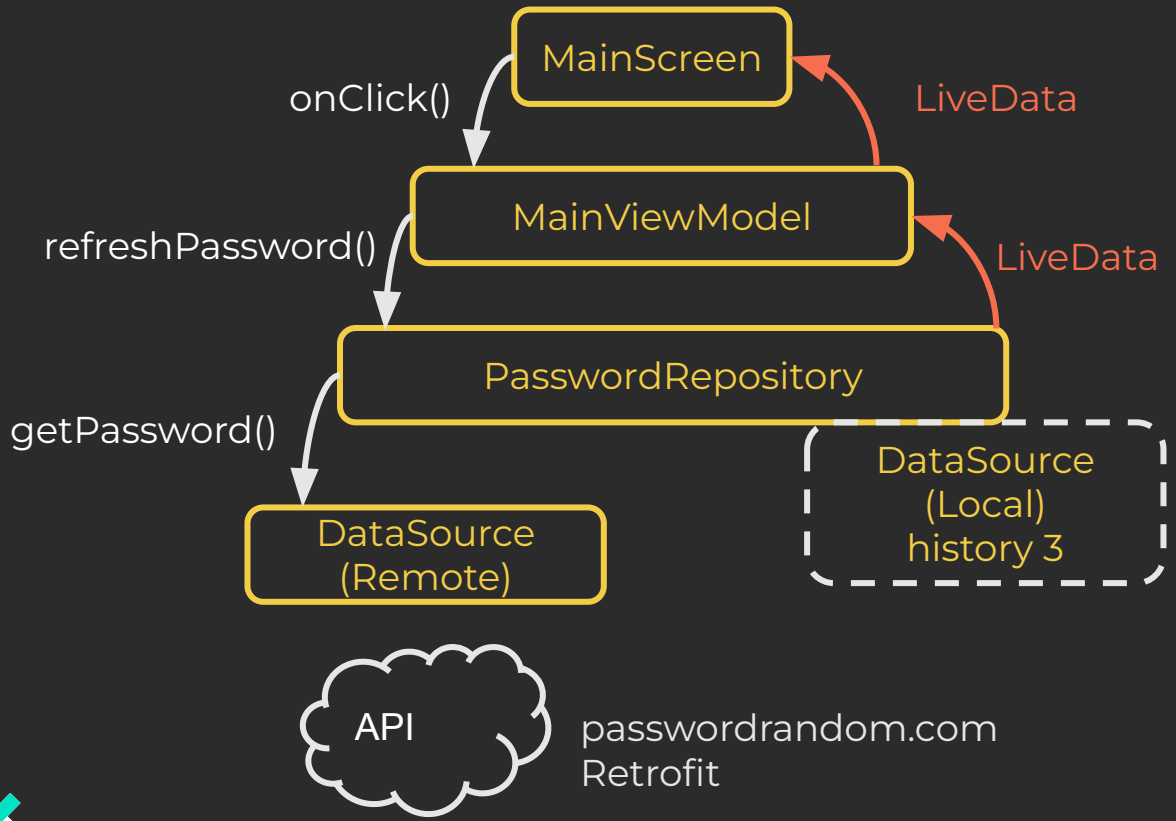
FreshPassword App: Fetch a fresh random password at the click of a button. Save a history of the previous 3 passwords.

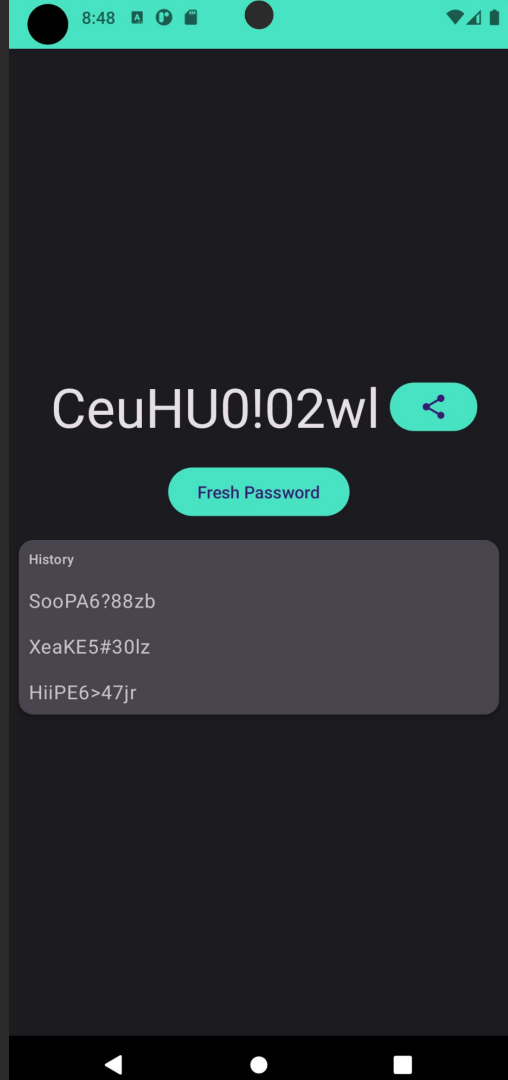
- Libraries used:
  - Jetpack Compose
  - Retrofit
  - Kotlin coroutines Flow
  - kotlin.test
  - kotlinx-coroutines-test

www.passwordrandom.com

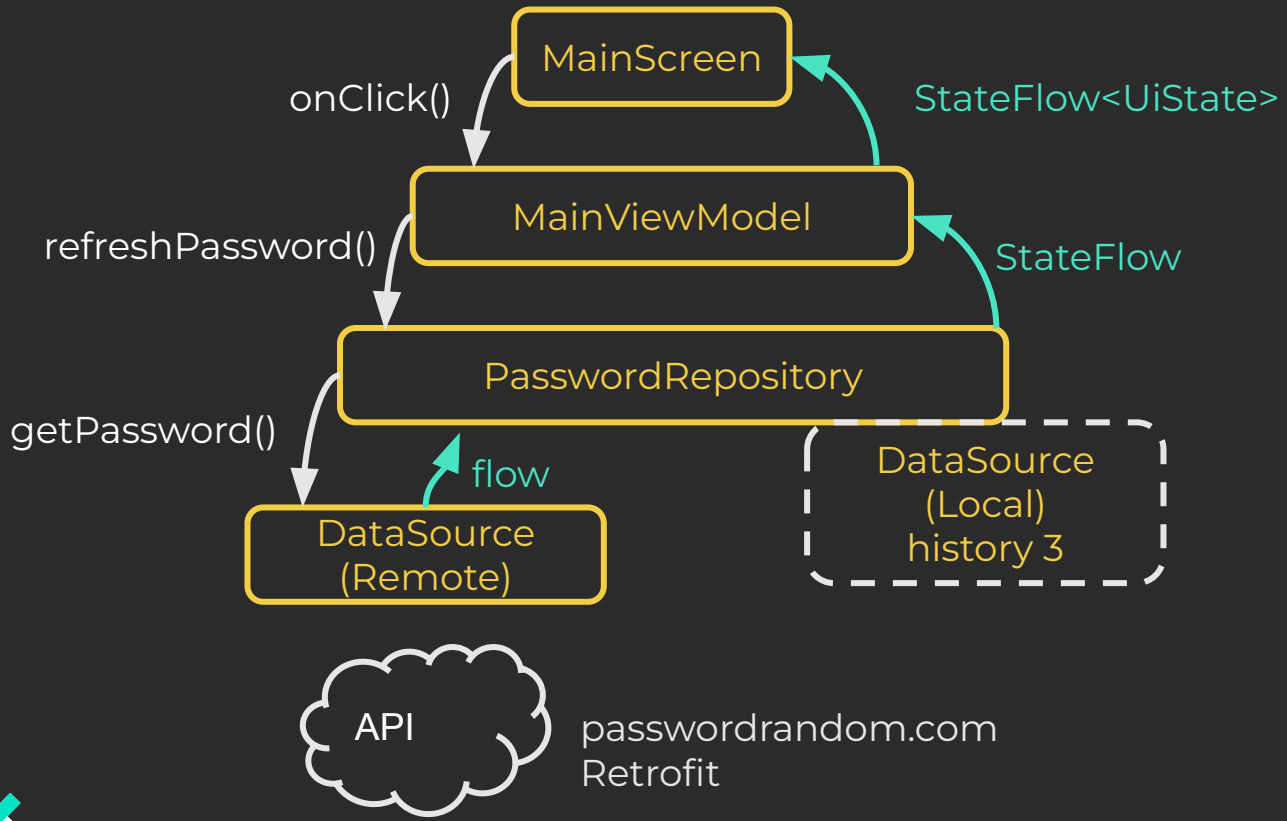


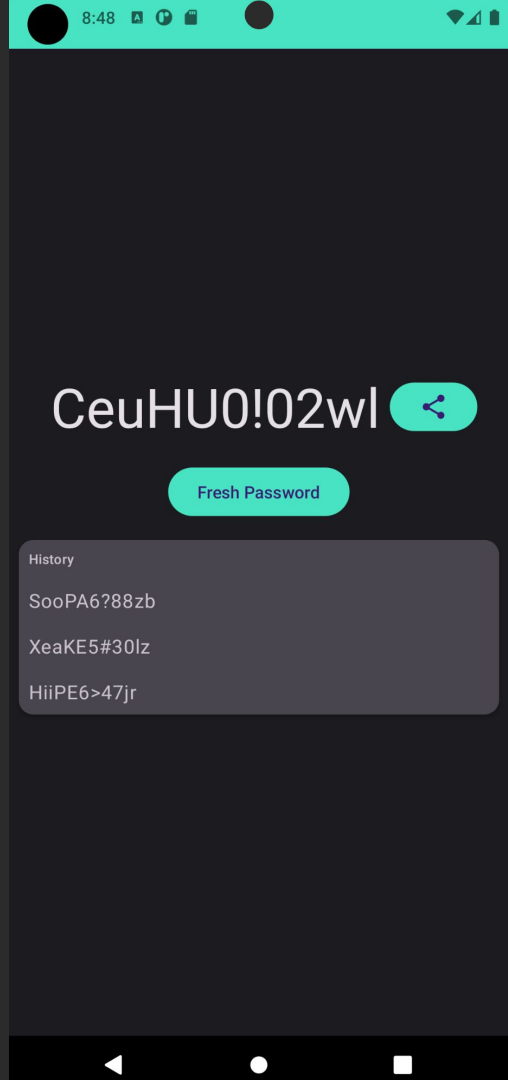
FreshPassword App





FreshPassword App





Flows vs Live Data

droidcon academy

Flow Refresher

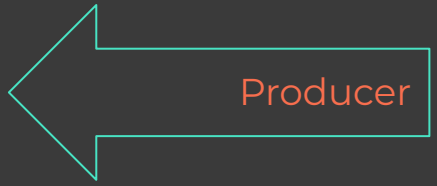
- Producer
  - Asynchronous data stream
  - Emit multiple values
- Collector
  - Collects in suspend block
- Default Cold
  - Produce code only runs -> collect
  - Completes

```
val myFlow = flow {
    // inside suspend
   emit(1)
   delay(1000)
   emit(2)
}

myFlow.collect { value ->
    // inside suspend
    doSomething(value)
}
```

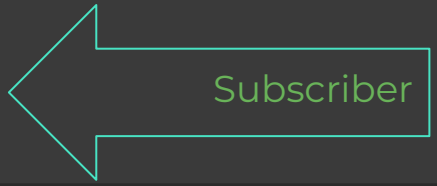
Flow completes

```
class MainViewModel: ViewModel {
    private val _password = MutableStateFlow("")
    val _password = _password.asStateFlow() // this is a StateFlow
    fun update() {
        _password.update { "Password" } // atomic, concurrent multiple subscribers ok
    }
}
```




```
// in UI need to handle lifecycle correctly: launch in the correct scope and use repeatOnLifeCycle
viewModel.password.collect { password ->
    doSomething(password)
}

// or in Compose
val password by viewModel.password.collectAsStateWithLifecycle()
```



Hot Flow - StateFlow and Mutable State Flow

- *Can* start without subscribers
- *Can* continue without subscribers

Live Data vs StateFlow

LiveData	StateFlow
Holds **state** value	Holds a **state** value
Observe state asynchronously	**Observe**/subscribe asynchronously
Lifecycle aware	***Can*** be **lifecycle** aware
Dispatcher **Main thread** UI update	**Fine grain** dispatcher control including **Main thread**
*Starts with **null***	*Starts with a **known value***

Let’s migrate the app

Up Next

droidcon academy

Coding Challenge 

droidcon academy

Coding Challenge Solution 

Summary

- Explore the starter project and find the LiveData to migrate
- Flow vs LiveData
  - Flows use coroutines and can emit **multiple** values
  - Hot flows vs Cold flows
  - StateFlow can replace LiveData
  - Don’t use LiveData in Repositories
    - Lifecycle aware
    - Uses main thread

- Migrate the `PasswordSource` to use a cold flow to fetch the password

Summary continued

- Migrate the `PasswordRepository`
  - Collect the cold flow and handle errors
  - Create StateFlow for password and history
  - Add tests - Build `FakePasswordSource`
    - - `runTest`
- Migrate the `MainViewModel`
  - Combine password and history in a `UIState`
  - Add tests - Build `FakePasswordRepository`
    - `MainDispatcherRule` - `AdvanceUntilIdle` - `backgroundScope`

- Coding Challenge
- Connect the `MainViewModel` StateFlow to the Composable `MainScreen`
  - `collectAsStateWithLifeCycle()`

droidcon academy

droidcon academy

Thank You

Maia Grotepass

@maiatoday

in/maiagrotepass