**Dagger viewModel (including passing SavedStateHandle)**

<https://stackoverflow.com/questions/64068435/how-to-inject-savedstatehandle-to-viewmodel-in-dynamic-feature-module>

<https://www.strv.com/blog/how-to-set-up-dagger-viewmodel-saved-state-module-engineering> !!!

<https://medium.com/@mikhail_zhalskiy/injecting-savedstatehandle-into-viewmodel-using-dagger-2-and-jetpack-compose-30b34df9ffd1>

<https://medium.com/wriketechclub/how-to-inject-viewmodel-with-dagger-what-might-go-wrong-7954372a7fb9>

**MVI**

<https://github.com/adidas/mvi> + <https://www.youtube.com/watch?v=hBkQkjWnAjg>

<https://medium.com/swlh/mvi-architecture-with-android-fcde123e3c4a>

<https://proandroiddev.com/a-robust-mvi-with-jetpack-compose-e08882d2c4ff>

UiState: Sealed class or single data class

<https://slack-chats.kotlinlang.org/t/4963387/what-are-you-using-for-state-sealed-class-or-data-class-whic> : Casey Brooks11/03/2022, 3:53 PM:

I’ve found that it’s best to manage the state of a screen with a single **data class**, rather than with **sealed classes**. The main reason being that in a real application with non-trivial data, it’s not easy to break the state into discrete units like **Initial, Loading, Value** and **Failure** and still give a good user experience. Instead, those sealed classes work much better as properties of the data class, and give you more flexibility to create more complex UI states without exponentially increasing the number of sealed subclasses. I’ve got a more thorough explanation in this portion of the documentation for the Ballast MVI library.

<https://www.youtube.com/watch?v=NG0PPt-CaYE> - Sealed Classes for UI State are an ANTI-PATTERN - Here's why!

**Paging**

[Android Paging 3: LoadType.APPEND returns null remote keys - Stack Overflow](https://stackoverflow.com/questions/66813622/android-paging-3-loadtype-append-returns-null-remote-keys)

[android - How to save paging state of LazyColumn during navigation in Jetpack Compose - Stack Overflow](https://stackoverflow.com/questions/69739108/how-to-save-paging-state-of-lazycolumn-during-navigation-in-jetpack-compose)

[android - How do I make my own Paging Source when using Remote Mediator in the Paging Library? - Stack Overflow](https://stackoverflow.com/questions/75950852/how-do-i-make-my-own-paging-source-when-using-remote-mediator-in-the-paging-libr)

[android room - Cannot use LazyListScope.items with PagingData in Jetpack Compose - Stack Overflow](https://stackoverflow.com/questions/77139100/cannot-use-lazylistscope-items-with-pagingdata-in-jetpack-compose)

<https://github.com/android/architecture-components-samples/issues/889>

https://stackoverflow.com/questions/64370736/how-to-show-empty-view-while-using-android-paging-3-library

Paging issues:

<https://issuetracker.google.com/issues/288023763>

<https://issuetracker.google.com/issues/183495984>

**Navigation**

<https://proandroiddev.com/navigating-through-multi-module-jetpack-compose-applications-6c9a31fa12b6>

<https://www.youtube.com/watch?v=fp1-YSmdzh8> - Full Guide to Multiple Back Stacks in Jetpack Compose

**Deeplinks**:

<https://stackoverflow.com/questions/57752778/android-deep-links-opens-in-the-app-i-linked-from-instead-of-launching-a-separat>

<https://stackoverflow.com/questions/61480730/deeplink-isnt-correctly-redirect-if-the-app-is-already-opened> - related to previous one

**Gradle precompiled scripts**

* Novinandroid
* <https://github.com/gradle/gradle/issues/15383#issuecomment-779893192> - Make generated type-safe version catalogs accessors accessible from precompiled script plugins

**Compose/UI**

Sample project for modules and material design architecture: nowinandroid

!!! DESIGN: <https://www.figma.com/design/BwnwF9SasanDXGpfKb6Ksu/Rick-and-Morty-(Community)?node-id=135-10917&node-type=canvas&t=rYWWWs7ei5pr0ucW-0>

**Animations**

<https://developer.android.com/develop/ui/compose/animation/shared-elements>

<https://stackoverflow.com/questions/78656716/jetpack-compose-preview-for-screen-that-have-sharedtransitionscope-animatedvisi>

<https://medium.com/@kursatkumsuz/how-to-use-shared-element-transitions-in-jetpack-compose-96e01c266f70>

<https://github.com/fornewid/placeholder> - For use ShimmerEffect. I took files and updated them to be able to define InfiniteTransition. That allows to synchronize shimmer or fade animation, f.e. in lazy items. I used this source - <https://stackoverflow.com/questions/76105638/animating-a-group-of-shimmering-composables-controlling-sync-timing>

<https://stackoverflow.com/questions/67727773/collapse-navigation-bottombar-while-scrolling-on-jetpack-compose> - collapsing appbar during scroll. – interesting, but don’t use because of some bugs**. Better use**: <https://canlioya.medium.com/appbarlayout-in-compose-d379903fd45b> - FlexibleTopBar uses the same scroll behaviors as Material3 top bars, but it doesn't have a layout of its own. It is simply a container in which you can put whatever you want

Google sample project Jetsnack. SharedBound() vs sharedElement(): <https://github.com/android/compose-samples/pull/1314> - It depends, if you have an Async image like this in this case, you'd want to use shared bounds as you may be loading up a higher resolution image. If you are using a standard image that you dont expect to be different, then yes you can use sharedElement :)

<https://developer.android.com/develop/ui/compose/animation/shared-elements#shared-bounds>

Modifier.skipToLookaheadSize() can be used for animation of big texts, as it said here <https://fvilarino.medium.com/shared-element-transitions-in-jetpack-compose-8f553078101e> : "The skipToLookaheadSize() modifier instructs the animation framework to measure the animated child at its final size and to layout the child at that size."