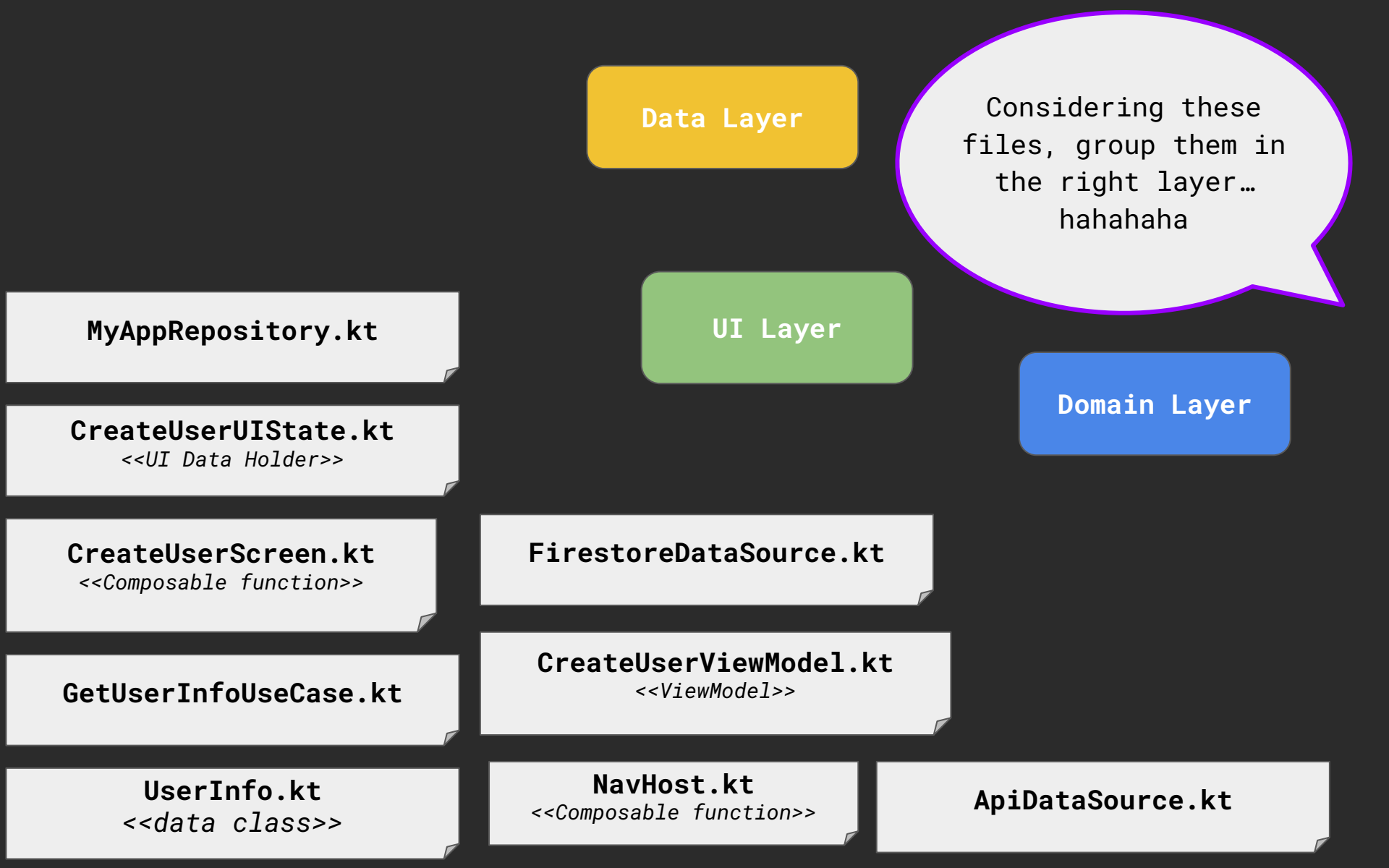
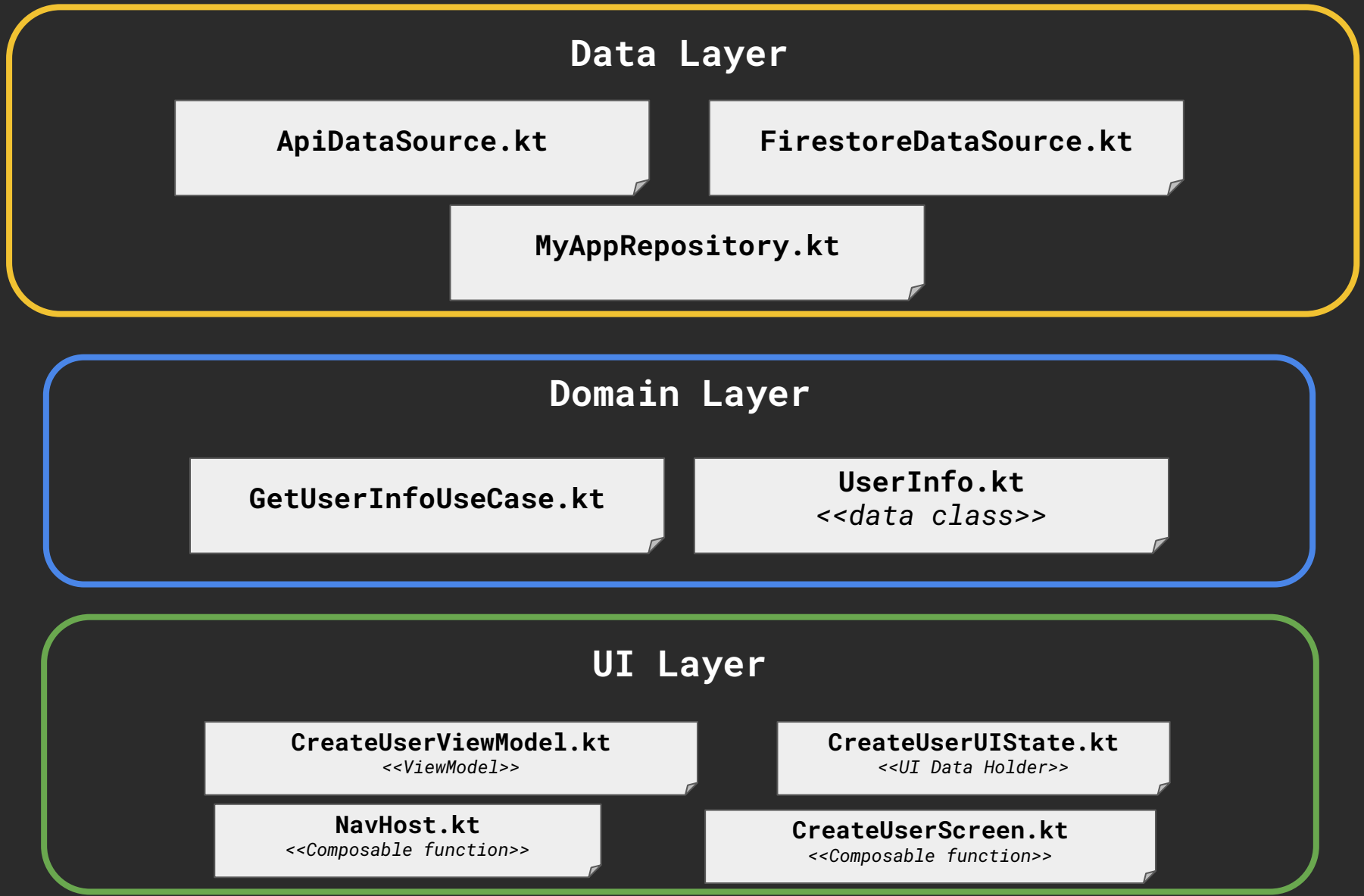
App Architecture

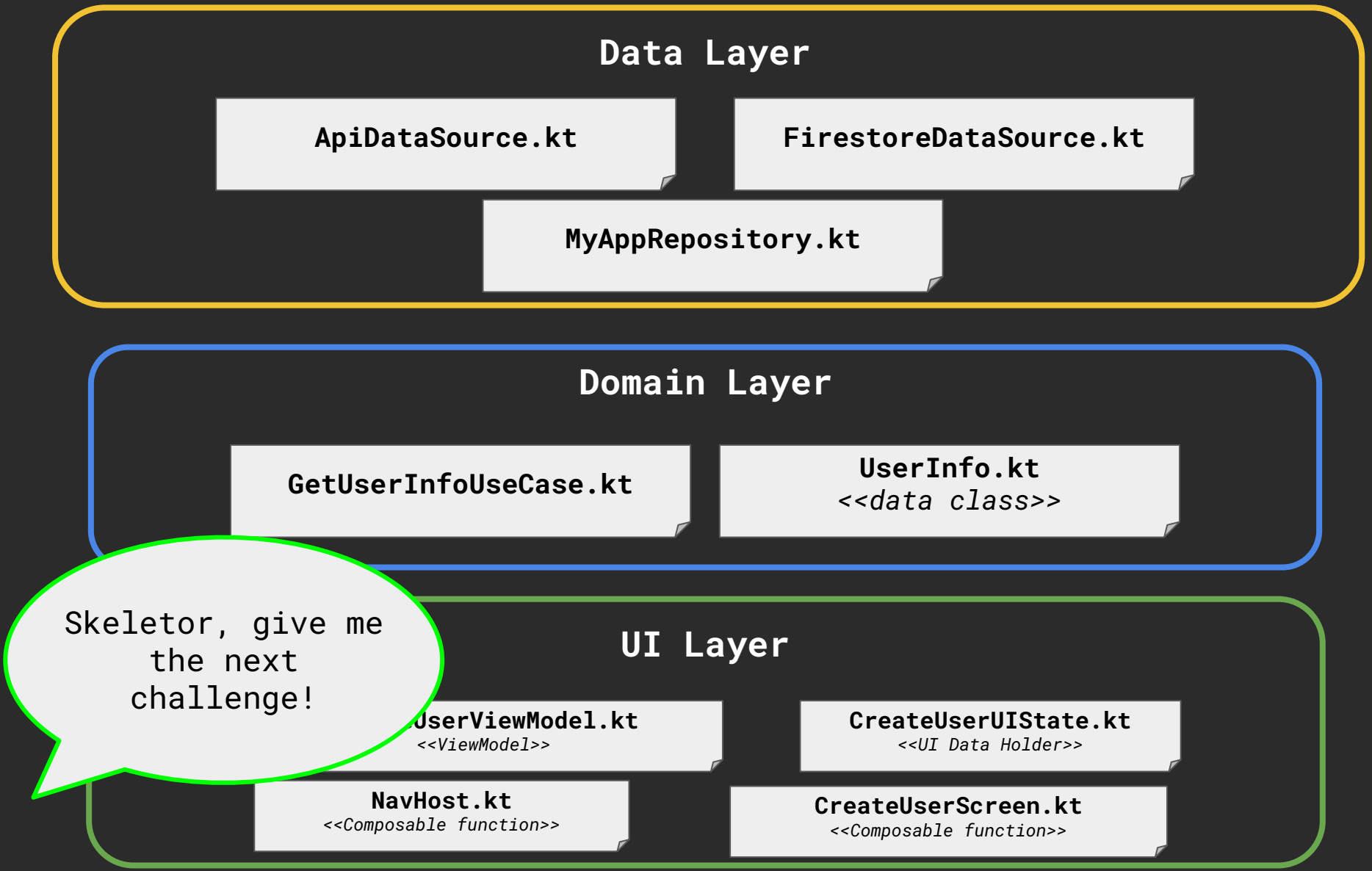
By Moro

Let’s destroy He-Man. Here is a challenge for him, hahahaha


Data Layer
Considering these files, group them in the right layer… hahahaha
UI Layer
Domain Layer
MyAppRepository.kt
CreateUserUIState.kt
<<UI Data Holder>>
CreateUserScreen.kt
<<Composable function>>
FirestoreDataSource.kt
CreateUserViewModel.kt
<<ViewModel>>
GetUserInfoUseCase.kt
UserInfo.kt
<<data class>>
NavHost.kt
<<Composable function>>
ApiDataSource.kt


By the power of
Grayskull…
I HAVE THE POWER


Data Layer
ApiDataSource.kt
FirestoreDataSource.kt
MyAppRepository.kt
Domain Layer
GetUserInfoUseCase.kt
UserInfo.kt
<<data class>>
UI Layer
CreateUserViewModel.kt
<<ViewModel>>
CreateUserUIState.kt
<<UI Data Holder>>
NavHost.kt
<<Composable function>>
CreateUserScreen.kt
<<Composable function>>


Data Layer
ApiDataSource.kt
FirestoreDataSource.kt
MyAppRepository.kt
Domain Layer
GetUserInfoUseCase.kt
UserInfo.kt
<<data class>>
Skeletor, give me the next challenge!
UI Layer
UserViewModel.kt
<<ViewModel>>
CreateUserUIState.kt
<<UI Data Holder>>
NavHost.kt
<<Composable function>>
CreateUserScreen.kt
<<Composable function>>

Next challenge is:
“How to export a string resource from a ViewModel without use context inside of the ViewModel?”

I don’t have the answer, but I have a friend who has. Orkooo!

Of course He-Man, an idea I like is to use **ResourcesProvider**. I will show you.

First, define an interface:

```
interface ResourcesProvider {
    fun getArray(@ArrayRes arrayRes: Int): Array<String>
    fun getString(@StringRes stringRes: Int): String
}
```

Second, implement the interface:

```
class AndroidResourcesProvider @Inject constructor(
    private val resources: Resources
) : ResourcesProvider {

    override fun getArray(@ArrayRes arrayRes: Int): Arr
        return resources.getStringArray(arrayRes)
    }

    override fun getString(stringRes: Int): String {
        return resources.getString(stringRes)
    }
}
```

```kotlin
@Module
@InstallIn(SingletonComponent::class)
object ProvidersModule {

    @Provides
    fun resourcesProvider(@ApplicationContext context: Context): ResourcesProvider {
        return AndroidResourcesProvider(context.resources)
    }
}
```

```
@HiltViewModel
class SettingsViewModel @Inject constructor(
    private val resourcesProvider: ResourcesProvider,
    private val selectThemeUseCase: SelectThemeUseCase,
    private val getSelectedThemeUseCase: GetSelectedTher
) : ViewModel() {
    private val _uiState = mutableStateOf(
        SettingsUIState(
            themeSelection = DropDownValue(
                currentOption = "Automatic",
```

Forth, ask for the provider!


In today's story, we learn to ask for help. If you have a friend, you can beat every challenge.
Upscale

By Moro