

# Dagger Hilt



KA

ALICIA BELTRAN.  
@ALICERESPONDE



# AGENDA

- ▶ Definiciones
- ▶ Motivación – Por que usar un DI?
- ▶ Anotaciones relevantes (Hilt)
- ▶ Componentes & Contenedores
- ▶ GitHub code
- ▶ Bibliografía

# Inversion de Control e Inversion de dependencia

- ▶ Crea un **codigo honesto** – Sin Secretos
- ▶ Expon al mundo lo que necesites
- ▶ Adios a las concreciones
- ▶ Evita el chisme, y has tu trabajo



# Como obtener las dependencias?

- ▶ Inicializar valores por defecto en el constructor
- ▶ Service Locator
- ▶ Injector de dependencias

# Inicializar en el constructor

```
class MoviesRepository(  
    private val localDataSource: LocalDataSource = RoomDataSource(),  
    private val remoteDataSource: RemoteDataSource = TheMovieDbDataSource(),  
    private val regionRepository: RegionRepository = PlayServicesLocationDataSource()  
    private val apiKey: String = MoviesApp.instance.getString(R.string.api_key)  
)
```

- ✓ Facil uso
- ✓ Posible unittest

- ✗ No integration Test
- ✗ Dependencia transitiva
- ✗ Problemas de scope

# Service Locator

- ▶ Componente que provee los servicio a la Aplicacion
- ▶ Application Scope
- ▶ Se crean dentro de la aplicacion
- ▶ Se accede a este desde cualquier parte

```
SL.put<LocalDataSource>(RoomDataSource(db))
SL.put<RemoteDataSource>(TheMovieDbDataSource())
SL.put<LocationDataSource>(PlayServicesLocationDataSource(this))
SL.put<PermissionChecker>(AndroidPermissionChecker(this))
SL.put(RegionRepository(SL.get(), SL.get()))
SL.put(MoviesRepository(SL.get(), SL.get(), SL.get()))
```

# Injector de dependencias

- ▶ Provee las dependencias, no las busco
- ▶ Emplean grafo de dependencias
- ▶ Genera código por mí que facilita la creación de elementos
- ▶ Grafos locales relacionados a componentes



# Inyección de dependencias



- ▶ Sólo pídele al genio lo que necesites que el sabra proveerlo, siempre que conozca la receta para crearlo.
- ▶ Ahora.. haz tu trabajo con ello.

Sin Inyeccion de  
dependencias ..  
Como seria?

- ▶ Creas todo tantas veces como lo necesites
- ▶ Largo – Tedioso – Aburrido
- ▶ Inflexible a cambios
- ▶ Costoso – No Mantenible



```
class MusicPlayer {  
    private val db = SQLiteDatabase()  
    private val codecs = listOf(CodecH264(), CodecFLAC())  
  
    fun play(id: String) { ... }  
}
```

Exponer las dependencias en  
el constructor



# Logica de Construcción Vs Logica del Negocio?

```
class MusicPlayer(  
    private val db: Database,  
    private val codecs: List<Codec>  
) {  
    fun play(id: String) { ... }  
}  
  
fun main() {  
    val db = SQLiteDatabase()  
    val codecs = listOf(CodecH264(), CodecFLAC())  
    val player = MusicPlayer(db, codecs)  
    player.play("YHLQMDLG")  
}
```

# Por qué tener un Wrapper de Dagger??

- ▶ Elementos del FrameWork de Android son instanciados por el SDK
- ▶ Factories
- ▶ Difícil configuracion Componentes, modulos, submodulos,factories,scopes, builders ...
- ▶ Standard

# Con Hilt

- ▶ Facil setup
- ▶ Olviate de los componentes – Hilt te los da
- ▶ Compatible con Dagger
- ▶ Androidx Extensions
- ▶ Soporte ViewModel, Work Manager
- ▶ Anotaciones de uso intuitivo

# Gradle dependencies

## Build.gradle : project

```
→ classpath "com.google.dagger:hilt-android-gradle-plugin:$hilt_version"
```

## Build.gradle : app

```
kapt {  
    → correctErrorTypes true  
}
```

```
plugins {  
    id 'com.android.application'  
    id 'kotlin-android'  
    id 'kotlin-android-extensions'  
    id 'kotlin-kapt'  
    id 'dagger.hilt.android.plugin'  
}  
→ }
```

```
// hilt  
implementation 'com.google.dagger:hilt-android:2.28.3-alpha'  
kapt 'com.google.dagger:hilt-android-compiler:2.28.3-alpha'
```

## @ViewModellInject

```
→ implementation 'androidx.hilt:hilt-lifecycle-viewmodel:1.0.0-alpha02'  
kapt 'androidx.hilt:hilt-compiler:1.0.0-alpha02'
```

# Anotaciones

- ▶ **@HiltAndroidApp** : al inicio de tu application para emplear el procesador de anotaciones de hilt y hacerla **inyectable**.
- ▶ **@AndroidEntryPoint**: Al inicio de los componentes de Android (Activity, Fragments, Views, Services) donde espero Hilt inyecte dependencias.
- ▶ **@EntryPont**: Al inicio de la interface para indicar a clases no soportadas por hilt, que ella va a usar las dependencias de hilt.

**@Inject**: para inyectar la dependencia en el constructor o como atributo de las clases propias de android, o en los test.  
**@Module**: objeto (**object**) que le dice a hilt las recetas de construccion  
**@InstallIn**: Indica en que componente de hilt deberia estar instalado indicando asi su scope y algunas dependencias para mi.  
**@Provides**: la receta a usar cuando se requiera una dependencia

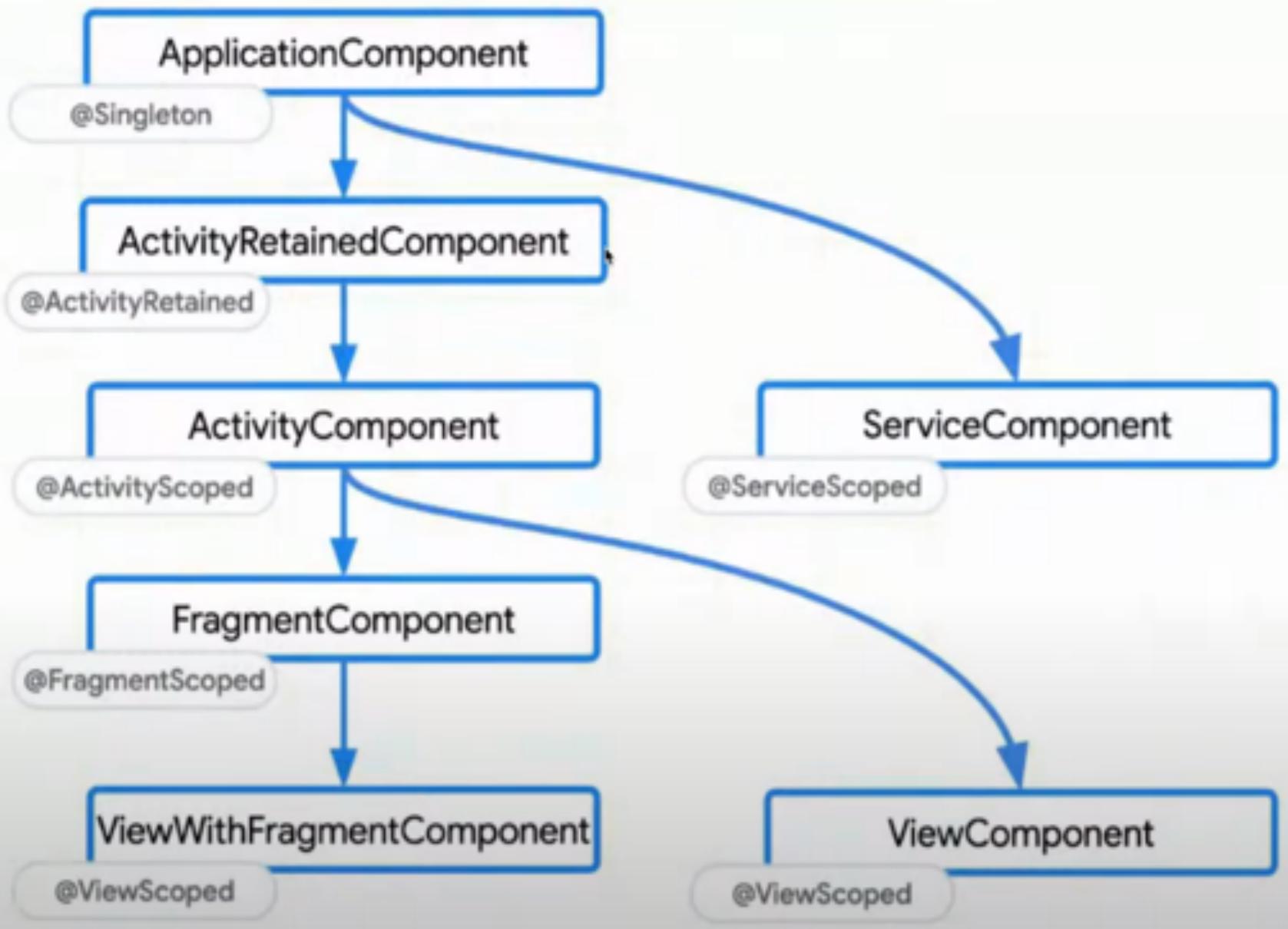
# Anotaciones - ktx

```
class PlayViewModel @ViewModelInject constructor(  
    @Assisted val savedStateHandle: SavedStateHandle  
    val db: MusicDatabase,  
) : ViewModel() {  
    ...  
}  
  
@AndroidEntryPoint  
class PlayActivity : AppCompatActivity() {  
  
    val viewModel: PlayViewModel by viewModels()  
  
    override fun onCreate(savedInstanceState: Bundle) {  
        super.onCreate(savedInstanceState)  
  
        viewModel.play()  
    }  
}
```

- ▶ **@ViewModelInject** : el **SDK** lo crea por ti usando el **by** delegate sin factories
- ▶ **@Assisted** : para dependencias del framework de android como por ejemplo `savedStateHandle`

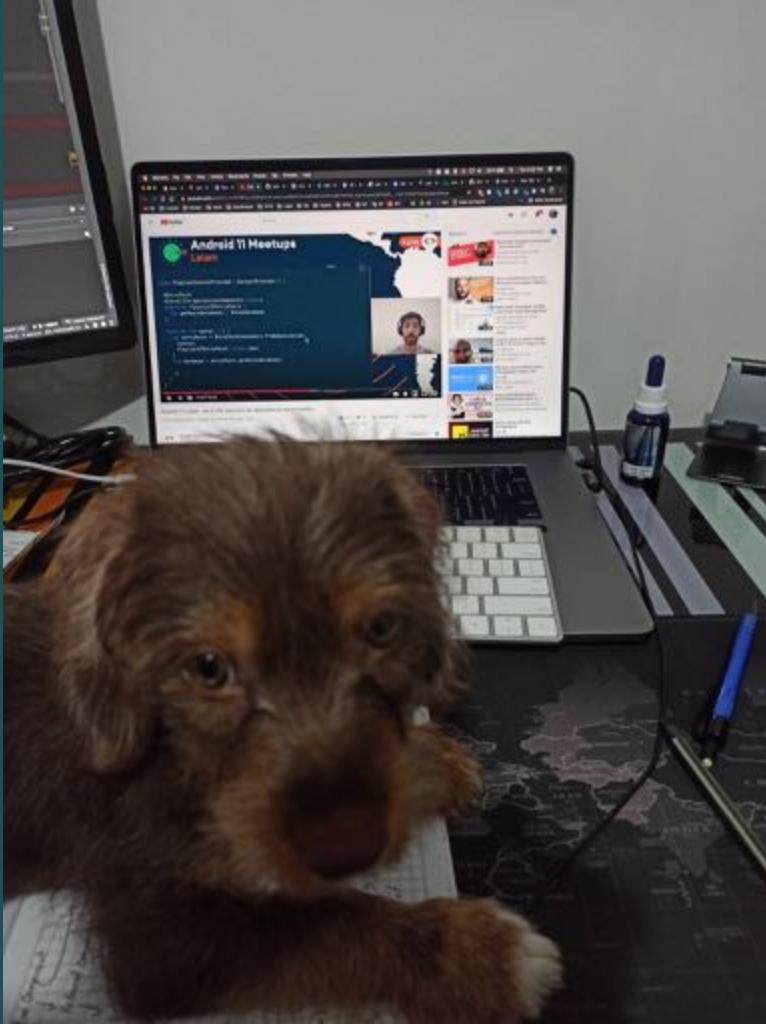
# Componentes :

## Contenedores de inyección de dependencias & Scopes



```
class PlaylistContentProvider : ContentProvider() {  
  
    @EntryPoint  
    @InstallIn(ApplicationComponent::class)  
    interface PlaylistCPEntryPoint {  
        fun getMusicDatabase(): MusicDatabase  
    }  
  
    override fun query(...){  
        val entryPoint = EntryPointAccessors.fromApplication(  
            context,  
            PlaylistCPEntryPoint::class.java  
        )  
        val database = entryPoint.getMusicDatabase()  
  
        // ...  
    }  
}
```

**ContentProvider**  
No esta  
soportado por  
Hilt  
directamente



Miremos el ejemplo:  
MVVM + Clean + Repository

Di (manual) vs Hilt

<https://github.com/aliceresponde/CompanyHumanResources>

# Bibliografia

- ▶ An Opinionated Guide to Dependency Injection on Android (Android Dev Summit '19) <https://youtu.be/o-ins1nvbDg>
- ▶ Migration Guide: <https://dagger.dev/hilt/migration-guide>
- ▶ Hilt Android Docs: <https://d.android.com/hilt>
- ▶ Codelabs:  
<https://codelabs.developers.google.com/codelabs/android-hilt/#0>
- ▶ Hilt - Android Dependency Injection:  
[https://www.youtube.com/watch?v=B56oV3IHxg&ab\\_channel=AndroidDevelopers](https://www.youtube.com/watch?v=B56oV3IHxg&ab_channel=AndroidDevelopers)
- ▶ GDG Buenos Aires:  
[https://www.youtube.com/watch?v=VI\\$myqGI3CE&t=3156s&ab\\_channel=GDGBuenosAires](https://www.youtube.com/watch?v=VI$myqGI3CE&t=3156s&ab_channel=GDGBuenosAires)

# Bibliografia

- ▶ <https://architectcoders.com/>
- ▶ Martin Fowler <https://martinfowler.com/articles/injection.html>
- ▶ Project Overview & Architecture - MVVM Running App

[https://www.youtube.com/watch?v=XqkFTG10sRk&list=PLQkwcJG4YTCQ6emtoqSZS2FVwZR9FT3BV&ab\\_channel=PhilippLackner](https://www.youtube.com/watch?v=XqkFTG10sRk&list=PLQkwcJG4YTCQ6emtoqSZS2FVwZR9FT3BV&ab_channel=PhilippLackner)