THE DEVELOPER'S
CONFERENCE

Trilha – Android/Kotlin

Diego Cunha
Android Developer IWS Brazil

Agenda

THE DEVELOPER'S CONFERENCE

- Android Jetpack
- O que é injeção de dependências
- Koin, o grande inimigo do Dagger2
- Android Jetpack + Koin: Casamento perfeito

THE DEVELOPER'S CONFERENCE

Sobre mim

- Android Developer 3 anos na IWS Brazil
- Formado em Analise e Desenvolvimento de Sistemas
- Pós graduado em Metodologias Ágeis
- Cozinheiro nas horas vagas
- Evangelista de Uncle Bob

THE DEVELOPER'S CONFERENCE

Android Jetpack

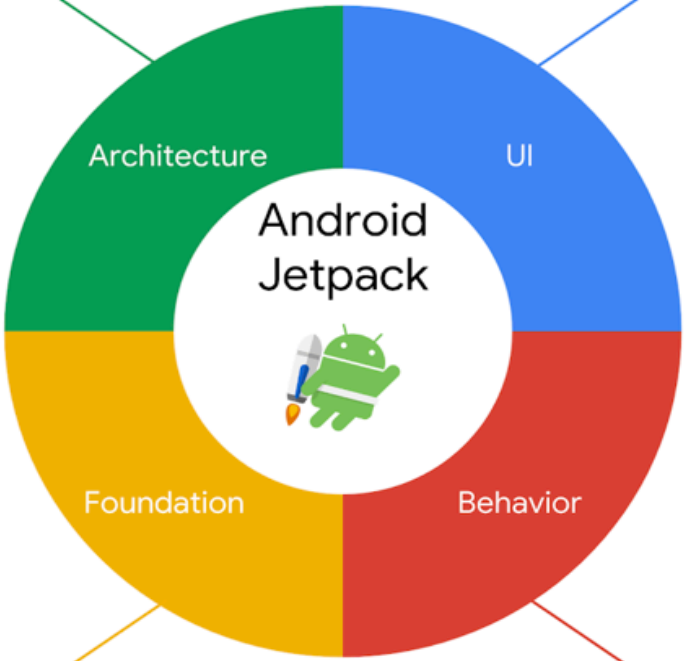
A Google corrigindo os erros do passado

Android Jetpack

THE DEVELOPER'S CONFERENCE

- Conjunto de bibliotecas que visam ajudar desenvolvedores a criar aplicativos mais facilmente.
- Trouxe um padrão de projeto para o Android (Antes eles faziam exemplos com o MVVM).
- Corrigiram os erros do passado
- Dividido em componentes

Android Jetpack



- Data Binding
- Lifecycles
- LiveData
- Navigation *new!*
- Paging *new!*
- Room
- ViewModel
- WorkManager *new!*

- AppCompat
- Android KTX *new!*
- Multidex
- Test

- Animation & Transitions
- Auto, TV & Wear
- Emoji
- Fragment
- Layout
- Palette

- Download Manager
- Media & Playback
- Permissions
- Notifications
- Sharing
- *new!* Slices

Android Jetpack: Architecture

THE DEVELOPER'S CONFERENCE

- Bibliotecas para tornar o aplicativo mais robusto, testável e de fácil manutenção

- Lida com problemas como ciclo de vida, data binding, navigation, workmanager, banco de dados entre outros componentes responsáveis por dar uma arquitetura ao projeto.

Android Jetpack: UI

THE DEVELOPER'S CONFERENCE

- Responsável por todos os componentes que tem interação direta com o usuário
- Atualizou e corrigiu erros existentes em fragmentos
- Facilitou a utilização do Material Design do Google

Android Jetpack: Behaviour

THE DEVELOPER'S CONFERENCE

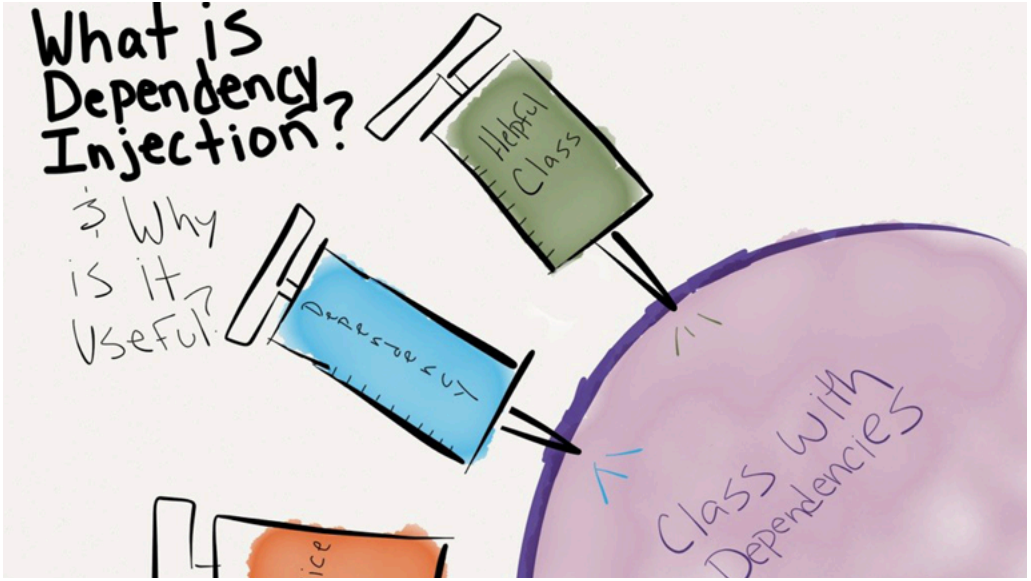
- Responsável por conectar com serviços do sistema operacional Android
- Removeu boilerplate para questões como uso de camera, requisição de notificações
- CameraX, Slice, Notifications

Android Jetpack: Foundation

THE DEVELOPER'S CONFERENCE

- Responsável pela retro compatibilidade das bibliotecas
- Trouxe o suporte do Kotlin ao Android
- Android KTX, AppCompat entre outros

THE DEVELOPER'S CONFERENCE



Injeção de dependência

O que é? Do que se alimenta? Onde vive?

Injeção de dependência

THE DEVELOPER'S CONFERENCE

- Padrão de projeto onde o foco é o baixo acoplamento
- Remover a necessidade de classes específicas
- Você pode imaginar ela como uma caixa de ferramentas enquanto você monta uma cadeira
- Spring, Dagger2, ASP.NET Core e…..Koin!

Koin

Say hello to my little friend

Koin

THE DEVELOPER'S CONFERENCE

- Biblioteca de injeção de dependência
- 100% escrita em Kotlin
- Mais limpo que Dagger2
- Suporte a Unity Tests/Testes de UI

Koin

THE DEVELOPER'S CONFERENCE

- Desenvolvida em módulos
- Sem geração de código nem se utiliza de proxys e reflexão
- 100% compatível com Android Jetpack ViewModel

Koin: Pilares

THE DEVELOPER'S CONFERENCE

- get: Injeta a instância requisitada no construtor da classe

- factory: Cria uma nova instância injetável cada vez que é chamada.

Koin: Pilares

THE DEVELOPER'S CONFERENCE

- single: Cria um singleton da classe/interface a ser utilizada

- viewModel: Utilizada em conjunto com Jetpack e gera uma instância de ViewModel com respeito ao ciclo de vida

Koin:Pilares

THE DEVELOPER'S CONFERENCE

```
1 val appModule = module {
2     factory { GsonBuilder().create() }
3     single { MarvelRetrofitRepository(get()) as MarvelRepository }
4     viewModel { HomeViewModel(get()) }
5 }
```

Koin: Inicialização

THE DEVELOPER'S CONFERENCE

```
override fun onCreate() {
  super.onCreate()
  startKoin(this, listOf(appModule))
}
```

Koin: ViewModel

THE DEVELOPER'S CONFERENCE

```
1 class HomeViewModel(private val repository: MarvelRepository) : ViewModel() {
2     private val _heroes = MutableLiveData<List<Hero>>()
3     val heroes: LiveData<List<Hero>> = _heroes
4   ...
5 }
```

Koin: Fragment

THE DEVELOPER'S CONFERENCE

```
1 private val viewModel: HomeViewModel by viewModel()
2     override fun onCreateView(...): View? {
3         val binding = FragmentHomeBinding.inflate(inflater, container, false)
4         binding.viewModel = viewModel
5       ...
6 }
```

THE DEVELOPER'S CONFERENCE

Koin + Android Jetpack

O casamento perfeito

Koin + Jetpack = <3

THE DEVELOPER'S CONFERENCE

- Koin é de fácil integração com Android, visto que tem um módulo só para isso
- Aplicativos com tamanho de .apk muito menor
- Menos tempo de builds

Koin + Jetpack = <3

THE DEVELOPER'S CONFERENCE

Description	Duration
Total Build Time	12.878s
Startup	0.500s
Settings and BuildSrc	0.001s
Loading Projects	0.003s
Configuring Projects	0.074s
Task Execution	16.619s

Description	Duration
Total Build Time	7.088s
Startup	0.519s
Settings and BuildSrc	0.002s
Loading Projects	0.006s
Configuring Projects	0.084s
Task Execution	8.333s

Koin + Jetpack = <3

THE DEVELOPER'S CONFERENCE

- Sintaxe mais limpa nas camadas de View/ViewModel
- Fácil aprendizado de código para iniciantes
- Menos boilerplate

Koin + Jetpack = <3

THE DEVELOPER'S CONFERENCE

- Sem geração de código
- Jetpack te proporciona abstrair coisas antigas como SQLite e WorkManagers
- Jetpack permite criar um aplicativo reativo com poucos passos

Koin + Jetpack = <3

THE DEVELOPER'S CONFERENCE

- Koin permite pessoas novas em desenvolvimento entender Injeção de Dependência

- Koin tem fácil integração com testes

Koin + Jetpack = <3 Teste

THE DEVELOPER'S CONFERENCE

```
1 val roomTestModule = applicationContext {
2     single {
3         Room.inMemoryDatabaseBuilder(get(), ComicDatabase::class.java)
4             .allowMainThreadQueries()
5             .build()
6     }
7 }
```

Koin + Jetpack = <3 Teste

THE DEVELOPER'S CONFERENCE

```
1 ...
2
3     val ComicDatabase: ComicDatabase by inject()
4     val ComicDAO: ComicDAO by inject()
5 ...
```

Koin + Jetpack = <3 Teste

THE DEVELOPER'S CONFERENCE

```
1 @Before
2 fun onSetup() {
3     loadKoinModules(roomTestModule)
4 }
5
6 @After
7 fun onAfter(){
8     ComicDataBase.close()
9     closeKoin()
10 }
```

Koin + Jetpack = <3 Teste

THE DEVELOPER'S CONFERENCE

```
1     @Test
2     fun testSave() {
3         val comic = 12343
4         val now = Date()
5         val entities = createComicEntities(comic, now)
6         ComicDAO.saveAll(entities)
7         val ids = entities.map { it.id }
8         val requestedEntities = ids.map {
          ComicDAO.findComicById(it).blockingGet() }
9
10        Assert.assertEquals(entities, requestedEntities)
11    }
```

Obrigado!

THE DEVELOPER'S CONFERENCE