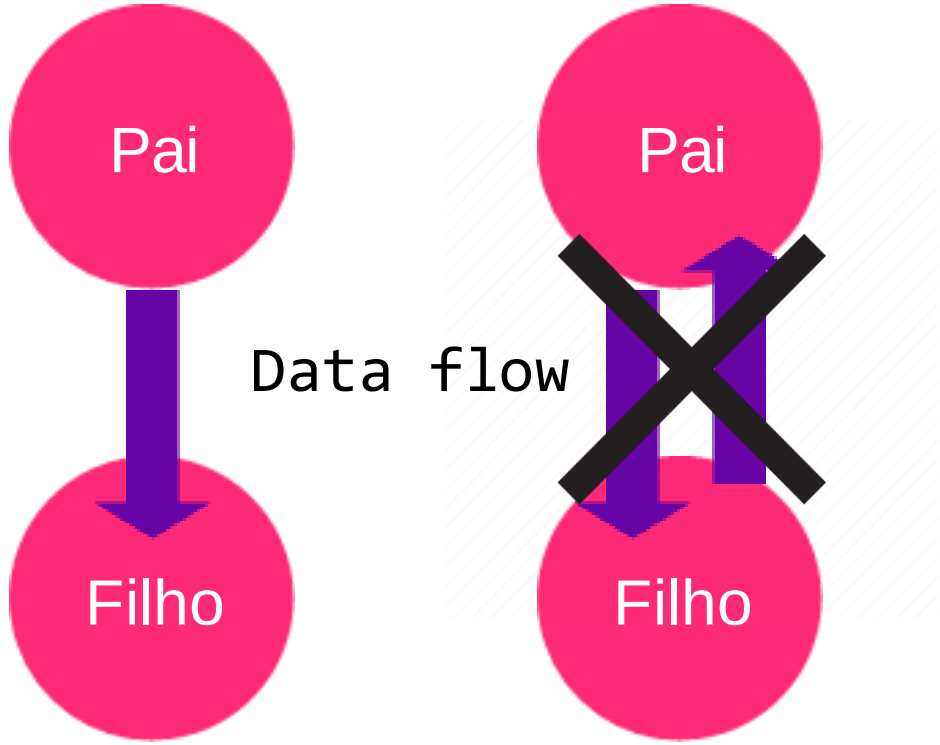
Android Tech Talk de Cássio Bruzasco

Arquitetura, observer pattern e injeção de dependência moderna

Arquitetura - MVVM

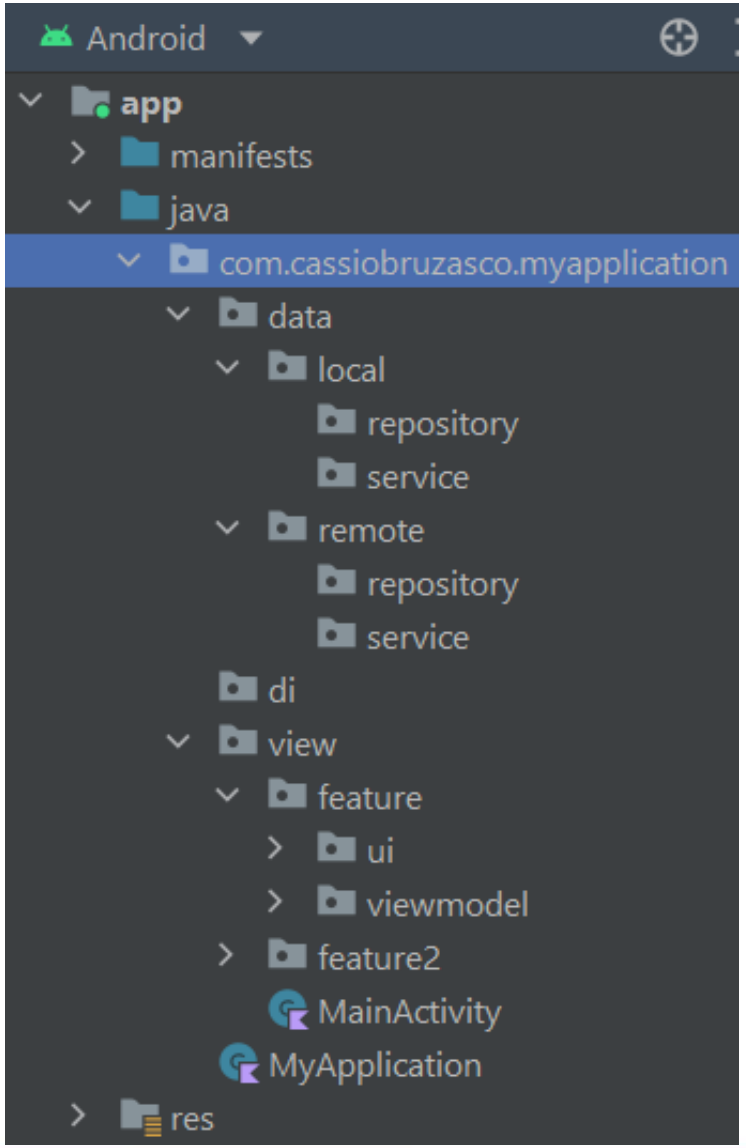
Model – ViewModel – View

- Manutenibilidade
- Separar responsabilidades
- Unidirectional Data Flow
- Entre outros motivos mais profundos...

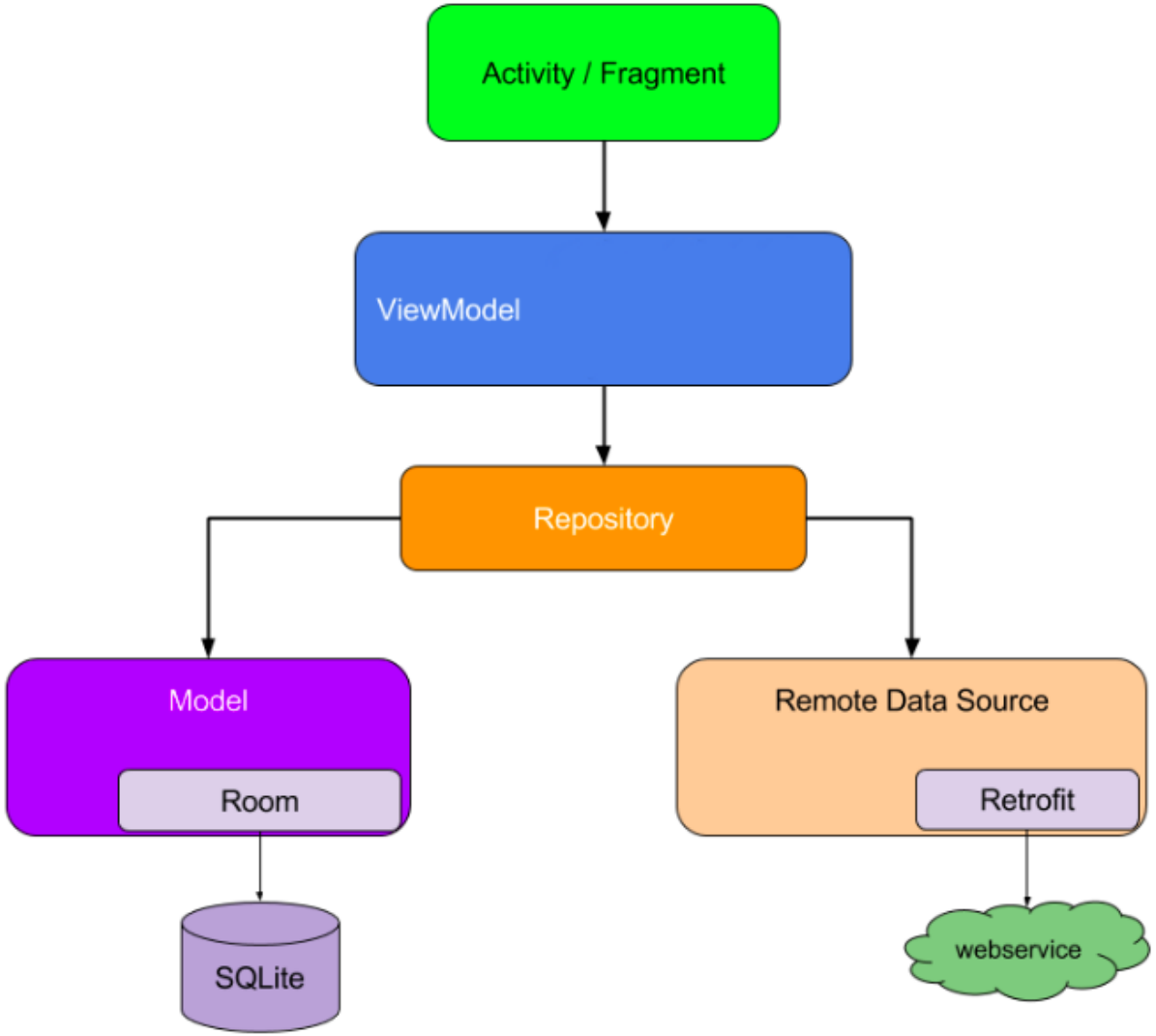


Model – ViewModel – View

- **data** – tudo relacionado a dados (Camada Model)
- **local** – requisições e modelar o banco de dados interno
- **remote** – requisições e modelar serviços externos (API)
- **repository** – interface/implementação que faz a comunicação do DB ou API com a camada do VM
- **di** – módulos para implementação do injetor de dependência
- **view** – pacote de apresentação
- **viewmodel** – pacote para relacionar seu fragment/activity com um VM



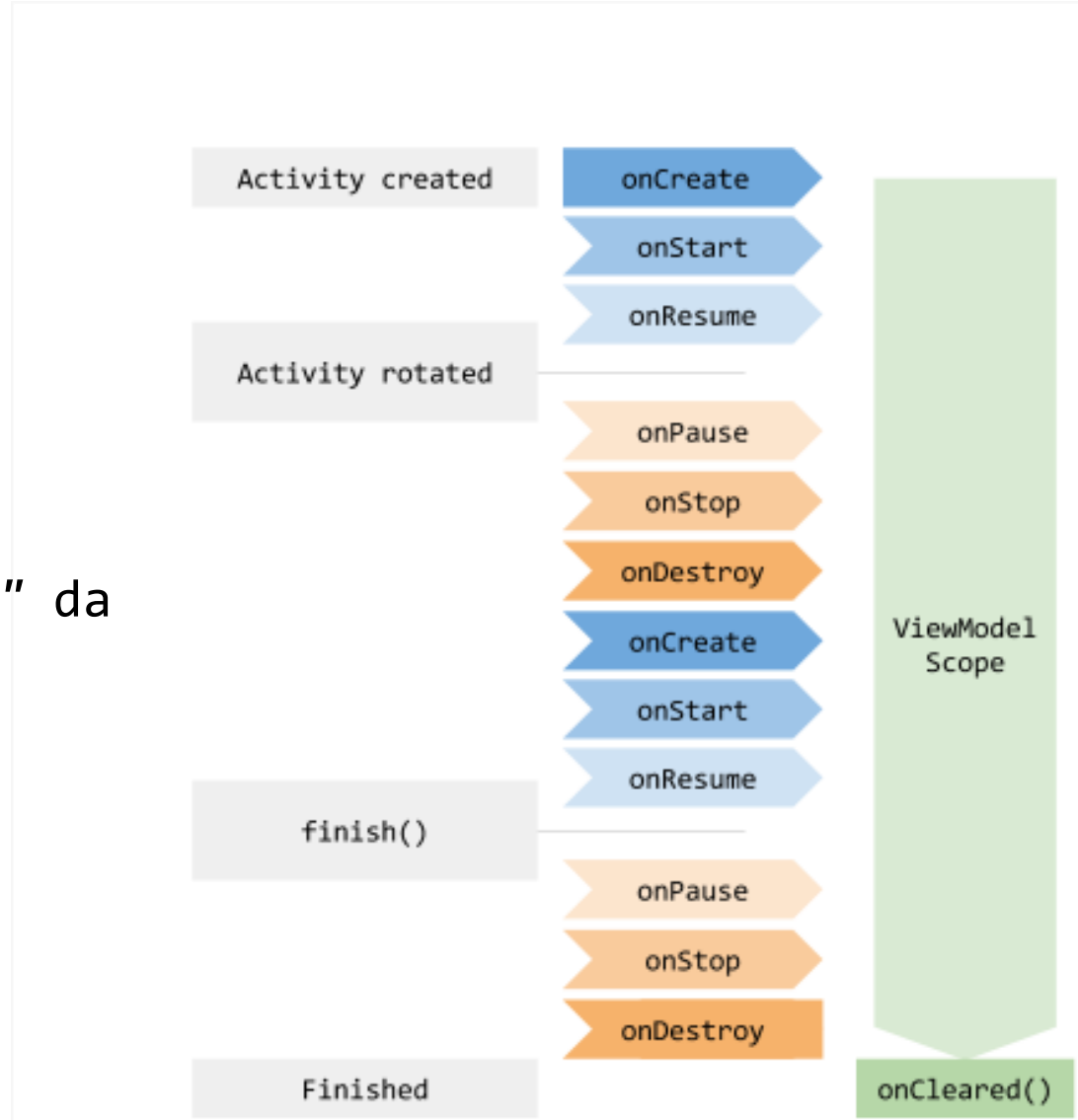
Model – ViewModel – View



ViewModel

ViewModel

- Persistir o *"current state"* da UI
- Acessar lógica de negócio
- *Lifecycle aware*

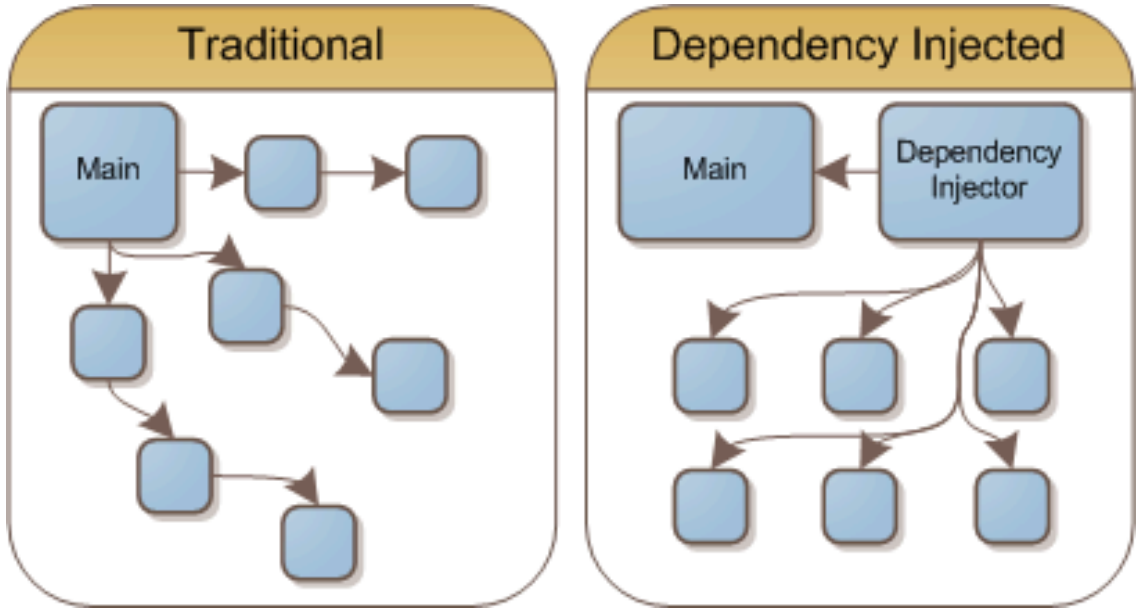


Observer Pattern

Programação reativa

- Antigamente se usada a expressão programação reativa
- Google não aconselha o uso do *LiveData* desde o lançamento do *Jetpack Compose*
- *Flow* e *StateFlow* são *state aware* e funcionam de passo em passo
- *StateFlow* tem conexão direta com estados da UI

Injeção de dependência



- Koin não é um injetor de dependência e sim um *Service Locator* o que atualmente é considerado um *anti-pattern*
- Grande ganho em testes unitários
- Uso mais inteligente de memoria
- Facilidade de comunicação entre submódulos

	build time	runtime performance
Koin	no impact	impact
Dagger	impact	no impact

Demonstração

github.com/cassiobruzasco/TechTalk-VNT