instituto
superior de
engenharia
de lisboa

POLITÉCNICO
DE LISBOA
POLYTECHNIC
UNIVERSITY OF LISBON

View Model

PDM - Programação para Dispositivos Móveis

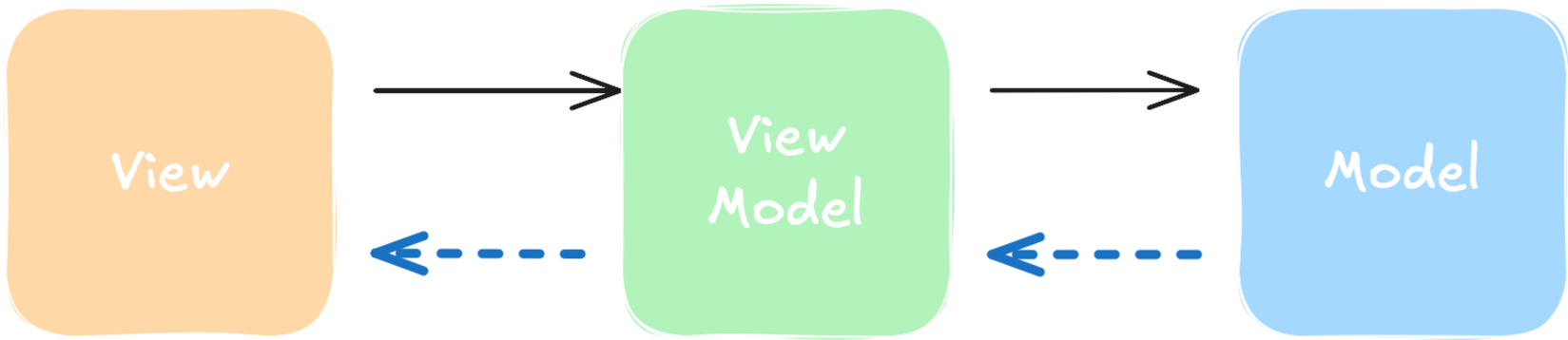
Paulo Pereira
paulo.pereira@isel.pt

PRO
DIGI

View Model

- O que é um View Model?
- Porque existe? Que problema resolve?
- Qual o seu papel? Que função tem na solução?
- Qual a sua relação com os restantes elementos da solução?
  - Como são realizadas as interações?

O padrão MVVM



MVVM em Android

- A Activity é a hospedeira da UI
  - Pode fazer as vezes do View Model do padrão MVVM?
  - Podem as Activities ser as interlocutoras com o “resto do sistema”?

- Numa palavra, **não**
  (mas no passado tentou-se)

O “resto do sistema”

- Dados e lógica de domínio
- Interacções com outros elementos da solução

I/O

- Backend servers (e.g. através de Web APIs)
- Outros dispositivos (e.g. através de Bluetooth ou Wi-fi)
- Armazenamento persistente de dados, local ou remoto

Concorrência em Android

- Todos os callbacks são executados na Main Thread
  - A sua execução TEM de ser breve
    - I/O bloqueante é proibido
    - Computação intensiva é proibida
- **As operações demoradas têm carácter assíncrono**

Activity nas reconfigurações

- A instância da Activity é descartada nas reconfigurações
- O que acontece às operações assíncronas em curso?
  - Cancelam-se?
  - Mantêm-se? E como se usam os resultados?
- **A Activity NÃO é um bom hospedeiro para estas operações**

Classe ViewModel

- Instâncias são os View Models do padrão MVVM
- Instâncias não são descartadas nas reconfigurações
  - Ficam associadas a ViewModelStoreOwners:
    - Activity, Fragment e NavBackStackEntry
- **São os hospedeiros para as operações assíncronas (por desenho, em Android moderno)**

Ciclo de vida

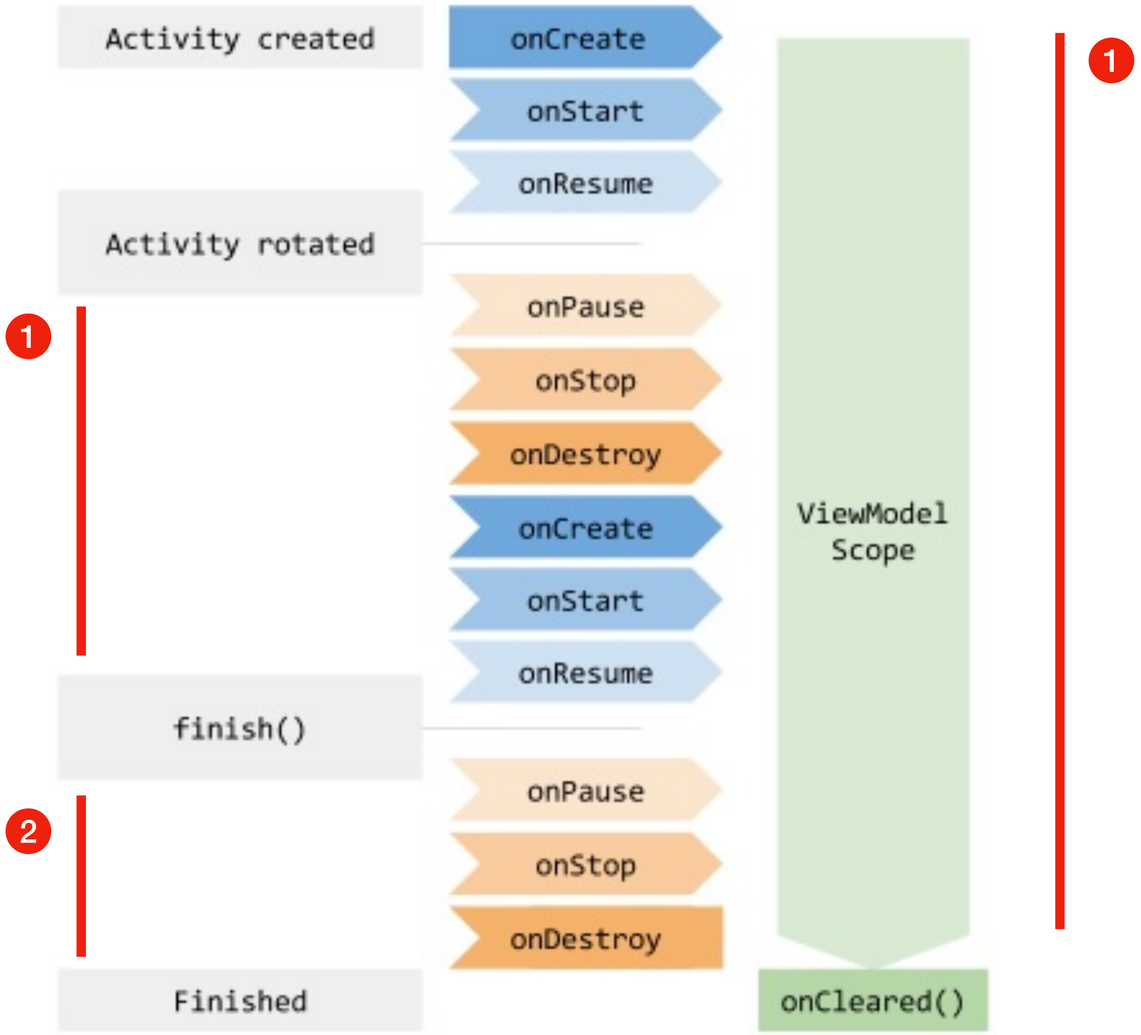
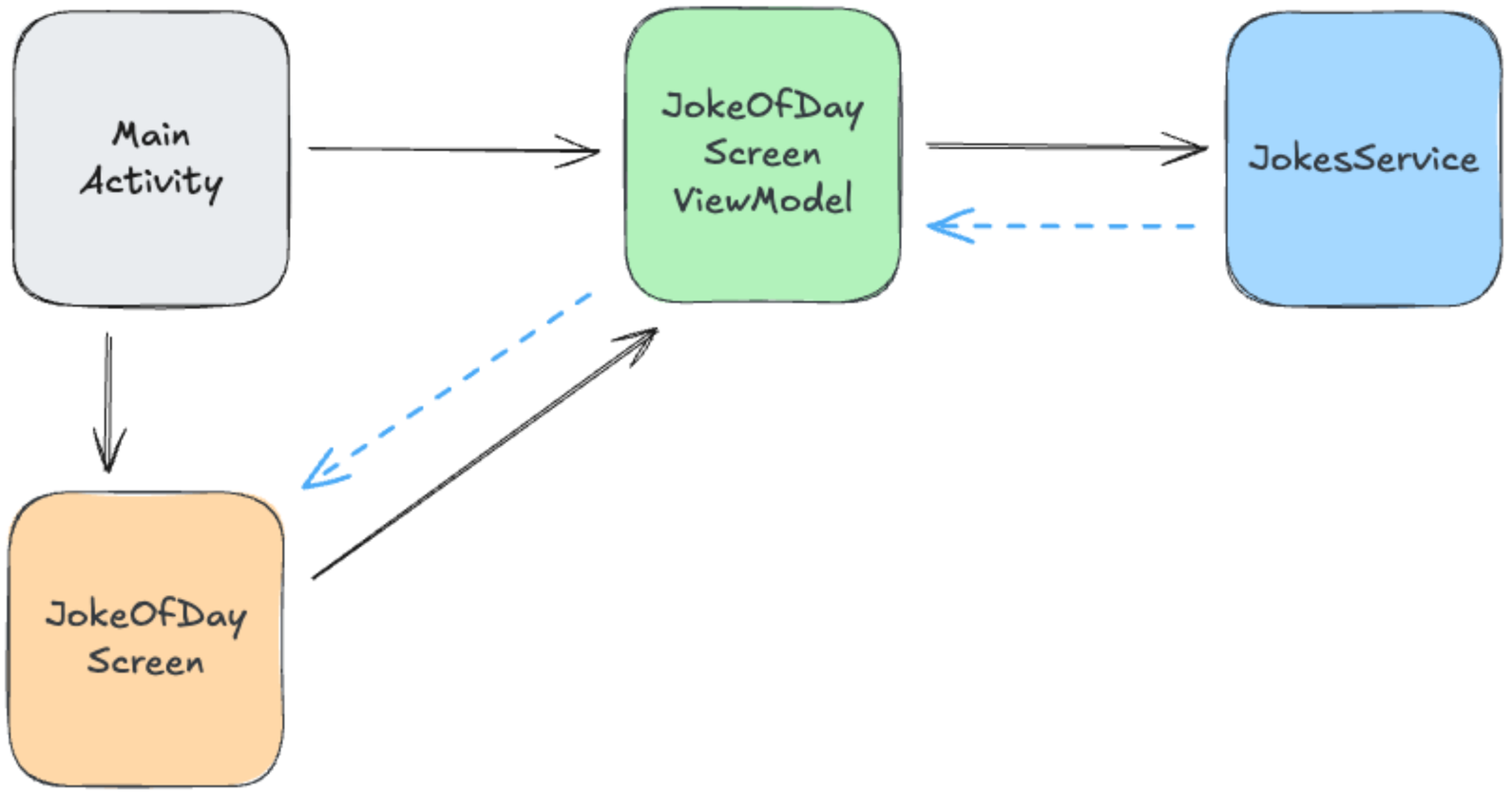


Imagem extraída da documentação oficial

Exemplo

- Demo *Joke Of Day*
  - Apresenta piadas obtidas a partir de web APIs
  - Não considera a existência de ecrãs com tamanhos variáveis
- Implementação encontra-se aqui:
  - https://github.com/palbp/pdm.prodigi/tree/main/demos/JokeOfDay
  - Usa a API https://icanhazdadjoke.com

Joke Of Day

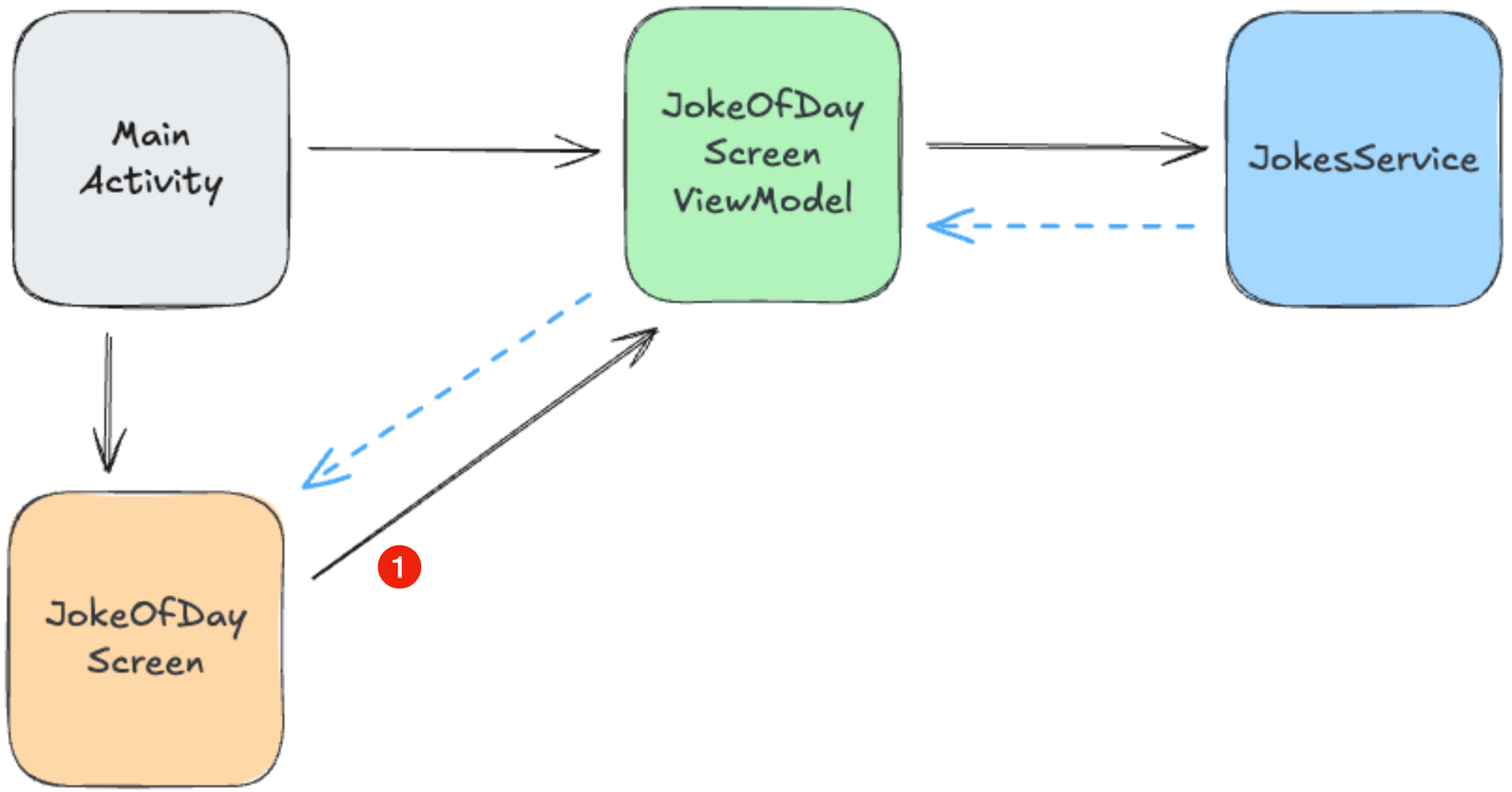

Main Activity
JokeOfDay Screen ViewModel
JokesService
JokeOfDay Screen

JokesService

```
interface JokesService {
    suspend fun fetchJoke(): Joke
}

data class Joke(val text: String, val source: URL) {
    init {
        require(text.isNotBlank()) { "The joke's text must not be blank" }
    }
}
```

- Requisito de acesso a web API modelado usando abstracção
- Operação de I/O representada através de função *suspend*

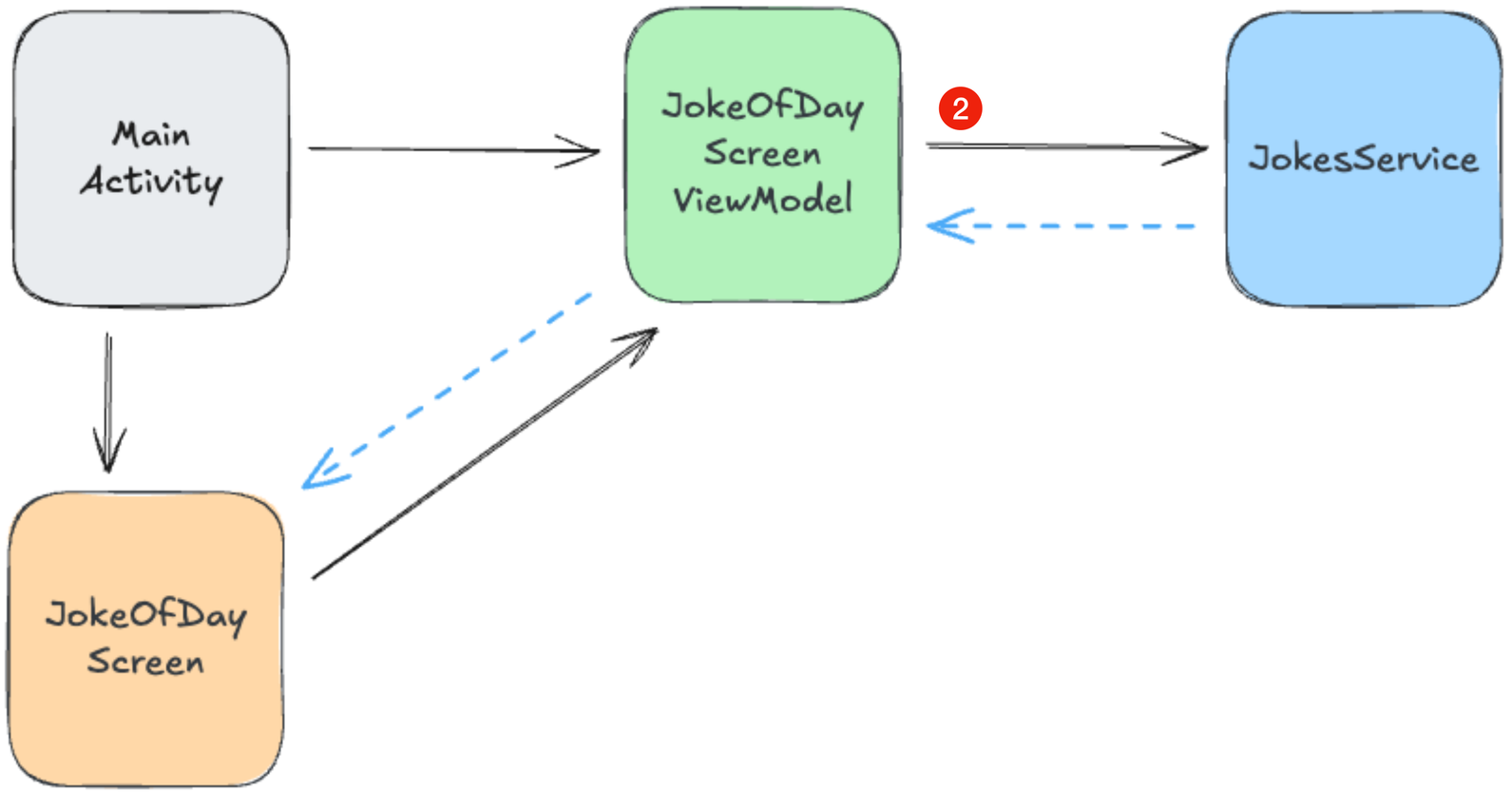
Joke Of Day (1)


Main Activity
JokeOfDay Screen ViewModel
JokesService
JokeOfDay Screen
1

JokeOfDayScreenViewModel (1)

```
class JokeOfDayScreenViewModel(val jokeService: JokesService) : ViewModel() {

    private var state: JokeOfDayScreenState by mutableStateOf(value = JokeOfDayScreenState.Idle)
    val currentState: JokeOfDayScreenState
        get() = state

    fun fetchJoke() {
1       if (state is JokeOfDayScreenState.Loading) {
            return
        }

        viewModelScope.launch {
            state = try {
                state = JokeOfDayScreenState.Loading
                val joke = jokeService.fetchJoke()
                JokeOfDayScreenState.Success(joke)
            } catch (e: Exception) {
                JokeOfDayScreenState.Error(e)
            }
        }
    }
}
```

Joke Of Day (2)

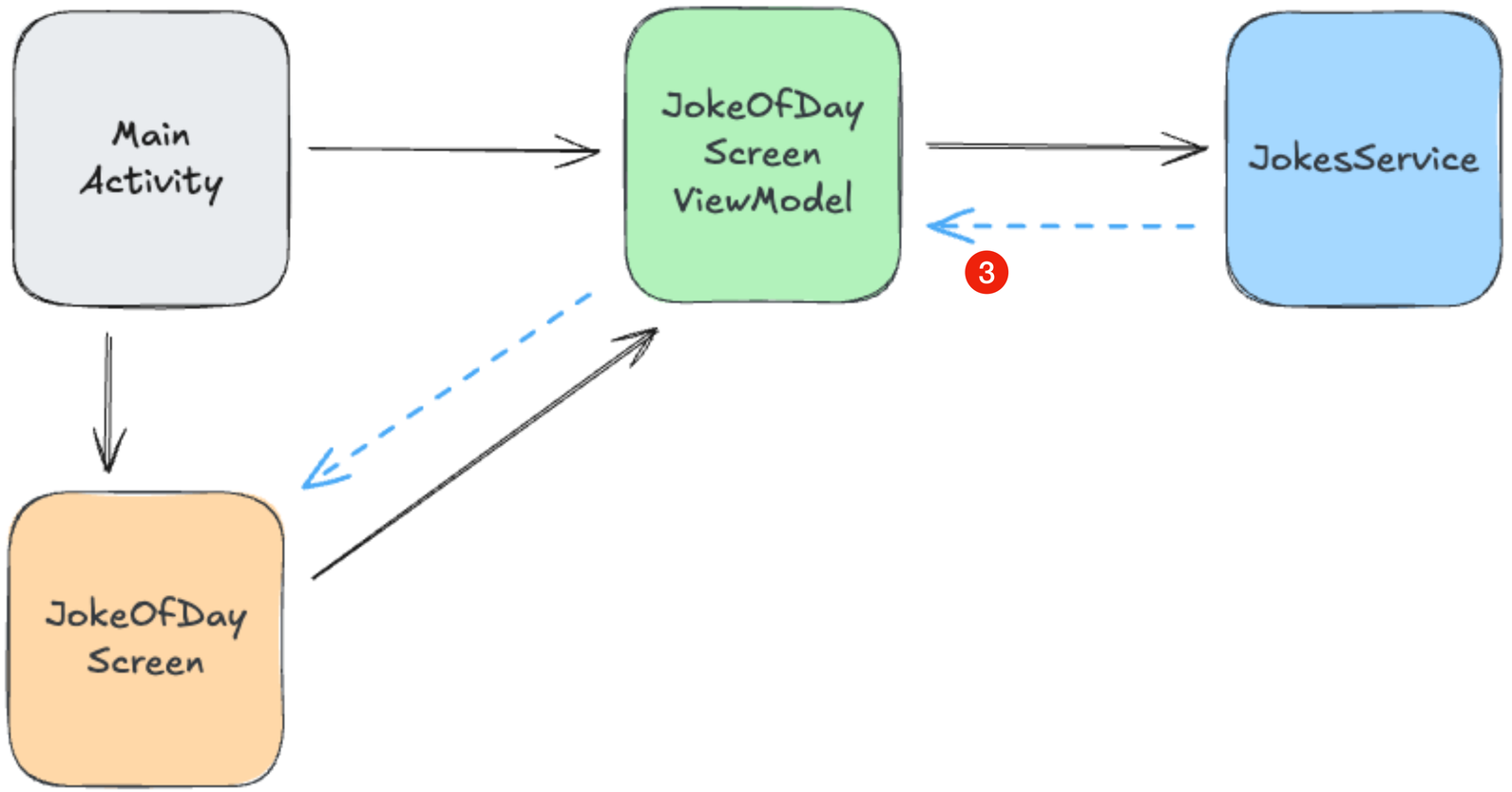

Main Activity
JokeOfDay Screen ViewModel
2
JokesService
JokeOfDay Screen

JokeOfDayScreenViewModel (2)

```
class JokeOfDayScreenViewModel(val jokeService: JokesService) : ViewModel() {

    private var state: JokeOfDayScreenState by mutableStateOf(value = JokeOfDayScreenState.Idle)
    val currentState: JokeOfDayScreenState
        get() = state

    fun fetchJoke() {
        if (state is JokeOfDayScreenState.Loading) {
            return
        }

        viewModelScope.launch {
            state = try {
                state = JokeOfDayScreenState.Loading
                val joke = jokeService.fetchJoke()
                JokeOfDayScreenState.Success(joke)
            } catch (e: Exception) {
                JokeOfDayScreenState.Error(e)
            }
        }
    }
}
```

2

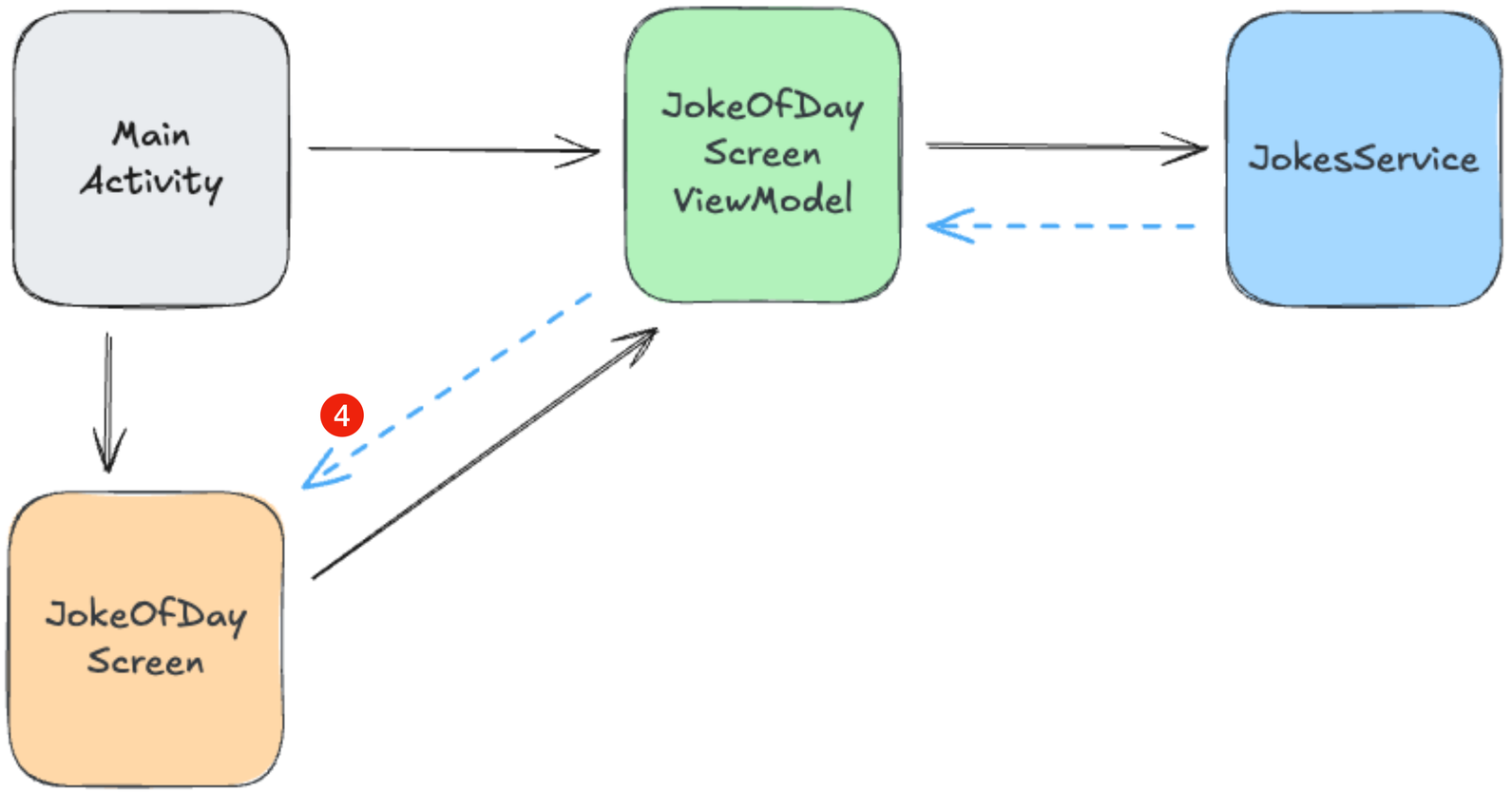
Joke Of Day (3)


Main Activity
JokeOfDay Screen ViewModel
JokesService
3
JokeOfDay Screen

JokeOfDayScreenViewModel (3)

```
class JokeOfDayScreenViewModel(val jokeService: JokesService) : ViewModel() {

    private var state: JokeOfDayScreenState by mutableStateOf(value = JokeOfDayScreenState.Idle)
    val currentState: JokeOfDayScreenState
        get() = state

    fun fetchJoke() {
        if (state is JokeOfDayScreenState.Loading) {
            return
        }

        viewModelScope.launch {
        3 state = try {
                state = JokeOfDayScreenState.Loading
                val joke = jokeService.fetchJoke()
                JokeOfDayScreenState.Success(joke)
            } catch (e: Exception) {
                JokeOfDayScreenState.Error(e)
            }
        }
    }
}
```

Joke Of Day (4)


Main Activity
JokeOfDay Screen ViewModel
JokesService
4
JokeOfDay Screen

JokeOfDayScreenViewModel (4)

```
class JokeOfDayScreenViewModel(val jokeService: JokesService) : ViewModel() {

    private var state: JokeOfDayScreenState by mutableStateOf(value = JokeOfDayScreenState.Idle)
    val currentState: JokeOfDayScreenState
        get() = state

    fun fetchJoke() {
        if (state is JokeOfDayScreenState.Loading) {
            return
        }

        viewModelScope.launch {
            state = try {
                state = JokeOfDayScreenState.Loading
                val joke = jokeService.fetchJoke()
                JokeOfDayScreenState.Success(joke)
            } catch (e: Exception) {
                JokeOfDayScreenState.Error(e)
            }
        }
    }
}
```

Documentação

- ViewModel em Android
  - https://developer.android.com/topic/libraries/architecture/viewmodel
- ViewModel Scoping
  - https://developer.android.com/topic/libraries/architecture/viewmodel/viewmodel-apis
  - https://developer.android.com/topic/libraries/architecture/coroutines#viewmodelscope

instituto
superior de
engenharia
de lisboa

POLITÉCNICO
DE LISBOA
POLYTECHNIC
UNIVERSITY OF LISBON

View Model

PDM - Programação para Dispositivos Móveis

Paulo Pereira
paulo.pereira@isel.pt

PRO
DIGI