instituto
superior de
engenharia
de lisboa

POLITÉCNICO
DE LISBOA
POLYTECHNIC
UNIVERSITY OF LISBON

Modelo de Concorrência

PDM - Programação para Dispositivos Móveis

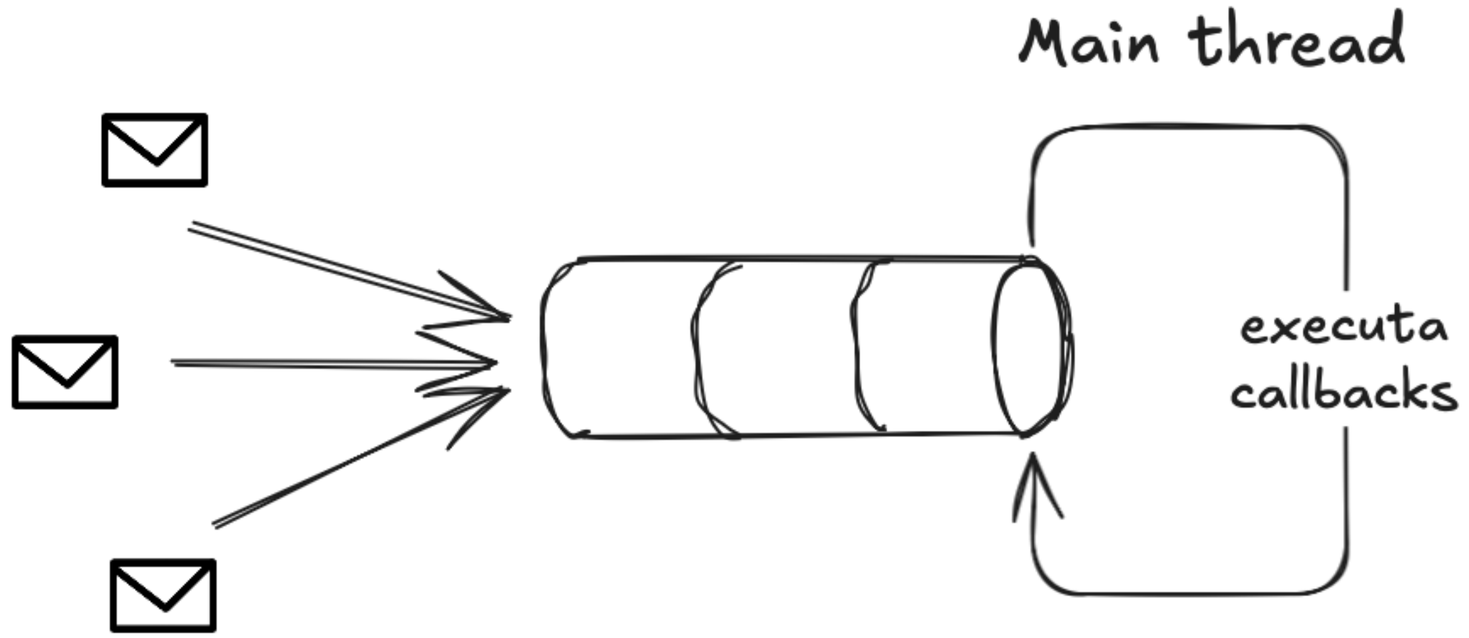
Paulo Pereira
paulo.pereira@isel.pt

PRO
DIGI

Execução em Android (1)

- Em Android existe uma thread “especial”, a Main thread (a.k.a. UI thread)
- A Main thread executa:
  - **Todos** os handlers de evento da UI
  - **Todos** os callbacks de ciclo de vida
  - Genericamente, callbacks Android são executados nessa thread (existem excepções devidamente documentadas)

Execução em Android (2)



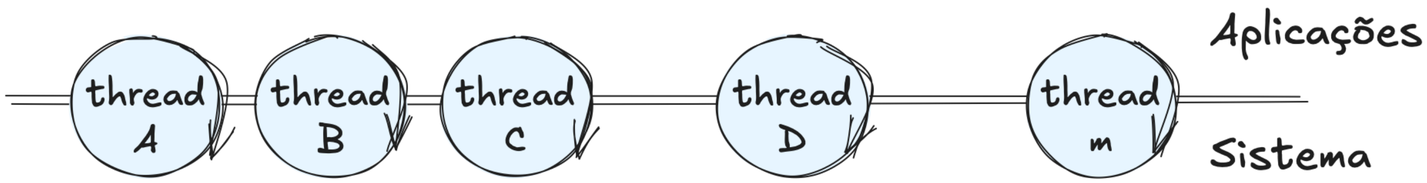
Consequências

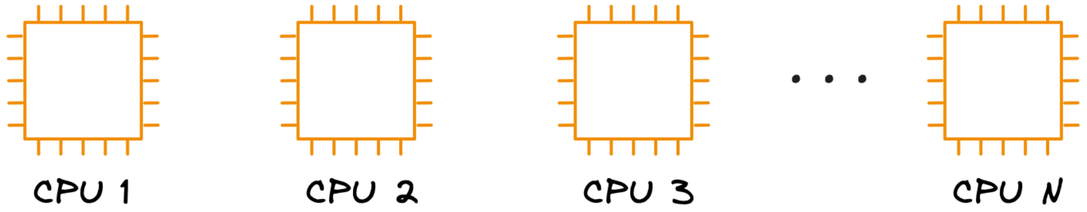
- Os efeitos do callback são observados pelos seguintes
- A execução do callback atrasa **TODOS** os outros
  - **TEM** de ser breve, ou
  - A experiência de utilização (UX) será prejudicada
- E se for preciso demorar? (i.e. I/O ou computações exigentes)

Concorrência em Android

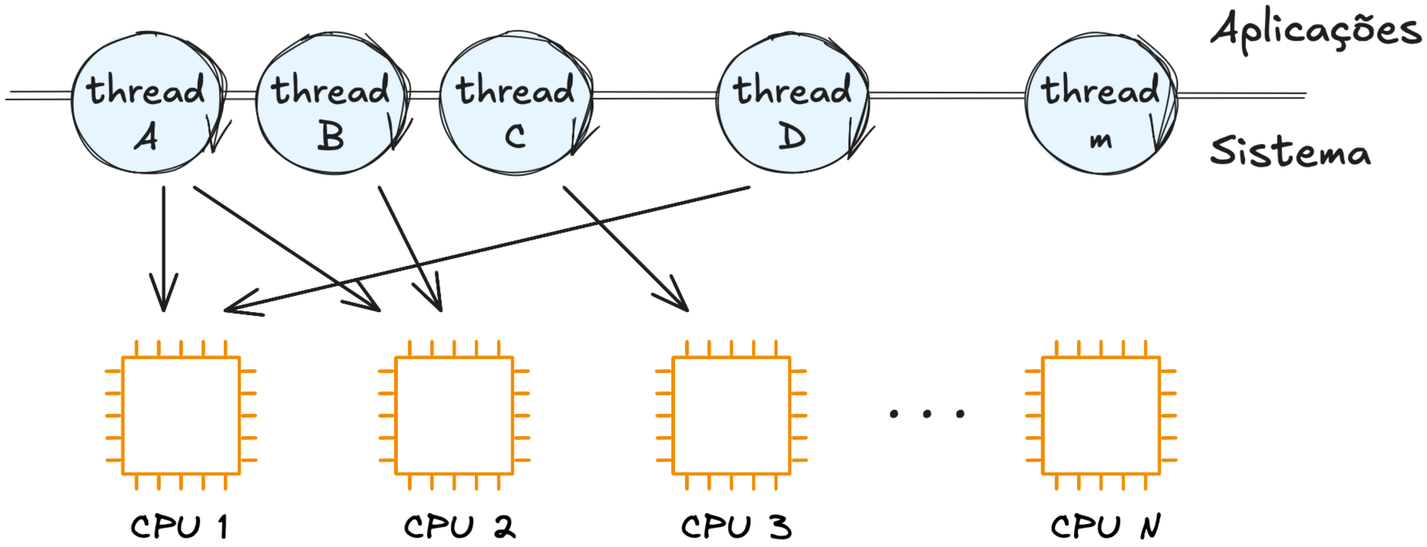
- Baseado no modelo de concorrência Kotlin
- Principais elementos:
  - Threads
  - Coroutines
  - Dispatchers
  - Scopes

Participantes

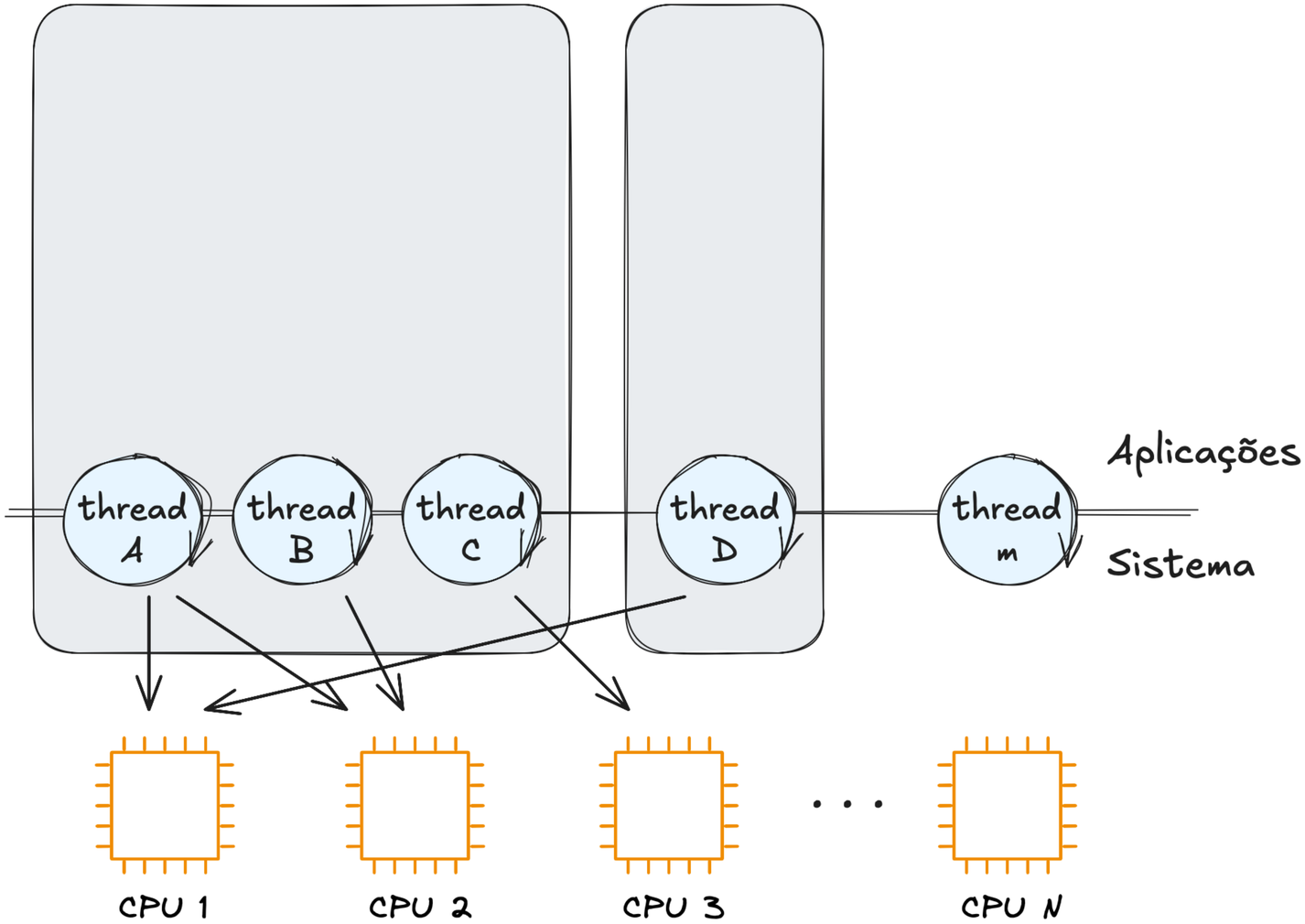




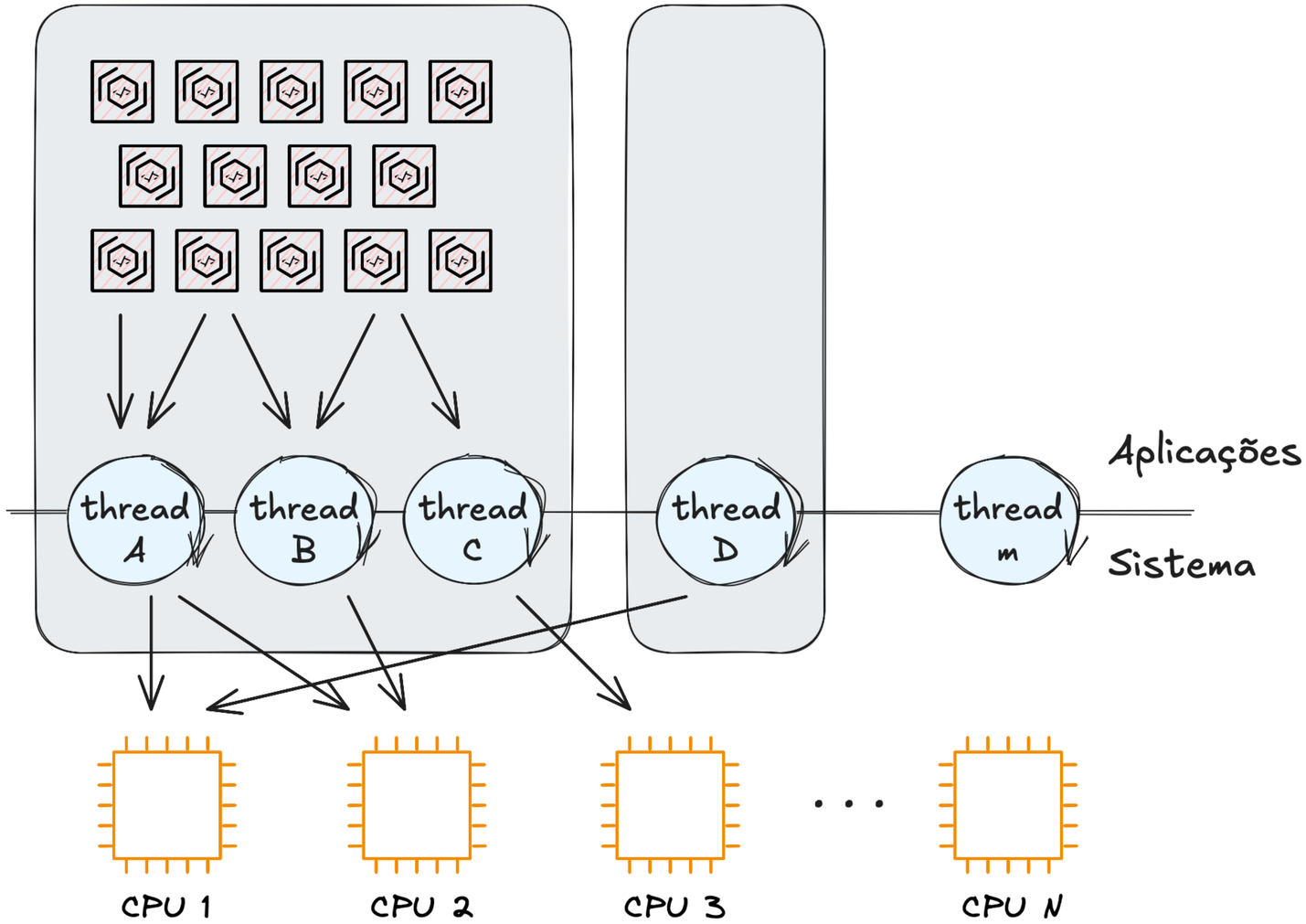
Participantes



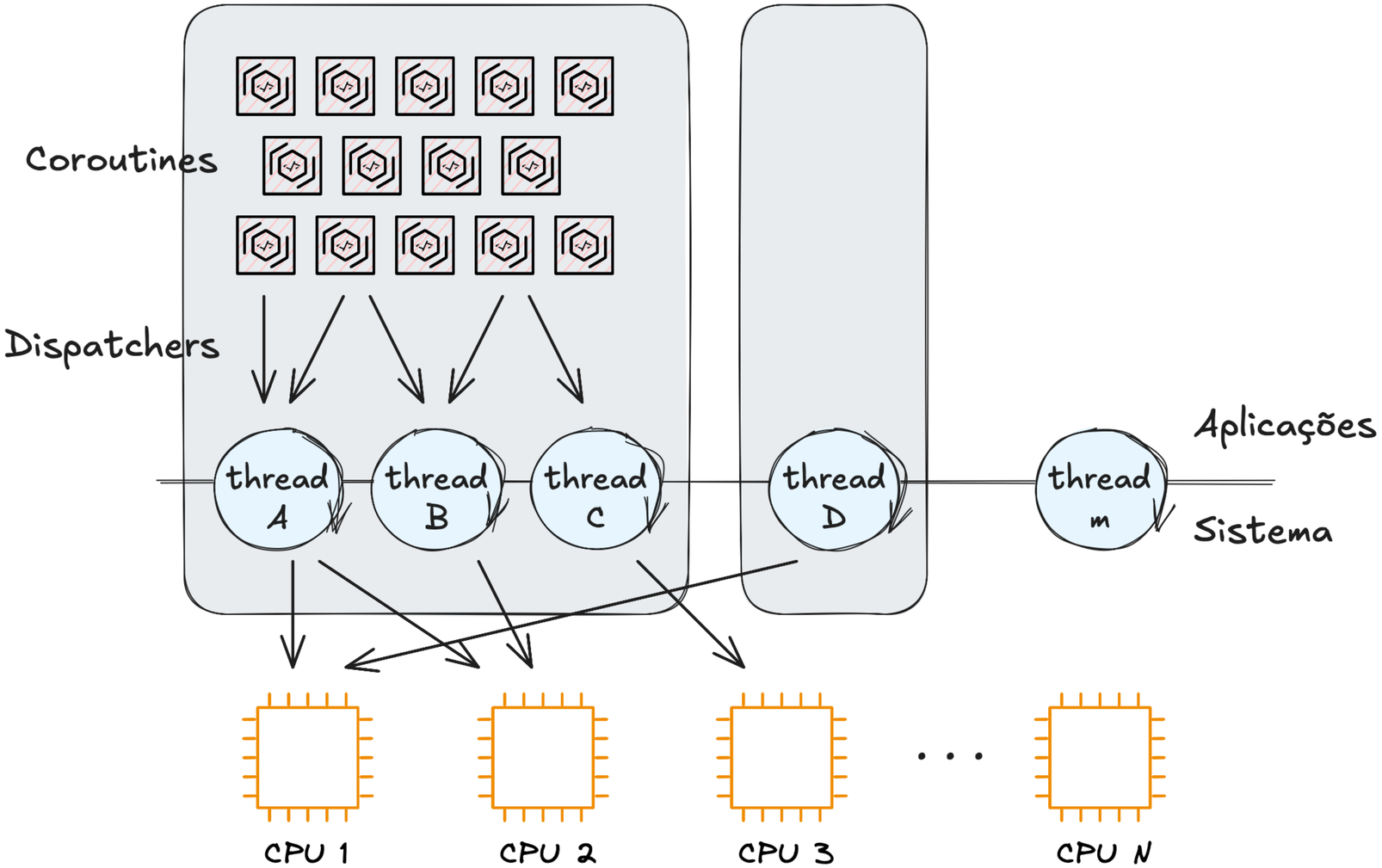
Participantes



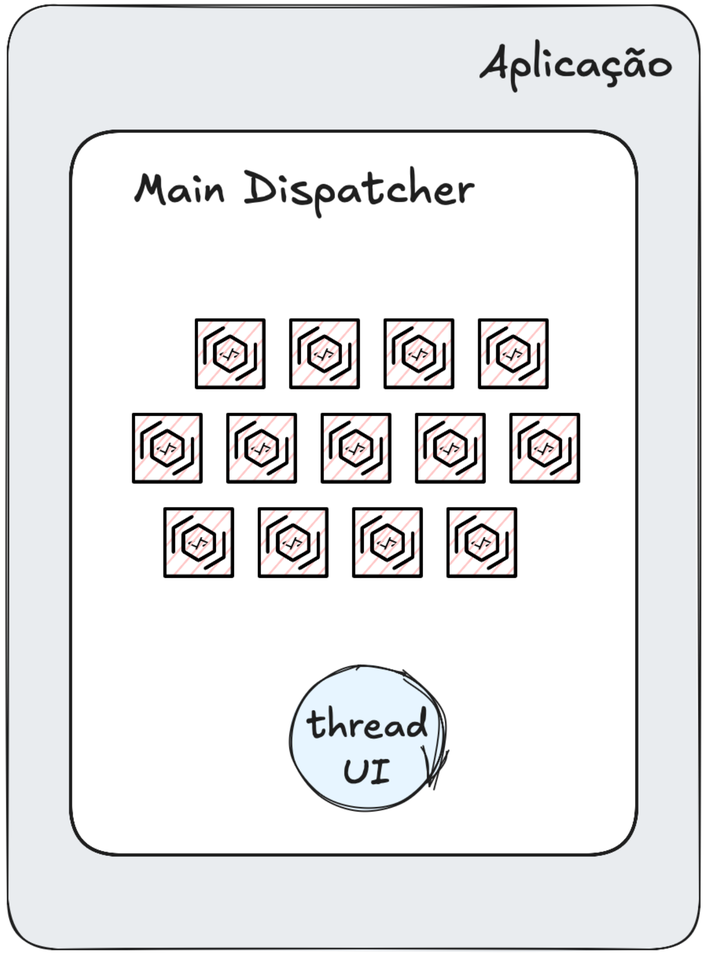
Participantes



Participantes

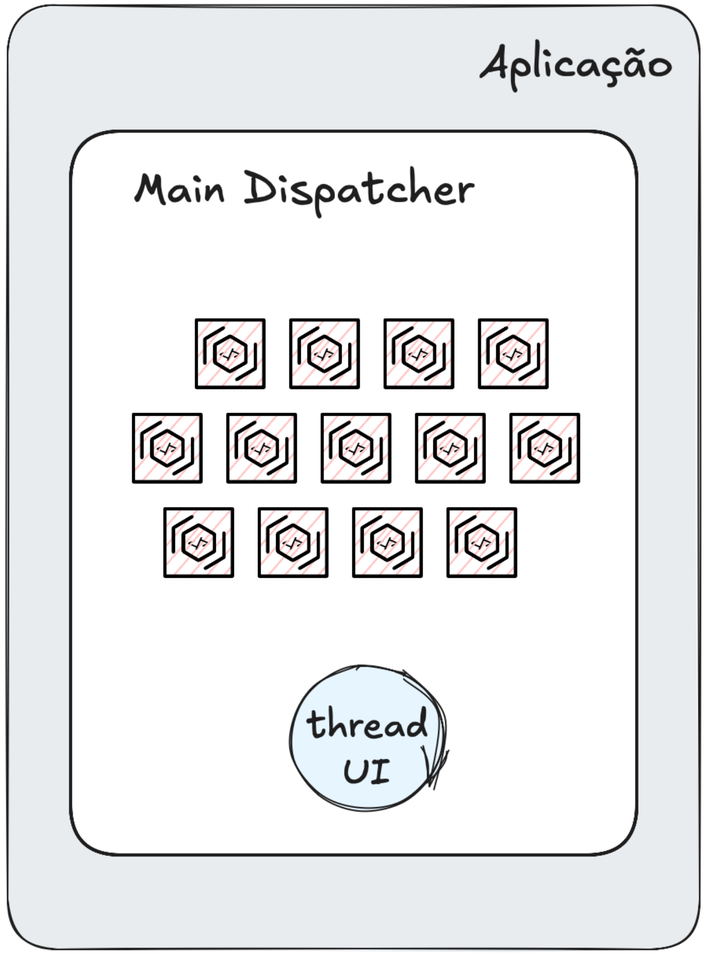


Modelo simplificado (1)



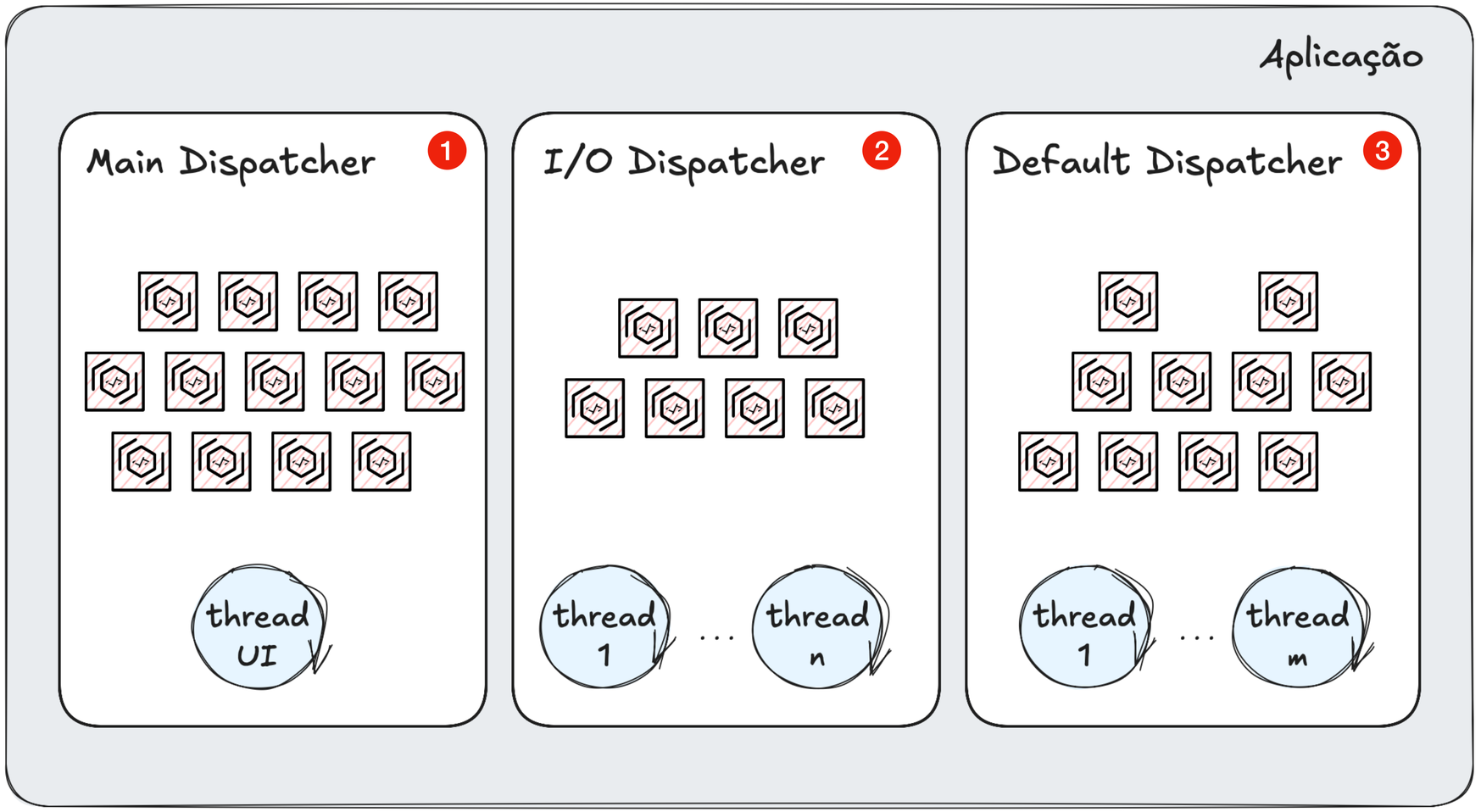
- A ter em conta com este modelo:
  - I/O bloqueante é proibido
  - Computação intensiva é proibida
    - Escalonamento de coroutines é cooperativo

Modelo simplificado (2)



- I/O é proibido?
  - Não! Apenas **I/O bloqueante**
  - I/O é representado através de funções *suspend*
- Funções *suspend* representam operações com pontos de suspensão

Participantes



Concorrência em Android

- Baseado no modelo de concorrência Kotlin
- Principais elementos:
  - Threads
  - Coroutines
  - Dispatchers
  - Scopes

Scopes

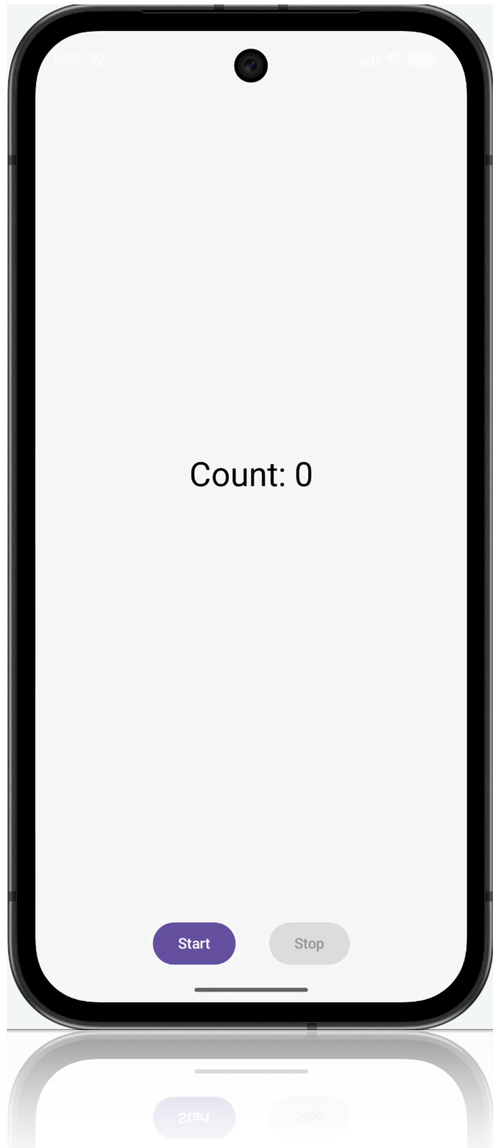
- Suporte para **Concorrência Estruturada**
- As *coroutines* estão sujeitas ao *scope* em que são lançadas
  - Lançar coroutines exige o uso de um *scope*
  - *Scope* termina => *coroutines* activas são canceladas
  - **Importante:** cancelamento é cooperativo

Exemplos de Scopes

- Propriedade **lifecycleScope** de **ComponentActivity**
  - Vinculado ao ciclo de vida da Activity
- Propriedade **viewModelScope** de **ViewModel**
  - Vinculado ao ciclo de vida do View Model
- Retorno da função **rememberCoroutineScope()**
  - Vinculado à composição onde a função é chamada

Exemplo (1)

```
@Composable
fun CoroutinesFunScreen() {
    var count by remember { mutableIntStateOf(0) }
    var job by remember { mutableStateOf<Job?>(null) }
    val scope = rememberCoroutineScope()
    Column( … ) {
        Box(
            modifier = Modifier.fillMaxSize().weight(1f),
            contentAlignment = Alignment.Center
        ) {
            Text(text = "Count: $count")
        }
        Row {
            Button(                          1
                onClick = { … },
                enabled = job == null,
            ) {
                Text("Start")
            }
            Button(                          2
                onClick = { … },
                enabled = job != null,
            ) {
                Text("Stop")
            }
        }
    }
}
```



Exemplo (2)

1

```
onClick = {
    if (job == null) {
        job = scope.launch {
            while (isActive) {
                delay(500L)
                count += 1
            }
        }
    }
}
```

2

```
onClick = {
    job?.cancel()
    job = null
},
```

```
while (isActive) {
    delay(500L)
    count += 1
}
```

```
while (true) {
    delay(500L)
    count += 1
}
```

Exemplo mais realista

1

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

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

    var currentState by mutableStateOf<JokeOfDayScreenState>(value = Idle)

    fun fetchJoke() {
        // ...
        viewModelScope.launch {
            currentState = JokeOfDayScreenState.Loading
            currentState = try {
                jokeService.fetchJoke().let { JokeOfDayScreenState.Success(it) }
            } catch (e: Exception) {
                JokeOfDayScreenState.Error(e)
            }
        }
    }
}
```

Documentação

- Modelo de concorrência Kotlin
  - https://kotlinlang.org/docs/coroutines-guide.html
- Modelo de concorrência Android
  - https://developer.android.com/kotlin/coroutines
  - https://developer.android.com/kotlin/coroutines/coroutines-adv
- Aulas de Programação Concorrente na LEIC do ISEL
  - https://www.youtube.com/watch?v=wGNDe1QB0LM&list=PL8XxoCaL3dBhX9Kqt_BfAE23D4zYqgLdN

instituto
superior de
engenharia
de lisboa

POLITÉCNICO
DE LISBOA
POLYTECHNIC
UNIVERSITY OF LISBON

Modelo de Concorrência

PDM - Programação para Dispositivos Móveis

Paulo Pereira
paulo.pereira@isel.pt

PRO
DIGI