Juntando as peças
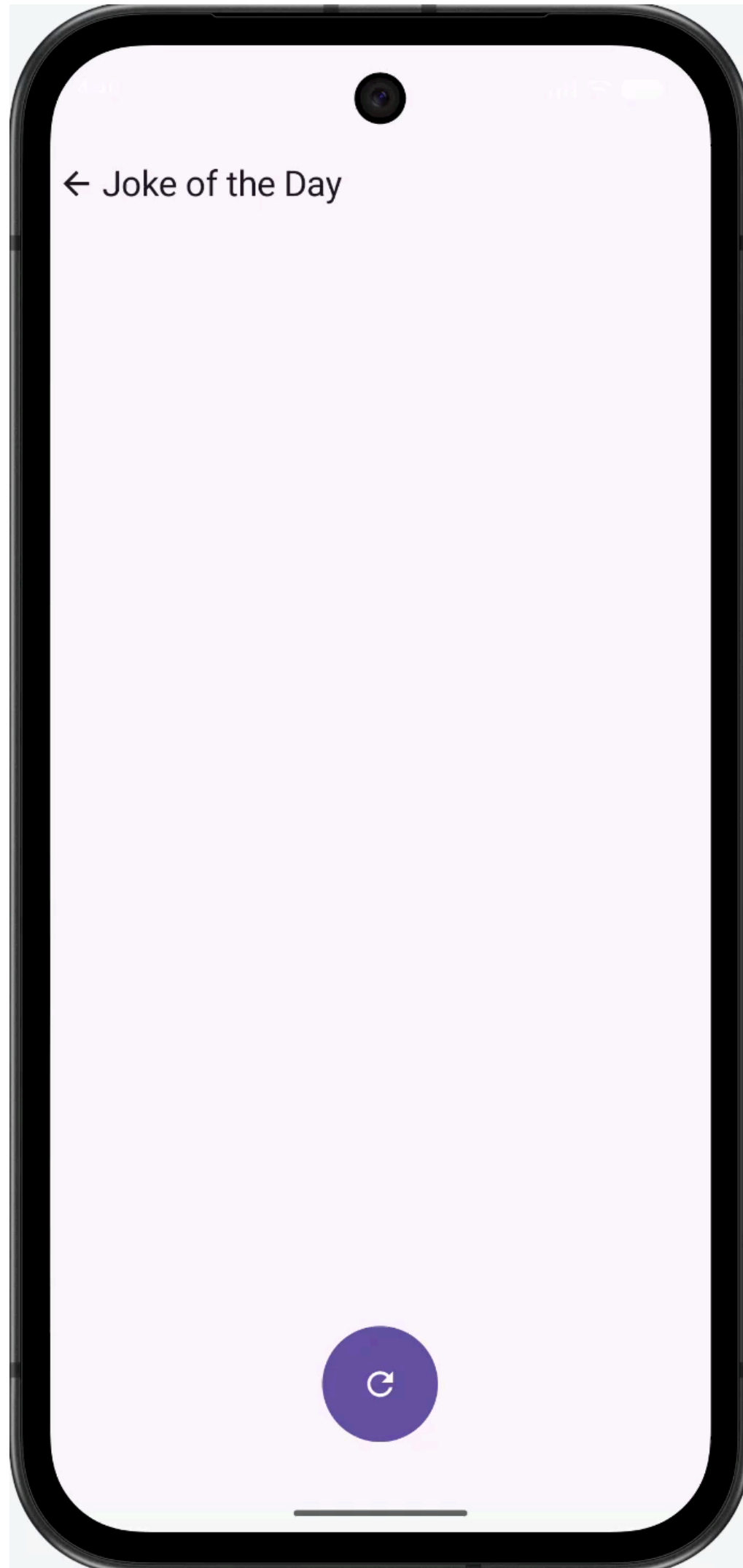
PDM - Programação para Dispositivos Móveis

Paulo Pereira

paulo.pereira@isel.pt

Joke Of Day

- 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

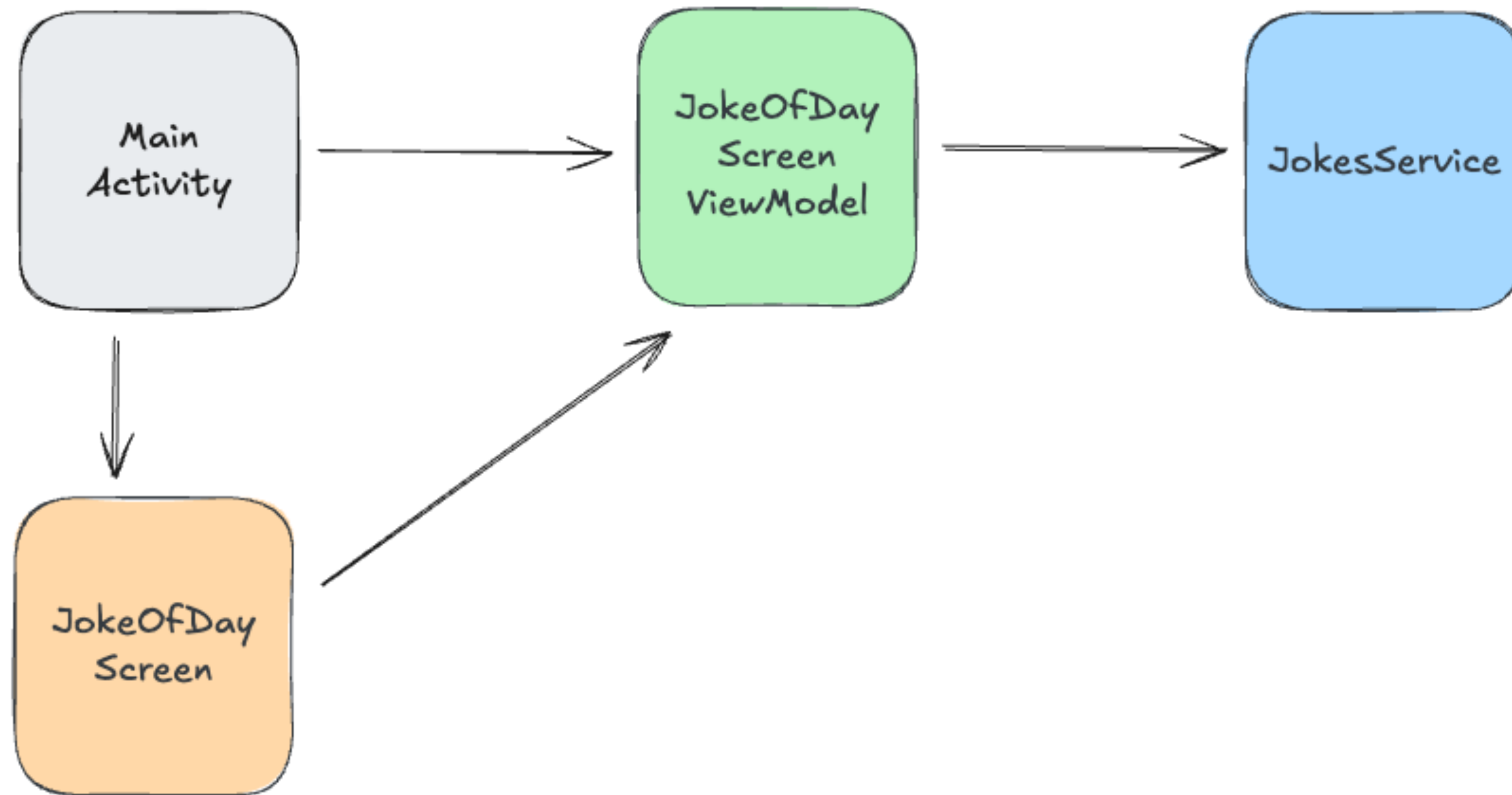 - Usa a API https://icanhazdadjoke.com

Objectivos

- Identificar o papel de cada elemento da solução

- Descrever o desenho da solução

    - Resolução de dependências (i.e. ligações entre peças)

    - Modelação da UI como máquina de estados

Joke Of Day

Resolução de dependências (1)

```kotlin
class JokeOfDayApplication : Application() {

    private val httpClient: HttpClient by lazy {
        HttpClient {
            install(plugin = ContentNegotiation) {
                json(
                    json = Json {
                        prettyPrint = true; isLenient = true; ignoreUnknownKeys = true
                    }
                )
            }
        }
    }


    val jokeService: JokesService by lazy {
        IcanhazDadJokes(client = httpClient)
    }
}
```

Resolução de dependências (2)

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <application
        android:name=".JokeOfDayApplication" 1
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:theme="@style/Theme.JokeOfDay">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

MainActivity

```kotlin
class MainActivity : ComponentActivity() {

    private val viewModel: JokeOfDayScreenViewModel by viewModels {  ②
        JokeOfDayScreenViewModel.getFactory(
            service = (application as JokeOfDayApplication).jokeService  ③
        )
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            JokeOfDayScreen(viewModel = viewModel, onBackIntent = { finish() })
        }  ①
    }
}
```

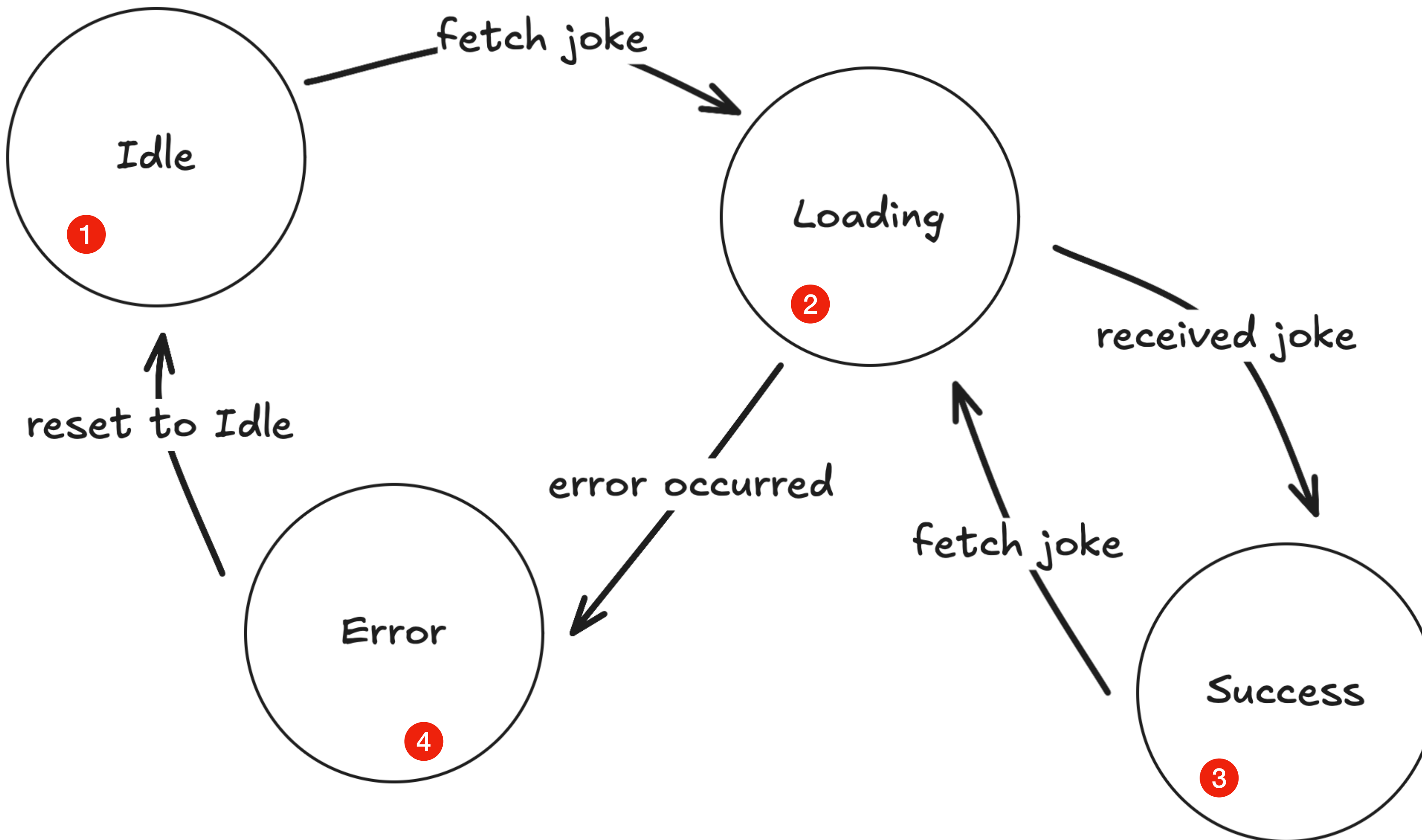# JokeOfDayScreenViewModel

```kotlin
class JokeOfDayScreenViewModel(val jokeService: JokesService) : ViewModel() {
                                                            ③
    companion object {
        fun getFactory(service: JokesService) = object : ViewModelProvider.Factory {
            @Suppress("UNCHECKED_CAST")    ①
            override fun <T : ViewModel> create(modelClass: Class<T>): T =
                if (modelClass.isAssignableFrom(JokeOfDayScreenViewModel::class.java)) {
                    JokeOfDayScreenViewModel(jokeService = service) as T
                }                                                      ②
                else throw kotlin.IllegalArgumentException("Unknown ViewModel class")
        }
    }

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

    fun fetchJoke() { /* Implementação aqui */ }
    fun resetToIdle() { /* Implementação aqui */ }
}
```
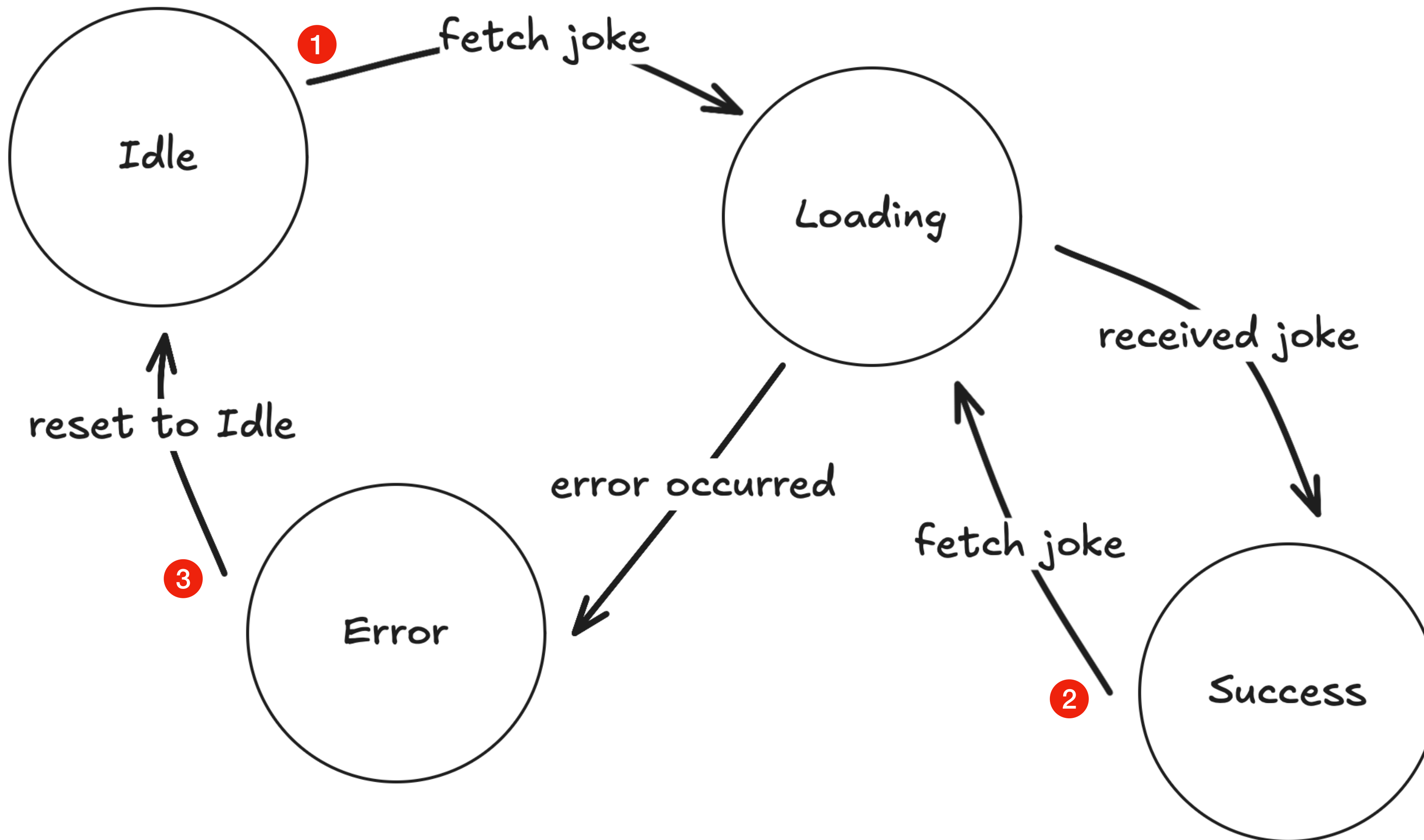
UI como máquina de estados (1)

Estado actual (currentState)

```kotlin
sealed interface JokeOfDayScreenState {
    data object Idle : JokeOfDayScreenState
    data object Loading : JokeOfDayScreenState
    data class Success(val joke: Joke) : JokeOfDayScreenState
    data class Error(val exception: Throwable) : JokeOfDayScreenState
}

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

    companion object { /* Implementação aqui */ }

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

    fun fetchJoke() { /* Implementação aqui */ }
    fun resetToIdle() { /* Implementação aqui */ }
}
```

UI como máquina de estados (2)

Transições de estado

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

    /* Implementação aqui */

(1) fun fetchJoke() {
        if (state is JokeOfDayScreenState.Idle || state is JokeOfDayScreenState.Success) {
            viewModelScope.launch {
                state = try {
                    state = JokeOfDayScreenState.Loading (2)
            (3) JokeOfDayScreenState.Success(joke = jokeService.fetchJoke())
                } catch (e: Exception) {
                    JokeOfDayScreenState.Error(e)
                }          (4)
            }
        }
    }

(5) fun resetToIdle() {
        if (state is JokeOfDayScreenState.Error) {
            state = JokeOfDayScreenState.Idle
        }
    }
}
```

Renderização do estado

```kotlin
@Composable
fun JokeOfDayScreen(viewModel: JokeOfDayScreenViewModel, onBackIntent: () -> Unit) {
    when (val state = viewModel.currentState) {
        is JokeOfDayScreenState.Idle -> IdleView(                    ❶
            onFetchRequested = { viewModel.fetchJoke() },
            onBackRequested = onBackIntent
        )

        is JokeOfDayScreenState.Loading -> LoadingView()  ❷

        is JokeOfDayScreenState.Success -> SuccessView(   ❸
            joke = state.joke,
            onFetchRequested = { viewModel.fetchJoke() },
            onBackRequested = onBackIntent
        )

        is JokeOfDayScreenState.Error -> ErrorAlert(      ❹
            title = R.string.jokes_error_api_title,
            message = R.string.jokes_error_could_not_reach_api,
            buttonText = R.string.jokes_error_retry_button_text,
            onDismiss = { viewModel.resetToIdle() }
        )
    }
}
```

Joke Of Day

- Aplicação simplificada para demonstrar os conceitos estudados

- Implementação encontra-se aqui:

  - https://github.com/palbp/pdm.prodigi/tree/main/demos/JokeOfDay

  - Usa a API https://icanhazdadjoke.com

Juntando as peças

PDM - Programação para Dispositivos Móveis

Paulo Pereira

paulo.pereira@isel.pt