ANDROID DEVELOPMENT

Class 6: Networking & Async

Fetching data from the internet without freezing the UI

Instructor: Mark Joseli

Today's Goal

Connect your app to the **World Wide Web.**

GET data (JSON) from APIs.

```
{
    "id": 1,
    "title": "Inception",
    "rating": 8.8
}
```

Use **Coroutines** for background tasks.

Handle **Loading** & **Error** states.

Loading

Error

Theory: HTTP & JSON

HTTP Methods

- **GET:** Retrieve data (e.g., fetch movie list).
- **POST:** Send data (e.g., login, post tweet).
- **Response Codes:** 200 (OK), 404 (Not Found), 500 (Server Error).

JSON

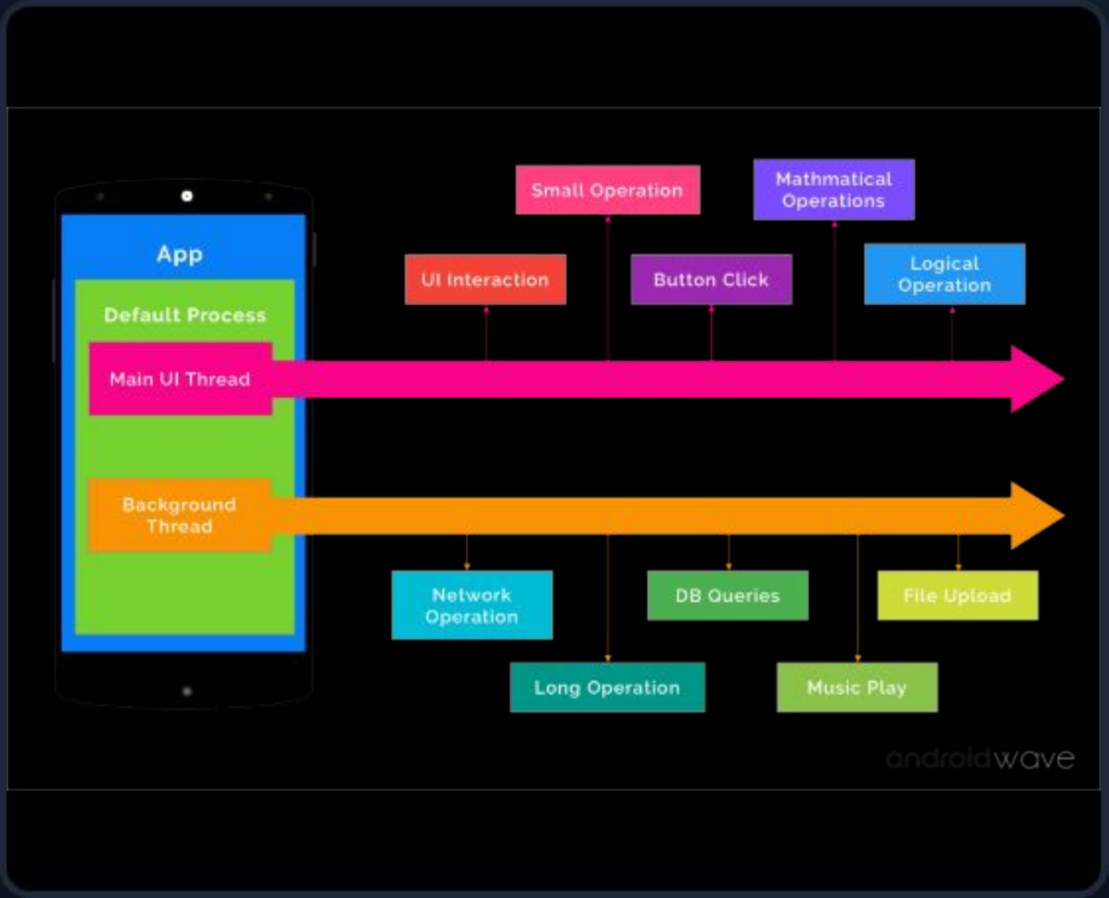
JavaScript Object Notation. The language of the web.

```
{ "id": 1, "title": "Inception", "rating":
8.8 }
```

Threading: Don't Freeze!

The Main Thread (UI Thread): Responsible for drawing every frame (60fps). If you do heavy work here, the app freezes (ANR).

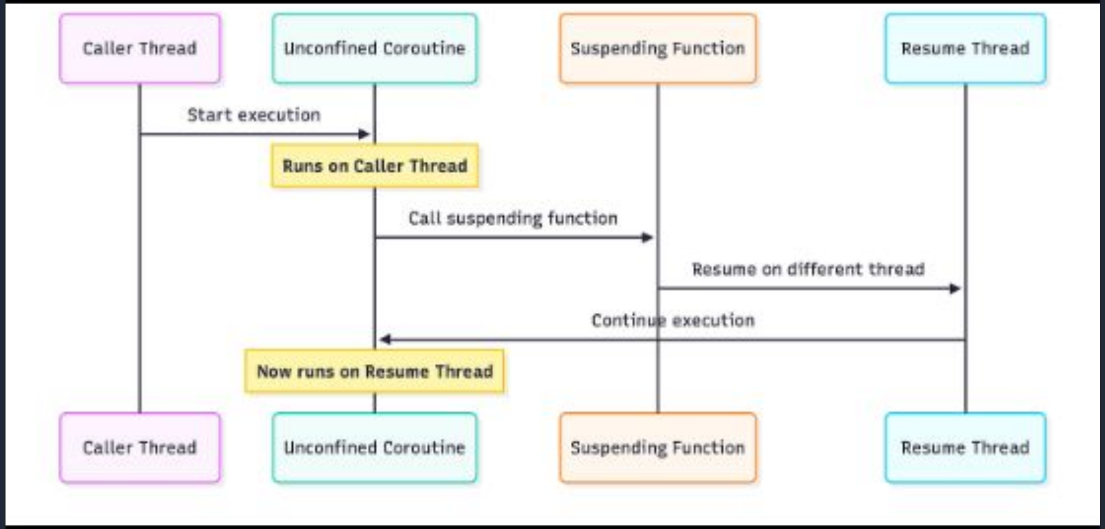
Background Thread (IO): Where network requests and database operations must happen.



Tools: Coroutines

Kotlin's solution for asynchronous code. They are "lightweight threads".

The suspend keyword marks a function that can pause execution without blocking the thread.



Tools: Retrofit

A type-safe HTTP client for Android. It turns your HTTP API into a Java/Kotlin interface.

```
interface ApiService { @GET("movies/popular")
suspend fun getMovies(): List }
```

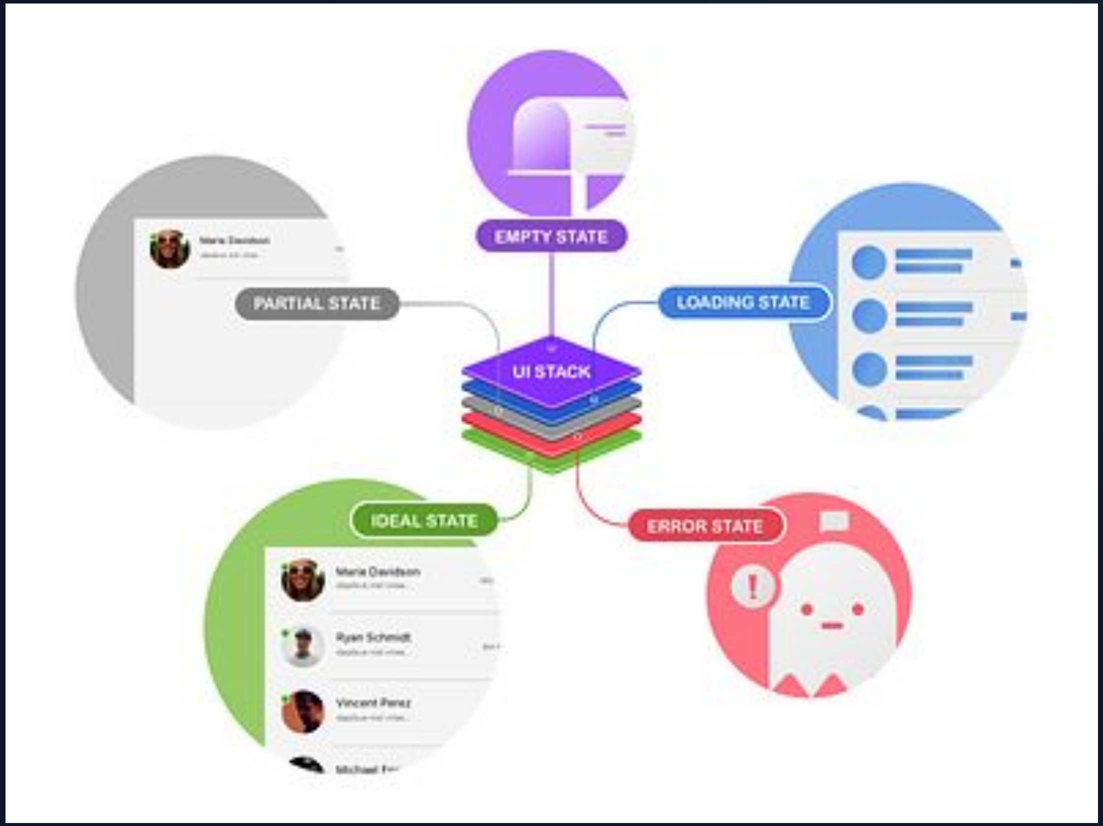
Retrofit handles the JSON parsing and threading setup for you automatically.

Interface → Network Call

Practice: UI States

Network calls take time and can fail. We need to represent these states.

```
sealed class UiState { object Loading : UiState()
data class Success(val data: List) : UiState()
      Error(val msg: String) : UiState() }
```



Handling in Compose

Use a simple `when` nent to switch the UI.

```
@Composable fun Screen(state: UiState) { when
(state) { is Loading → CircularProgressIndicator ()
is Error → Text("Error: ${state.msg}" ) is Success
→ MovieList(state.data) } }
```

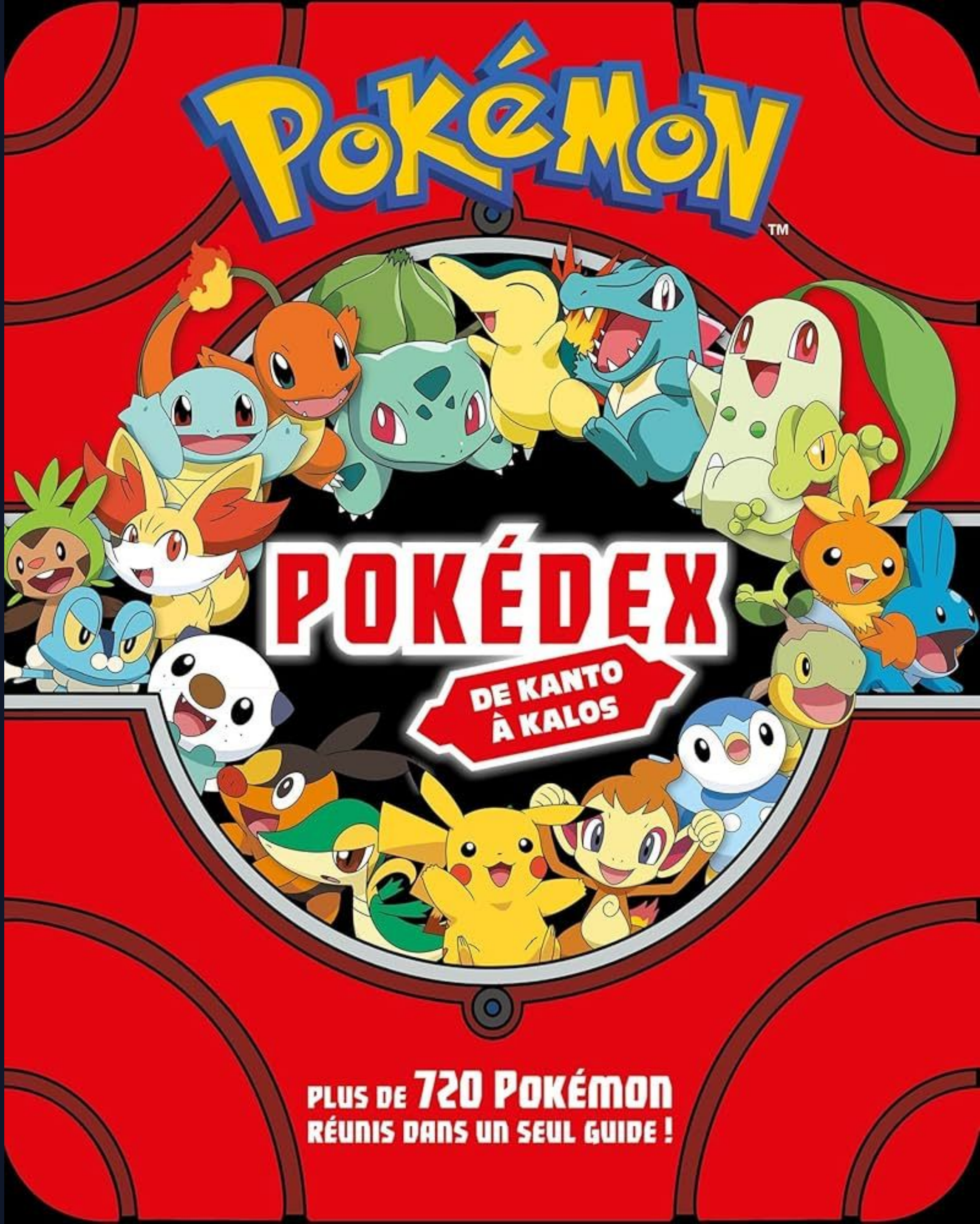
One Source of

Truth

Lab: Build a Pokedex

Build an app that fetches live Pokemon data using the **PokeAPI**.

- **Step 1:** Define with Retrofit.
- **Step 2:** Create ViewModel to fetch data in IO thread.
- **Step 3:** Build UI to display Pokemon sprites and names.



Questions?

Gotta catch 'em all!

Image Sources

https://miro.medium.com/1*PcvnALAfcav5PoG41hhSZg.png

Source: medium.com

https://miro.medium.com/1*I5KrSlqRpU64yd0vZkyLSw.png

Source: medium.com