Application architecture

Applikationsudvikling

NIFR - 2023

Make it work. Make it right. Make it fast

- Kent Beck

Today: Make it righter

Agenda

Application Architecture

- Software Architecture
- Layered Architecture
- Common architectural principles in layered architecture
  - Unidirectional Data Flow
  - Seperation of concerns
  - Single source of truth
  - Drive UI from data model
- ViewModel

Objective: Technical analysis (report)

Assess how well your application adheres to the introduced principles

What changes could better your application?

Software Architecture

Why

https://earthlymission.com/how-many-lines-of-code-does-it-have/

What is (source) code?

Developer Perspective

On the surface

```
  var scrollHeight
  element.clientHeight + 0.02 * w
 window.scroll(0, scrollHeight);
}
```

Application:
Collection of source code

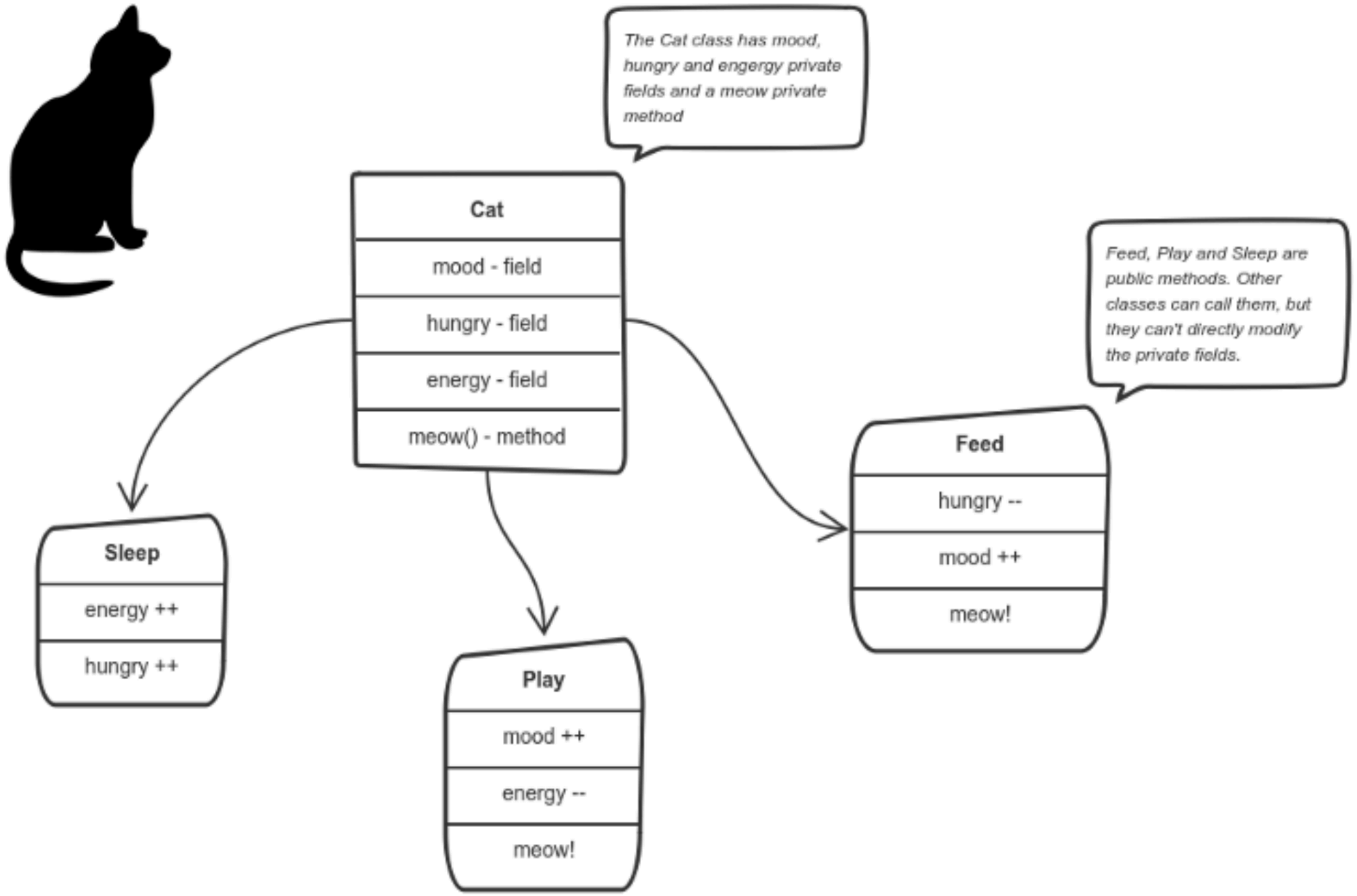
Developer Perspective

Behind the curtain

Application:
Collection of interacting components

Developer Perspective

Behind the curtain



The Cat class has mood, hungry and engergy private fields and a meow private method

Feed, Play and Sleep are public methods. Other classes can call them, but they can't directly modify the private fields.

Cat
- mood - field
- hungry - field
- energy - field
- meow() - method

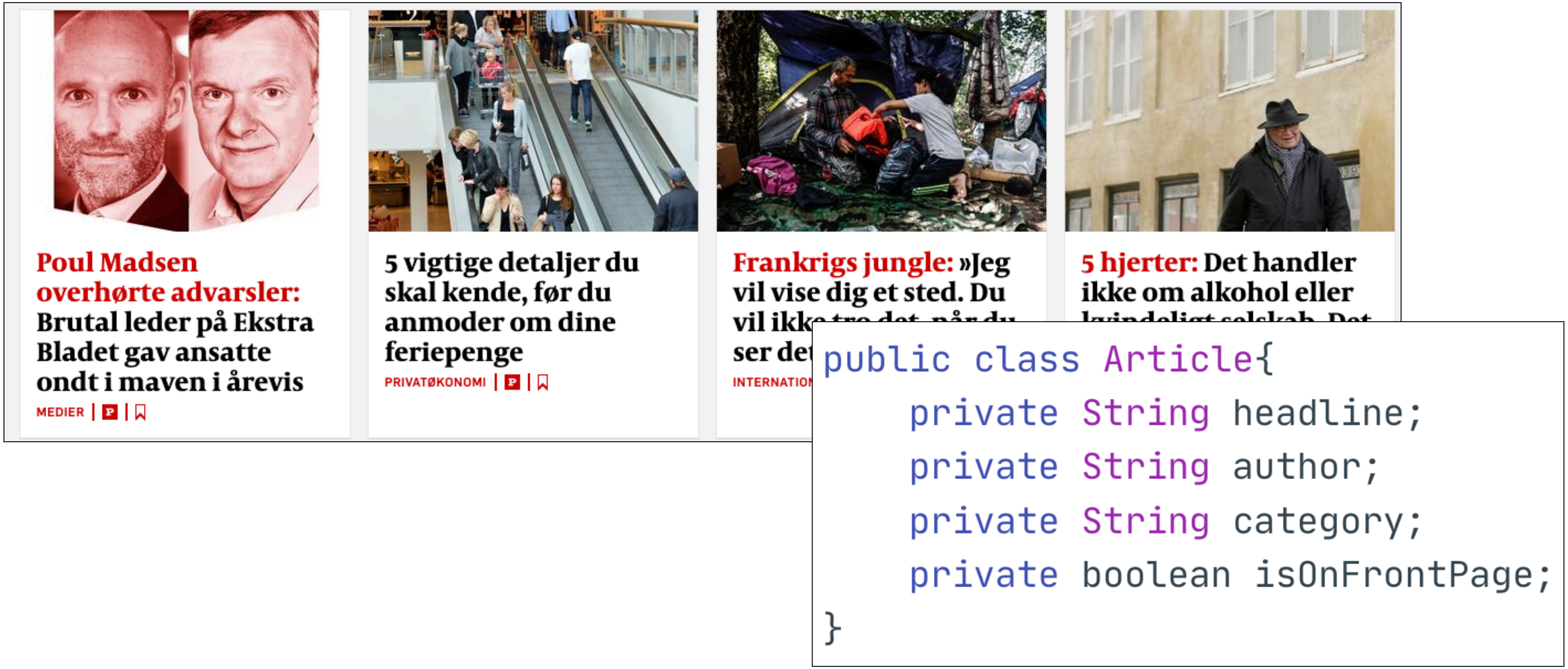
Sleep
- energy ++
- hungry ++

Play
- mood ++
- energy --
- meow!

Feed
- hungry --
- mood ++
- meow!

Developer Perspective

Behind the curtain



Poul Madsen overhørte advarsler: Brutal leder på Ekstra Bladet gav ansatte ondt i maven i årevis

MEDIER | P |

5 vigtige detaljer du skal kende, før du anmoder om dine feriepenge

PRIVATØKONOMI | P |

Frankrigs jungle: »Jeg vil vise dig et sted. Du vil ikk ser det

INTERNATIO

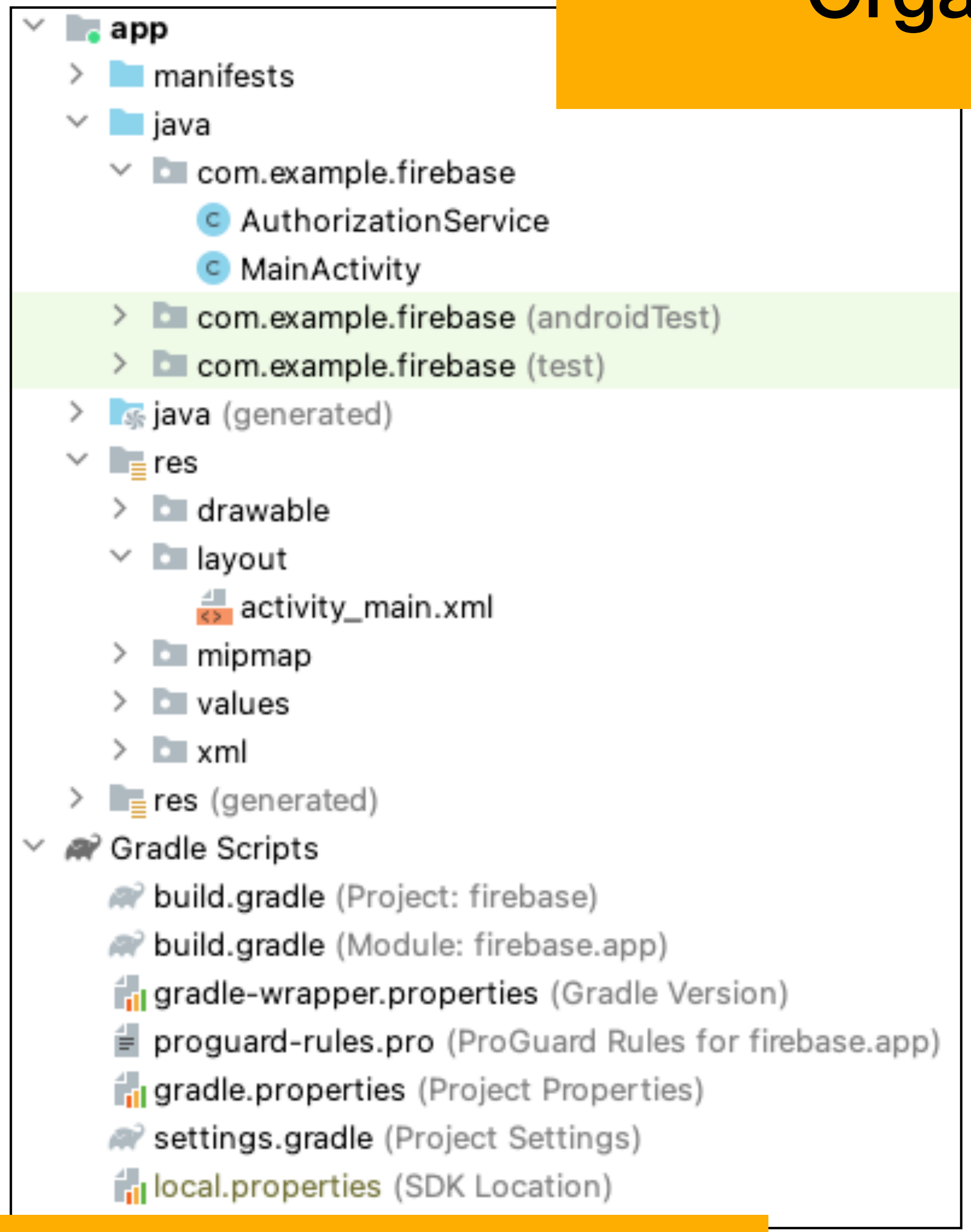
5 hjerter: Det handler ikke om alkohol eller

```
public class Article{
    private String headline;
    private String author;
    private String category;
    private boolean isOnFrontPage;
}
```

Software Architecture:

How to organise software components and enforce rules of interaction

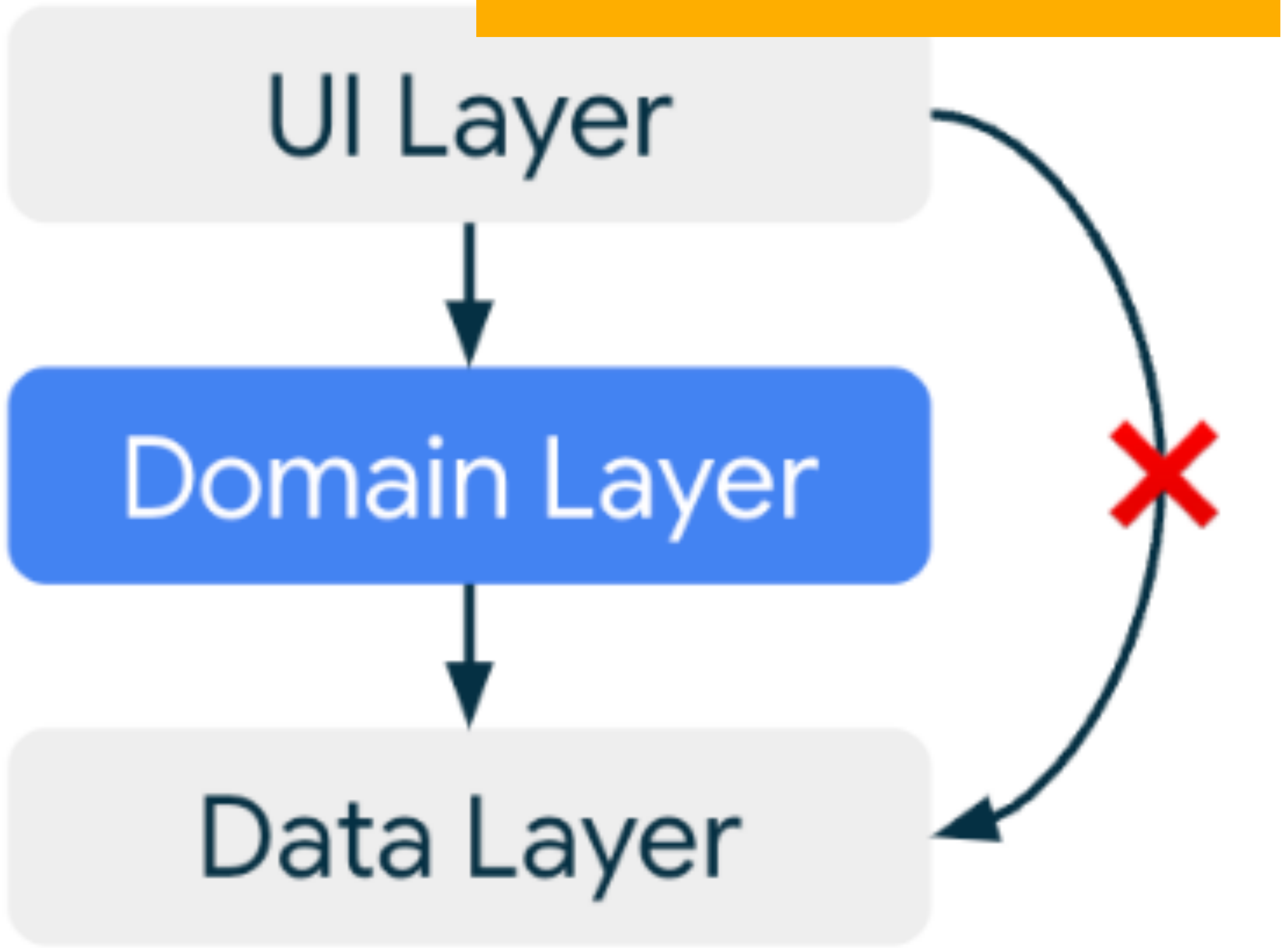
Organisation



```
app
  manifests
  java
    com.example.firebase
      AuthorizationService
      MainActivity
    com.example.firebase (androidTest)
    com.example.firebase (test)
  java (generated)
  res
    drawable
    layout
      activity_main.xml
    mipmap
    values
    xml
  res (generated)
Gradle Scripts
  build.gradle (Project: firebase)
  build.gradle (Module: firebase.app)
  gradle-wrapper.properties (Gradle Version)
  proguard-rules.pro (ProGuard Rules for firebase.app)
  gradle.properties (Project Properties)
  settings.gradle (Project Settings)
  local.properties (SDK Location)
```

Where do we put this file?

Rules of engagement



These classes should only interact with these classes

Layered Architecture

Recommended app architecture

- Seperation of concerns
- UI layer display application data
- Domain Layer handles *business logic*
- Data layer handles data (fetching, writing, CRUD)

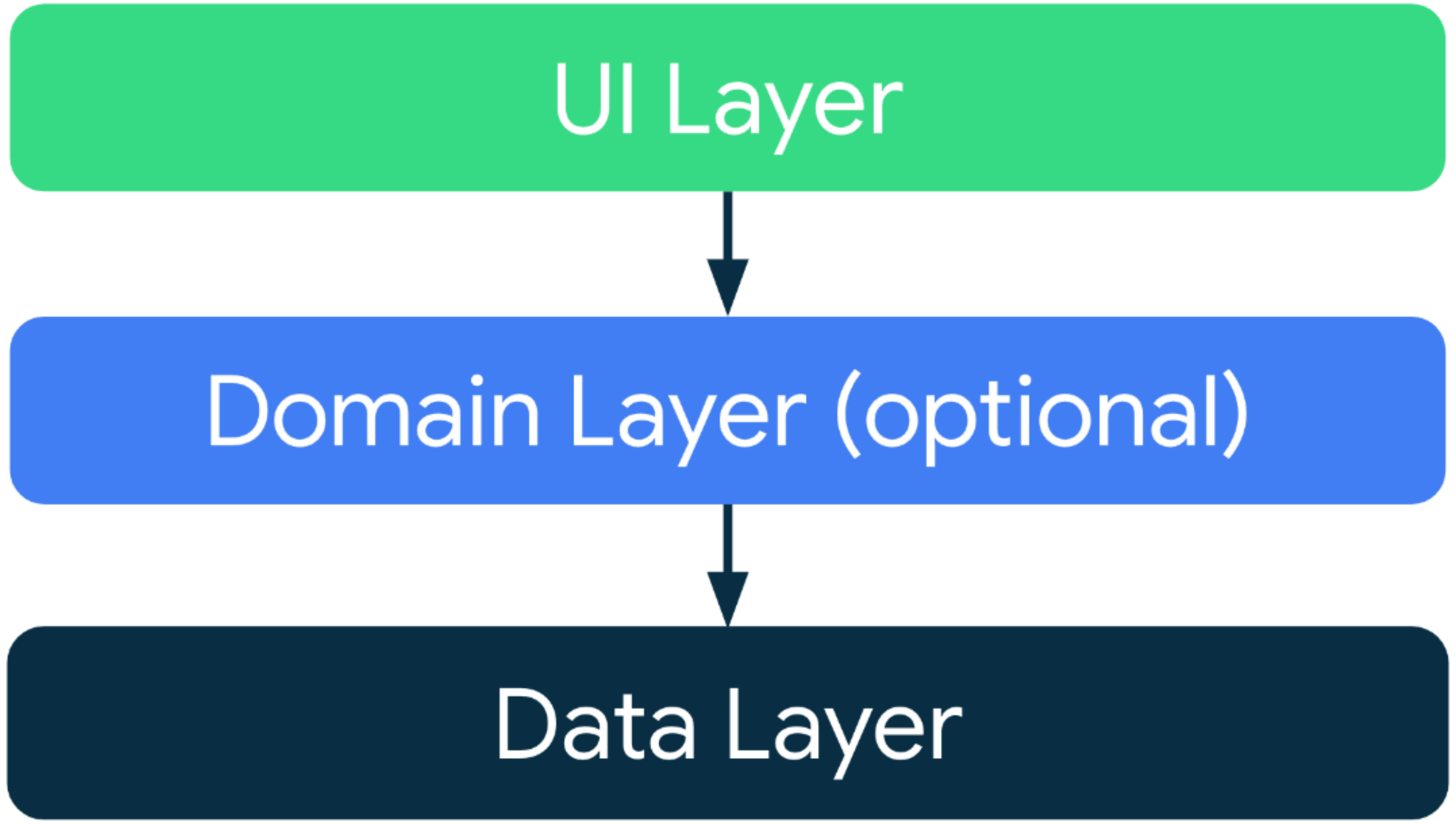


Figure 1. Diagram of a typical app architecture.

Domain/Business Layer

Definition

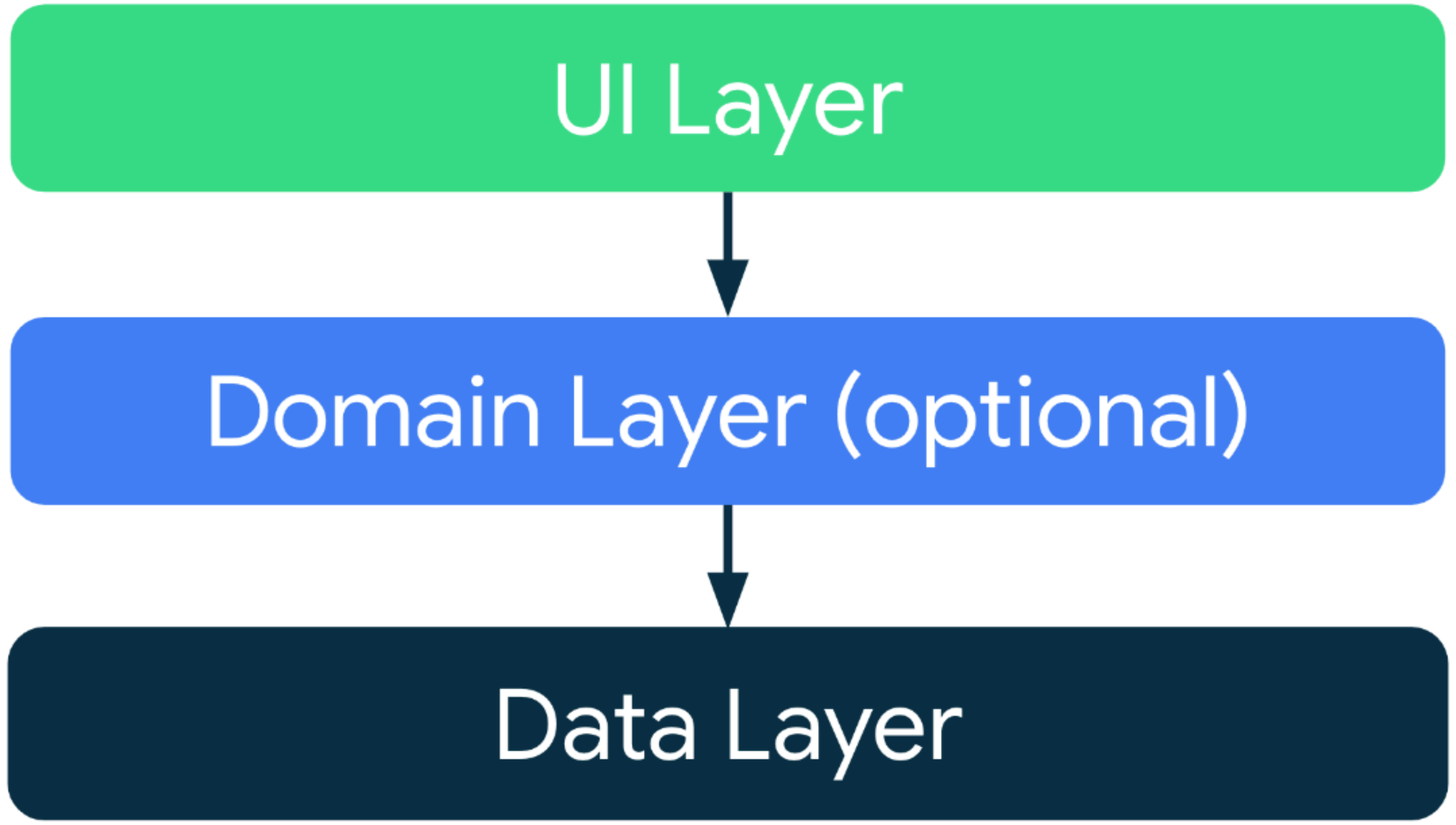
Sold out items should be marked with red font

Maximum tweet size is 280 characters

A user should always enter an address, otherwise we cannot deliver

Organisation

Application Architecture

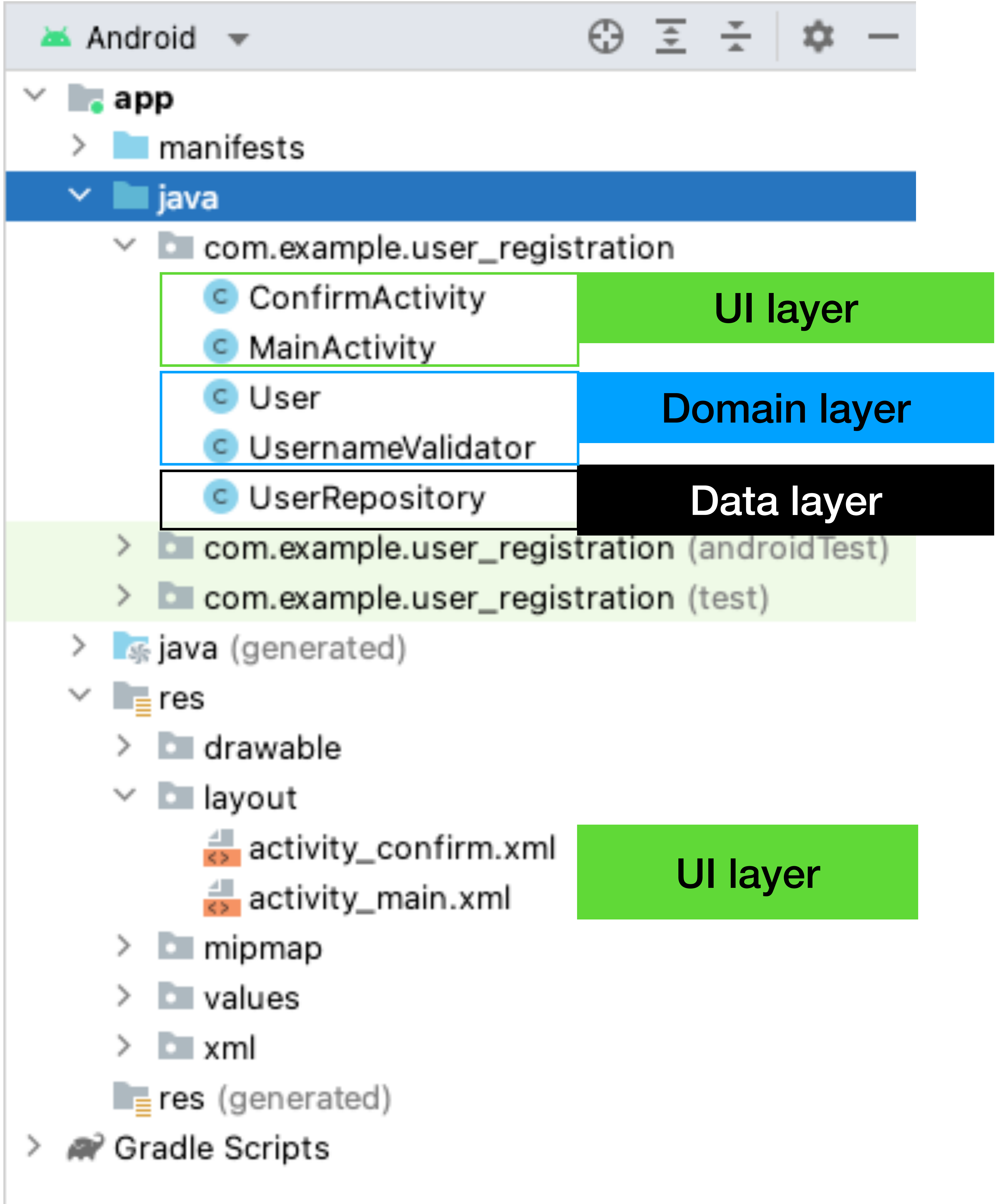


UI Layer

Domain Layer (optional)

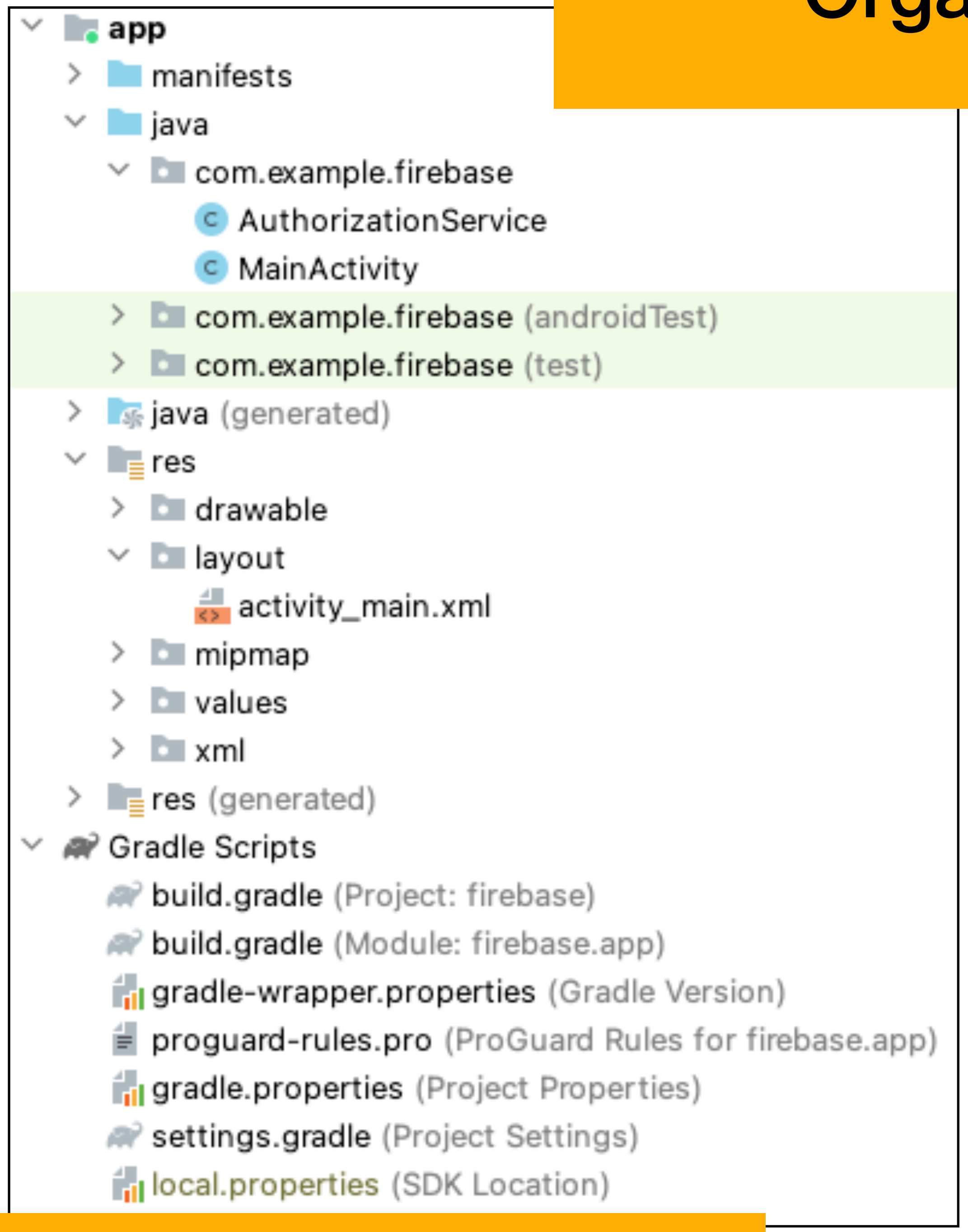
Data Layer

Figure 1. Diagram of a typical app architecture.



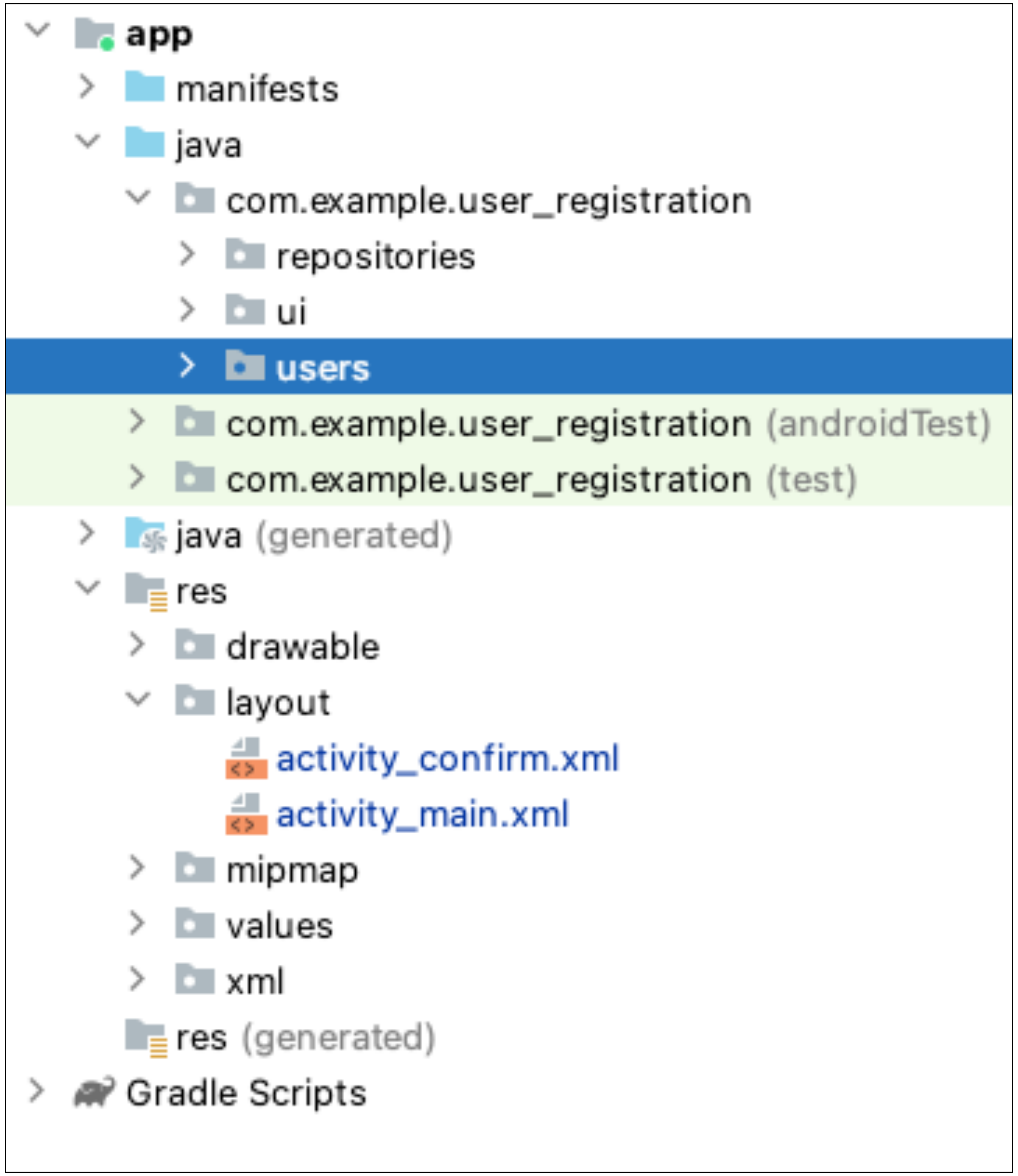
```
Android
app
  manifests
  java
    com.example.user_registration
      ConfirmActivity        UI layer
      MainActivity
      User                   Domain layer
      UsernameValidator
      UserRepository         Data layer
    com.example.user_registration (androidTest)
    com.example.user_registration (test)
  java (generated)
  res
    drawable
    layout
      activity_confirm.xml   UI layer
      activity_main.xml
    mipmap
    values
    xml
  res (generated)
Gradle Scripts
```

Organisation



```
app
  manifests
  java
    com.example.firebase
      AuthorizationService
      MainActivity
    com.example.firebase (androidTest)
    com.example.firebase (test)
  java (generated)
  res
    drawable
    layout
      activity_main.xml
    mipmap
    values
    xml
  res (generated)
Gradle Scripts
  build.gradle (Project: firebase)
  build.gradle (Module: firebase.app)
  gradle-wrapper.properties (Gradle Version)
  proguard-rules.pro (ProGuard Rules for firebase.app)
  gradle.properties (Project Properties)
  settings.gradle (Project Settings)
  local.properties (SDK Location)
```

Where do we put this file?



```
app
  manifests
  java
    com.example.user_registration
      repositories
      ui
      users
    com.example.user_registration (androidTest)
    com.example.user_registration (test)
  java (generated)
  res
    drawable
    layout
      activity_confirm.xml
      activity_main.xml
    mipmap
    values
    xml
  res (generated)
Gradle Scripts
```

Agenda

Application Architecture

- Software Architecture
- Layered Architecture
- Common architectural principles in layered architecture
  - Seperation of concerns
  - Unidirectional Data Flow
  - Single source of truth
  - Drive UI from data model
- ViewModel

Separation of concerns

Architectural Principles

It's a common mistake to write all your code in an Activity or a Fragment. These UI-based classes should only contain logic that handles UI and operating system interactions.

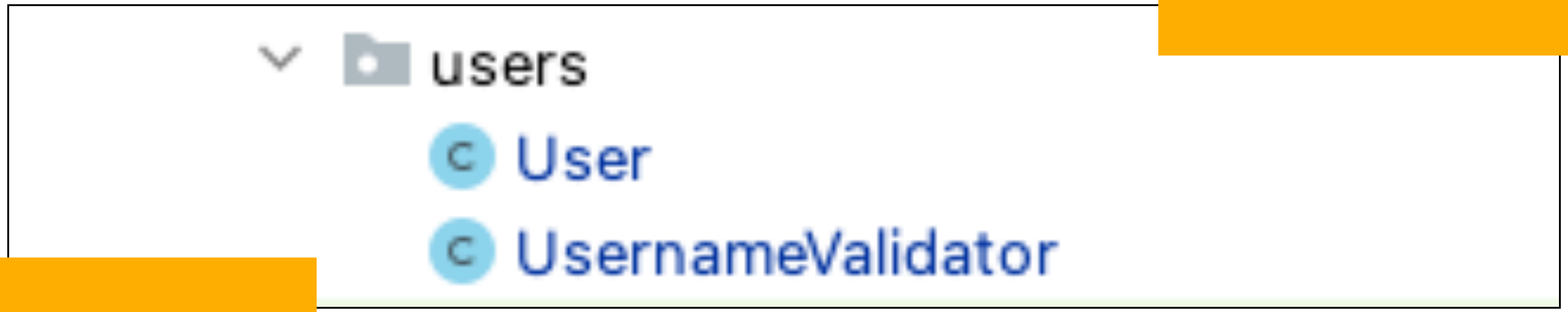
https://developer.android.com/topic/architecture

Separation of concerns

Architectural Principles

Keep code out of Activities (unless it is UI interactions)

Delegate to other classes



users

User

UsernameValidator

Separation of concerns + Reusability

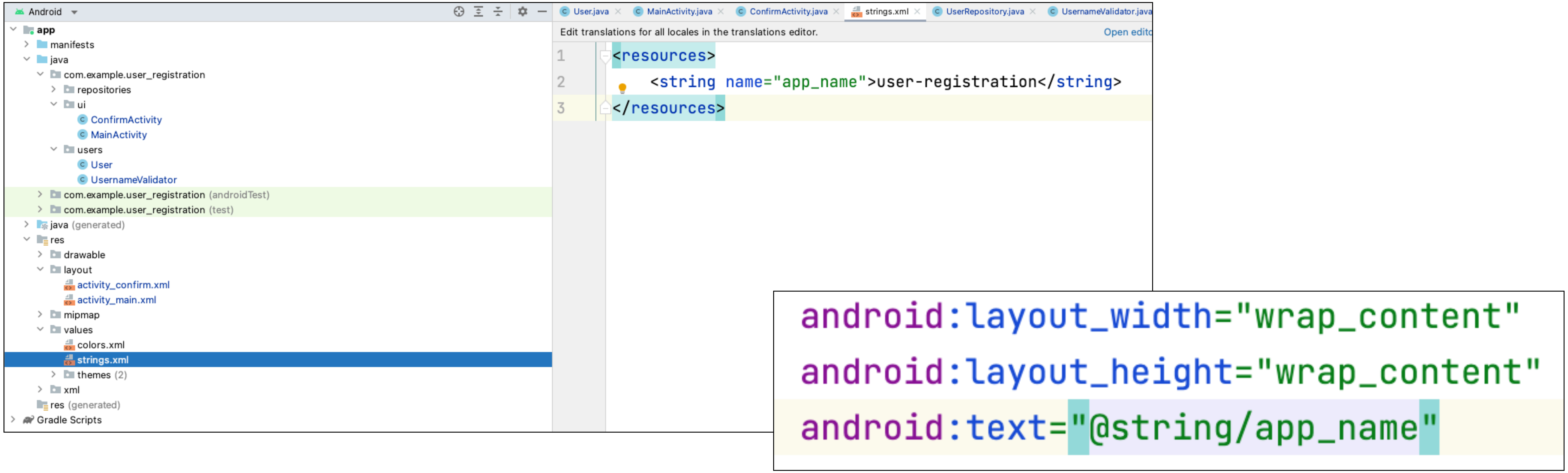
Single Source of Truth

Architectural Principles

The SSOT is the *owner* of that data, and only the SSOT can modify or mutate it. To achieve this, the SSOT exposes the data using an immutable type, and to modify the data, the SSOT exposes functions or receive events that other types can call.

Single Source of Truth

Architectural Principles



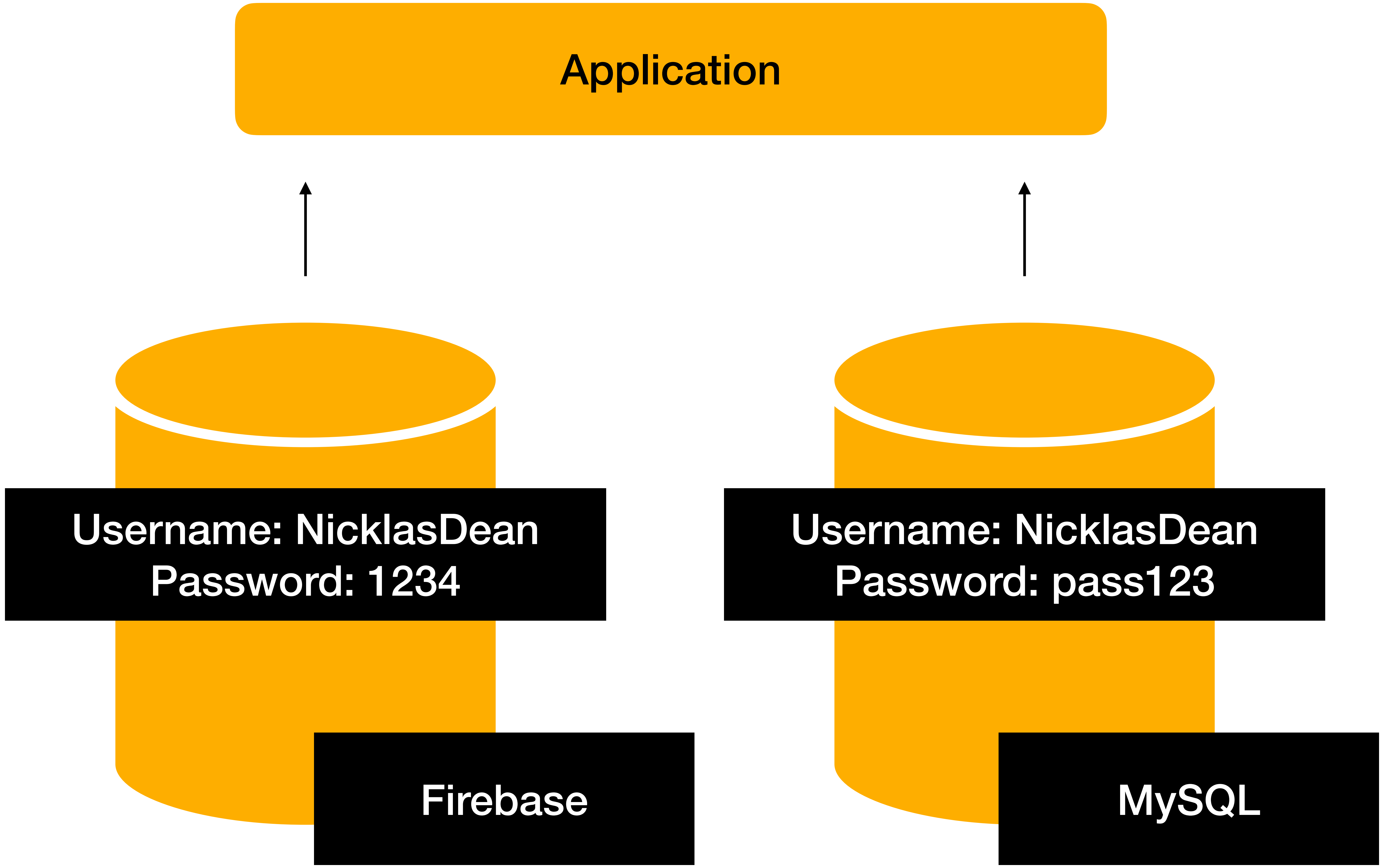
Edit translations for all locales in the translations editor.

```
<resources>
    <string name="app_name">user-registration</string>
</resources>
```

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/app_name"
```

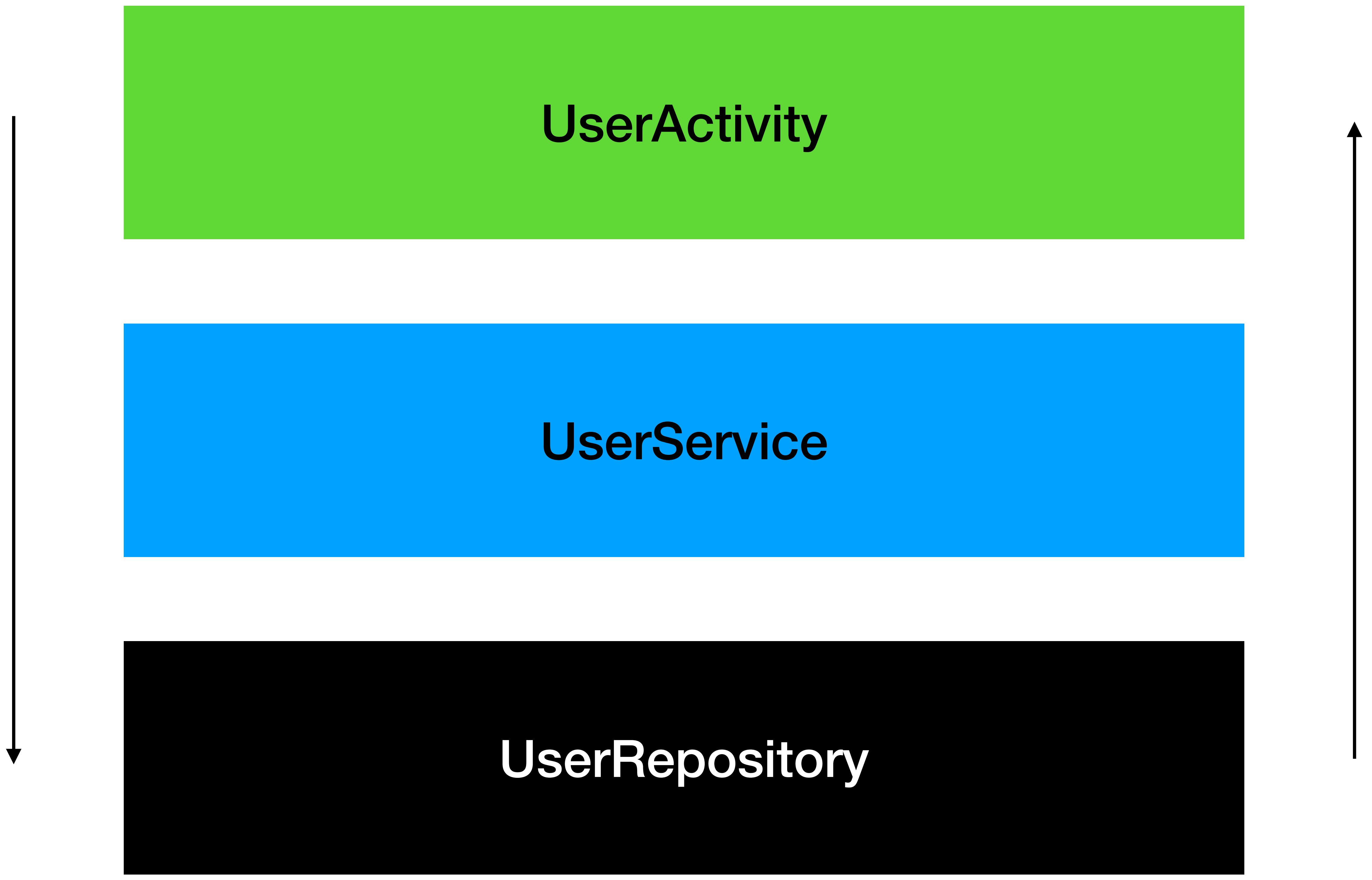
Single Source of Truth

Architectural Principles



Unidirectional data flow

Architectural Principles



User Request / Event

Response/data

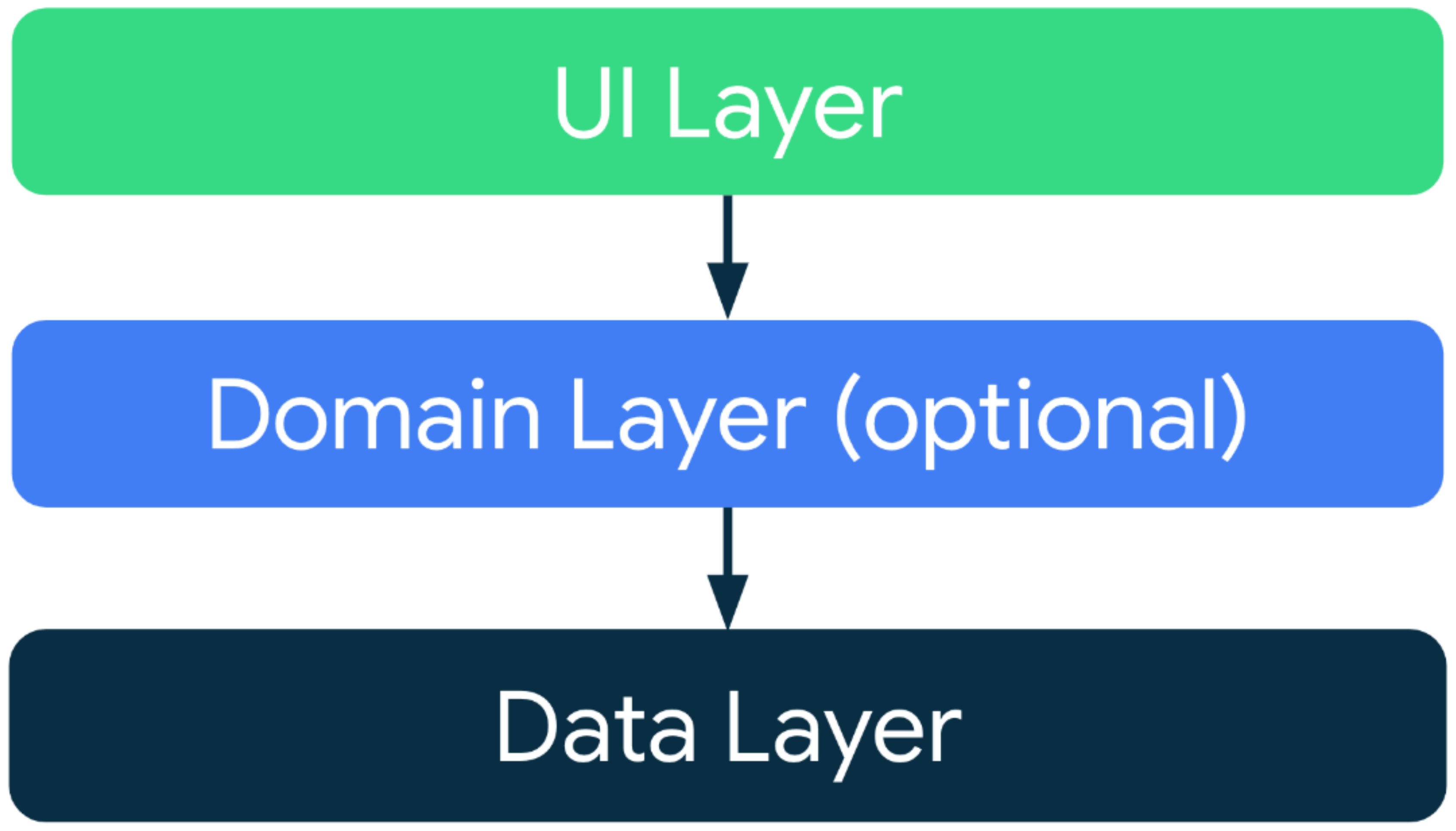
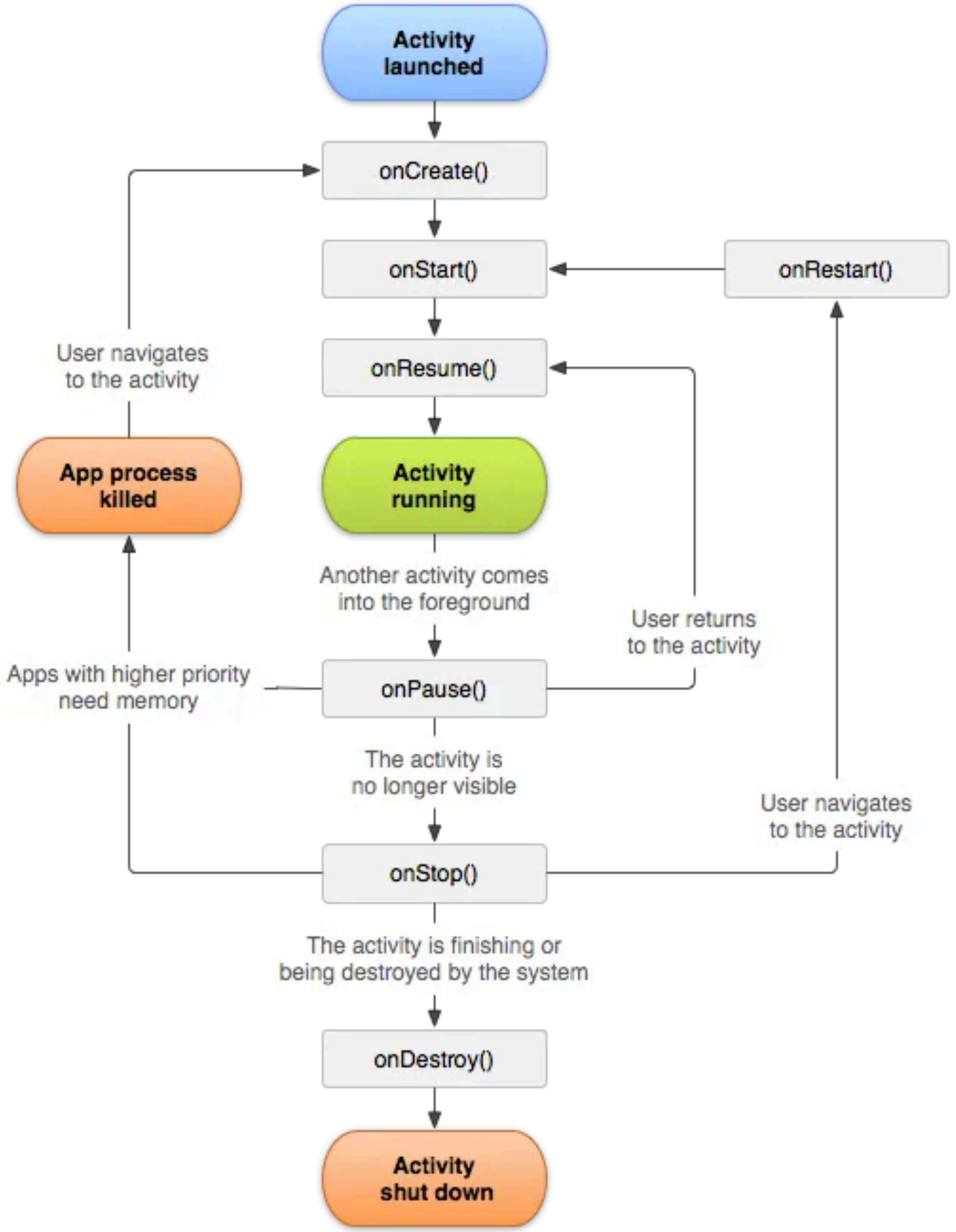
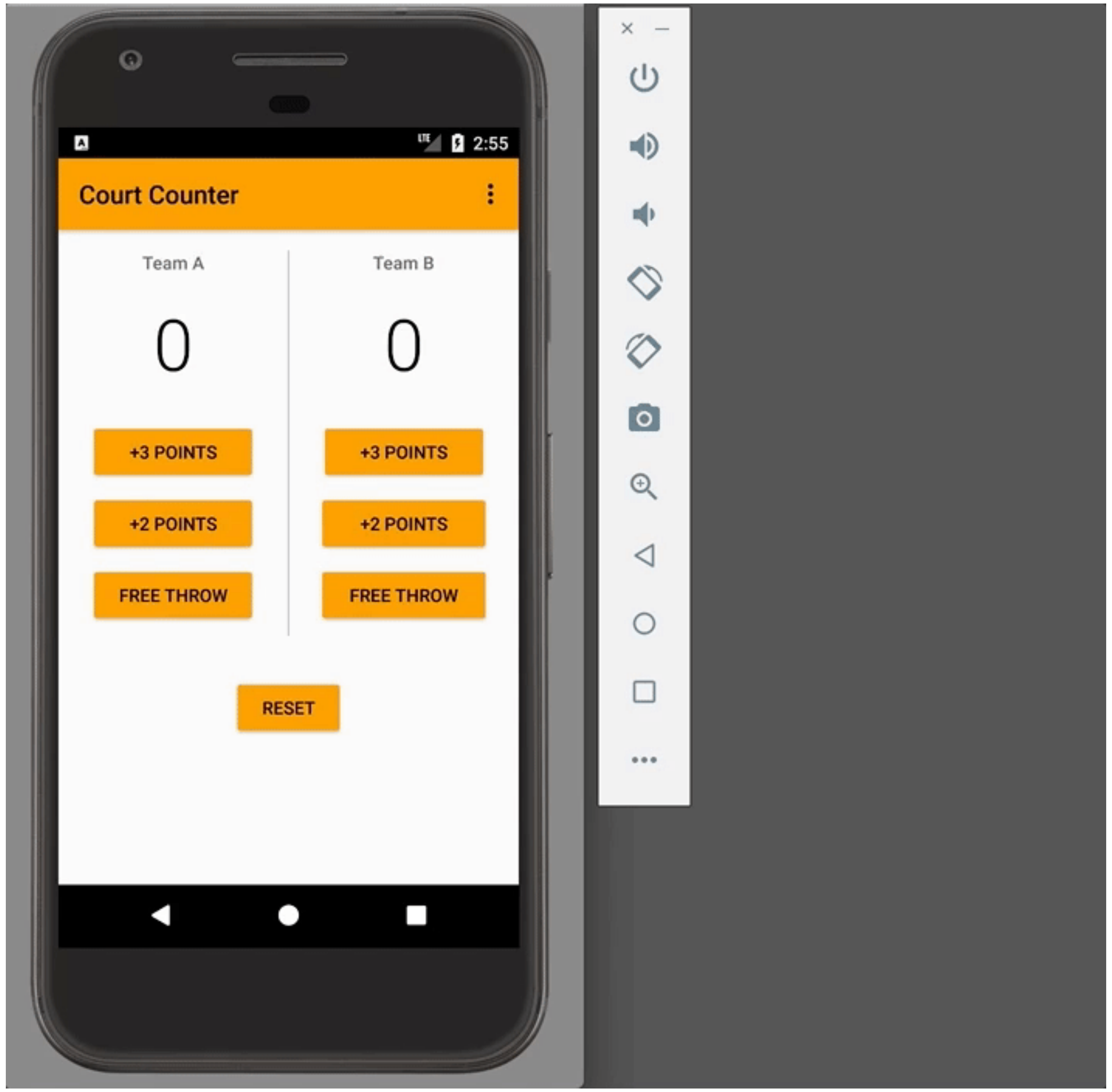


Figure 1. Diagram of a typical app architecture.

Drive UI from data model

Architectural Principles

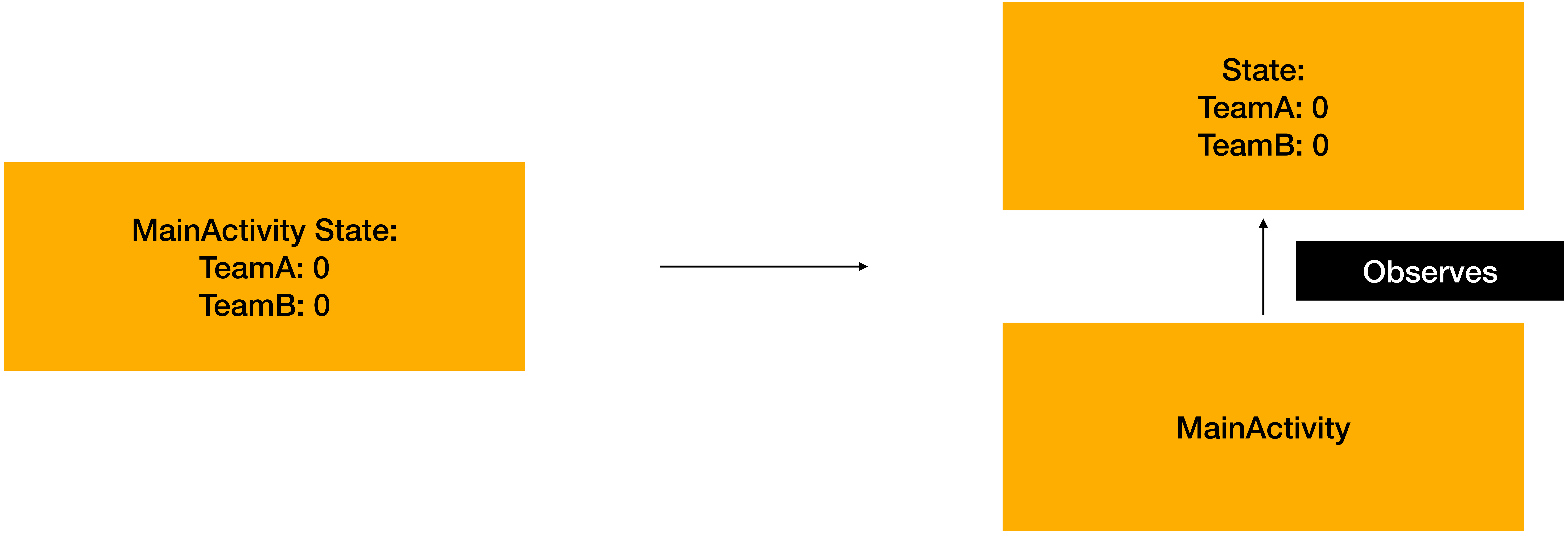

Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down



https://github.com/udacity/Court-Counter

https://medium.com/androiddevelopers/viewmodels-a-simple-example-ed5ac416317e

Drive UI from data model



MainActivity State:
TeamA: 0
TeamB: 0

State:
TeamA: 0
TeamB: 0

Observes

MainActivity

Drive UI from data model

Architectural Principles

Example: https://github.com/nicklasdean/court-counter/tree/main

Objective

Assess how well your application adheres to the introduced principles

What changes could better your application?