Android

Paolo Falcarin

Ca' Foscari University of Venice

Department of Enviromental Sciences, Informatics and Statistics

paolo.falcarin@unive.it

Software Security

4 April 2025

Outline

Ca' Foscari University of Venice

- Android OS, SDK, and Runtime
- Basics of Android programming
  - Basic App Components
    - Activities and Intents
  - Manifest and Resources
  - Permissions
  - Layouts and Responsive UI Design
  - Making REST requests

What is a Mobile Application?

Ca' Foscari University of Venice

- "A mobile app or mobile application is a computer program designed to run on a mobile device such as a phone/tablet or watch." - Wikipedia
- Isn't a mobile device just a smaller computer?
- Mobility imposes restrictions on program design:
  - Computational Power (small devices, limited power)
  - Battery (must last for as long as possible)
  - Input Methods (may be touch only)
  - Screen Size (watch, small screen)
  - Unreliable Network Connection (cannot assume stable connection)

Differences from “Regular” Apps

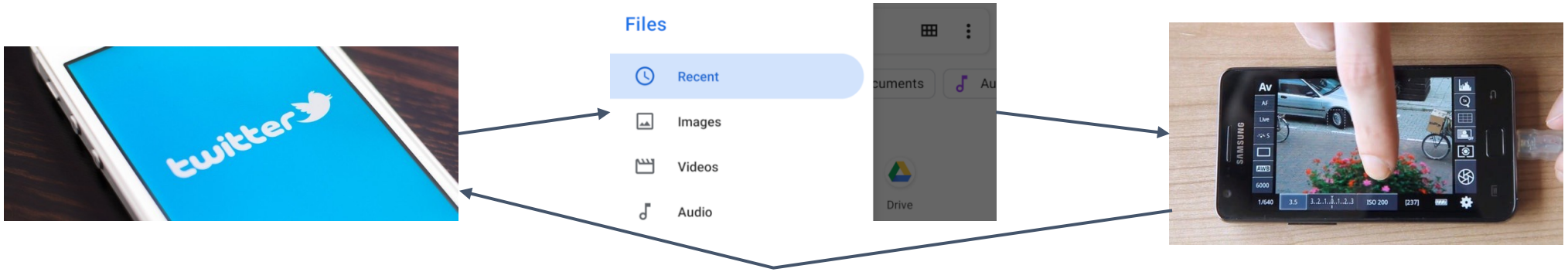
Ca’ Foscari University of Venice

- Not strictly defined, but…
- Mobile apps must adapt to multiple devices.
  - Different layouts for different screen sizes.
  - Can enable/disable features based on hardware.
  - Requires more thought about design and User Interface.

Differences from “Regular” Apps

Ca’ Foscari University of Venice

- Apps are designed to form an ecosystem.
  - Each app is a collection of components:
    - Activities provide a user interface.
    - The main activity starts when you tap the icon, but other apps can directly link to other “activities”.
    - Broadcast receivers and Services perform background tasks.
  - Apps often specialized for smaller tasks.
    - Purpose-built apps link to other apps for common features.



Android

Ca' Foscari University of Venice

- Most popular mobile operating system
  - 2.5 billion active devices
  - Phones, tablets, watches, car displays, TVs, IoT devices, speakers, home automation, ...
- Apps written in Android SDK.
  - Supports Java, Kotlin, C++.
- Code, data, resources compiled into APK.
  - Android Package
  - Compiled into device-specific code and run in Android OS (Linux-based)

Android Fundamentals

Ca' Foscari University of Venice

- Each app operates in a sandbox.
  - Each app is a "user".
  - Only that app can access its files.
  - Each app runs in its own virtual machine and process.
- Android implements "principle of least privilege".
  - Each app can access only components it requires.
  - Users must grant access to location, camera, bluetooth, files, etc.

Versioning

Ca' Foscari University of Venice

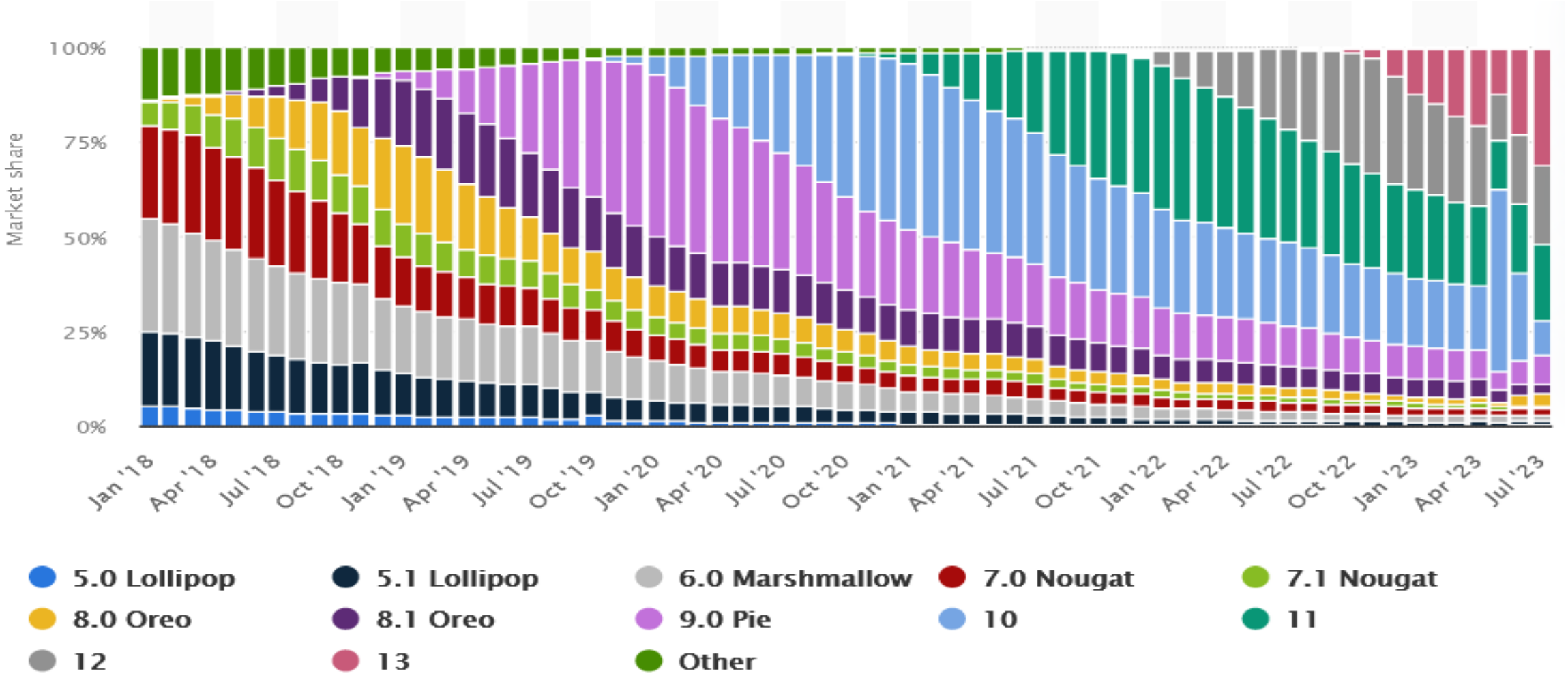
- Apps target a minimal "API level".
  - Defines available features.
- Apps must update compatibility or be disabled.

Android Jelly Bean	Jelly Bean	4.1 – 4.1.2	16	July 9, 2012		21.33.56 (September 2021)
		4.2 – 4.2.2	17	November 13, 2012		
		4.3 – 4.3.1	18	July 24, 2013		
Android KitKat	Key Lime Pie	4.4 – 4.4.4	19	October 31, 2013	October 2017	23.30.13 (August 2023)
		4.4W – 4.4W.2	20	June 25, 2014	?	
Android Lollipop	Lemon Meringue Pie	5.0 – 5.0.2	21	November 4, 2014[20]	November 2017	23.43.13 (November 2023)
		5.1 – 5.1.1	22	March 2, 2015[21]	March 2018	
Android Marshmallow	Macadamia Nut Cookie	6.0 – 6.0.1	23	October 2, 2015[22]	August 2018	
Android Nougat	New York Cheesecake	7.0	24	August 22, 2016	August 2019	
		7.1 – 7.1.2	25	October 4, 2016	October 2019	
Android Oreo	Oatmeal Cookie	8.0	26	August 21, 2017	January 2021	
		8.1	27	December 5, 2017	October 2021	
Android Pie	Pistachio Ice Cream[23]	9	28	August 6, 2018	January 2022	
Android 10	Quince Tart[24]	10	29	September 3, 2019	February 2023	
Android 11	Red Velvet Cake[24]	11	30	September 8, 2020	November 2023	
Android 12	Snow Cone	12	31	October 4, 2021		
Android 12L	Snow Cone v2	12.1[a]	32	March 7, 2022		
Android 13	Tiramisu	13	33	August 15, 2022		
Android 14	Upside Down Cake[27]	**14**	34	October 4, 2023		
Android 15	Vanilla Ice Cream[28]	15	TBA	Q3 2024	—	—

Legend: Old version | Older version, still maintained | **Latest version** | Latest preview version | Future release

Android versions share

Ca' Foscari University of Venice



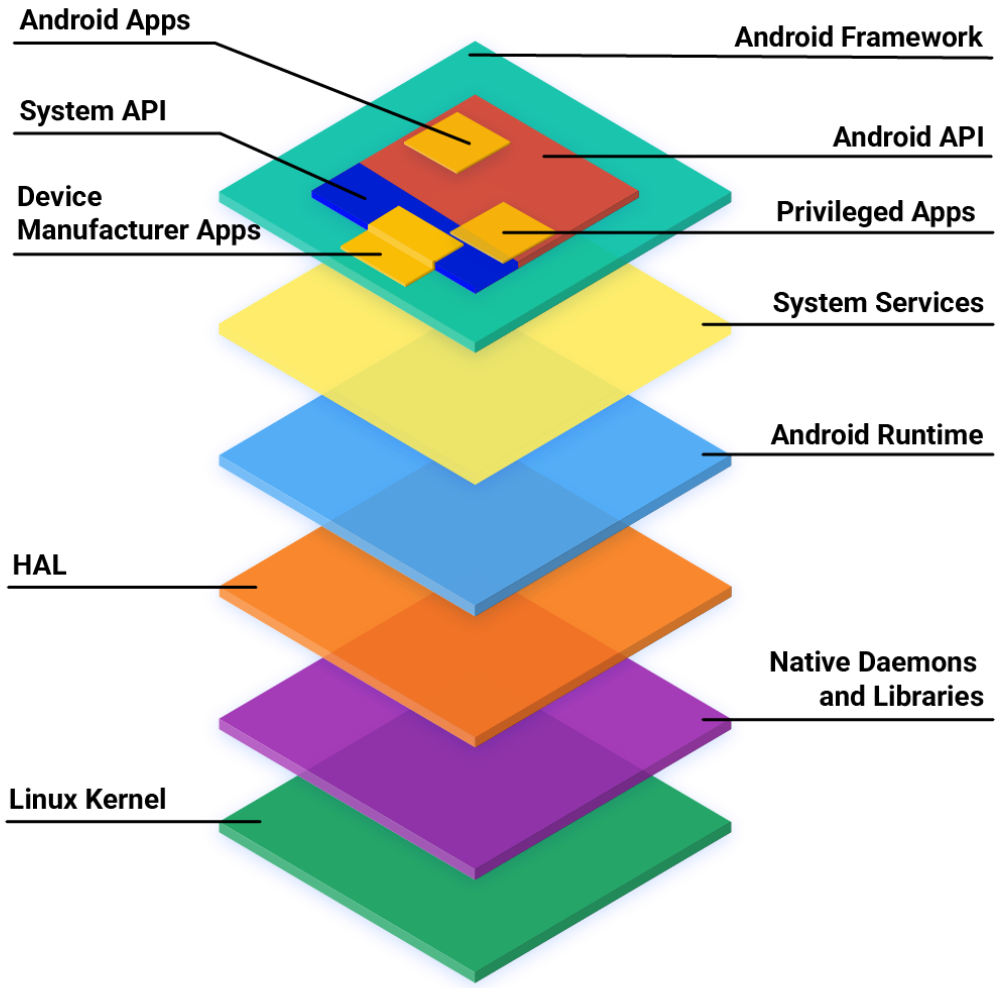
https://www.statista.com/statistics/921152/mobile-android-version-share-worldwide/

Android Fundamentals

Android Architecture

Ca' Foscari University of Venice

- Android is an open source software stack created for a wide array of devices with different form factors.
- A group of companies known as the Open Handset Alliance (OHA), led by Google, originated Android.
- Android Open Source Project (AOSP) maintains the Android Compatibility Program



Android Architecture - Apps

- *Android app*
  - Created solely using the Android API.
  - Google Play Store is widely used to find and download Android apps, though there are many other alternatives.
- *Privileged app*
  - created using a combination of the Android and system APIs.
  - These apps must be preinstalled as privileged apps on a device.
- *Device manufacturer app*
  - App created using a mix of Android API, system API, and direct access to the Android framework implementation.
  - Because a device manufacturer might directly access unstable APIs within the Android framework, these apps must be preinstalled on the device and can be updated only when the device's system software is updated.

Android Architecture – APIs and Services

- *Android API*
  - It is the publicly available API for third-party Android app developers.
- *Android framework*
  - A group of Java classes, interfaces, and other precompiled code upon which apps are built.
  - Parts of the framework are publicly accessible through the use of the Android API.
  - Other part of the framework are available only to OEMs through the use of the system APIs.
  - Android framework code runs inside an app's process.
- *System services*
  - are modular, focused components such as system_server, SurfaceFlinger, and MediaService.
  - Functionality exposed by Android framework API communicates with system services to access the underlying hardware.

Android Architecture – ART and HAL

- Android runtime (ART)
  - A Java runtime environment provided by AOSP.
  - ART performs the translation of the app's bytecode into processor-specific instructions that are executed by the device's runtime environment.
- Hardware abstraction layer (HAL)
  - It is an abstraction layer with a standard interface for hardware vendors to implement.
  - HALs allow Android to be agnostic about lower-level driver implementations
  - Using a HAL lets you implement functionality without affecting or modifying the higher level system.

Android Native and Kernel

- *Native daemons and libraries*
  - This layer include init, healthd, logd, and storaged.
  - These daemons interact directly with the kernel or other interfaces and do not depend on a userspace-based HAL implementation.
  - Native libraries in this layer include libc, liblog, libutils, libbinder, and libselinux. These Native libraries interact directly with the kernel or other interfaces and don't depend on a userspace-based HAL implementation.
- *Kernel*
  - It is the central part of any operating system and talks to the underlying hardware on a device.
  - Where possible, the AOSP kernel is split into hardware-agnostic modules and vendor-specific modules.

Android RunTime (ART)

- Android runtime (ART) is the managed runtime used by applications and some system services on Android.
- ART and its predecessor Dalvik were originally created specifically for the Android project.
  - ART as the runtime executes the Dalvik Executable format and Dex bytecode specification.
- ART and Dalvik are compatible runtimes running Dex bytecode, so apps developed for Dalvik should work when running with ART.
- However, some techniques that work on Dalvik do not work on ART.

Ahead-of-time (AOT) compilation

Ca' Foscari University of Venice

- ART introduces ahead-of-time (AOT) compilation, which can improve app performance.
- ART also has tighter install-time verification than Dalvik.
- At install time, ART compiles apps using the on-device dex2oat tool.
- This utility accepts DEX files as input and generates a compiled app executable for the target device.
- The utility should be able to compile all valid DEX files without difficulty

Garbage Collection

Ca' Foscari University of Venice

- Garbage collection (GC) is very resource intensive, which can impair an app's performance, resulting in choppy display, poor UI responsiveness, and other problems.

ART improves garbage collection in several ways:

- Mostly concurrent design with a single GC pause
- Concurrent copying to reduce background memory usage and fragmentation
- The length of the GC pause is independent of the heap size
- Collector with lower total GC time
  - for the special case of cleaning up recently-allocated, short-lived objects

Debugging Features

Ca' Foscari University of Venice

- ART supports a number of new debugging options, particularly in monitor- and garbage collection-related functionality.
- For example, you can:
  - See what locks are held in stack traces, then jump to the thread that holds a lock.
  - Ask how many live instances there are of a given class, ask to see the instances, and see what references are keeping an object live.
  - Filter events (like breakpoint) for a specific instance.
  - See the value returned by a method when it exits (using "method-exit" events).
  - Set field watchpoint to suspend the execution of a program when a specific field is accessed and/or modified.

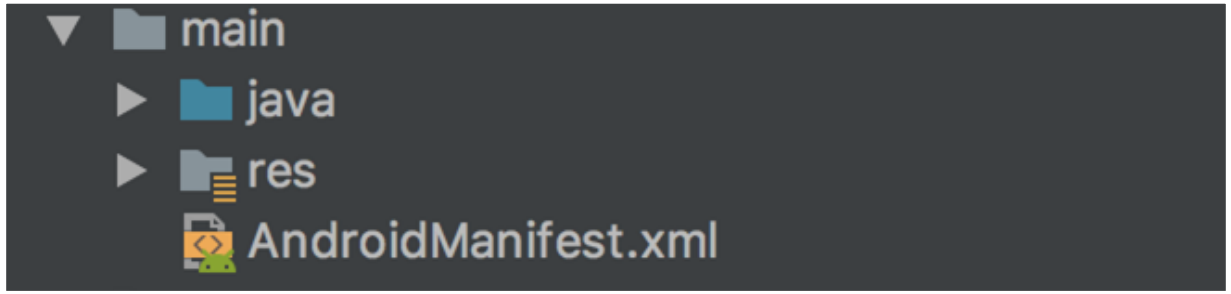
Android NDK

- Lets Android application developers build performance-critical portions of apps in Native code
- The NDK provides:
  - Tools and build files used to generate native code libraries from C, C++
  - A way to embed the corresponding native libraries into app package
  - A set of native system headers and libraries
  - Documentation, samples, and tutorials
  - Provides stable headers for libc (the C library), libm (the Math library), the JNI interface, and other libraries

Android Apps - Structure

Ca' Foscari University of Venice

- Android Code
  - All the logic
- Resources
- Manifest



App Components

Ca' Foscari University of Venice

- Core building blocks of an app.
- Entry points for an app.
  - Activities are "screens" with a UI.
  - Services run a task in the background.
  - Broadcast receivers deliver events to apps outside of regular user flow.
  - Content providers manage a pool of information.

Activities

- A single screen with a user interface
  - Most apps have multiple independent activities
    - E-mail: Show messages, compose, read.
- If allowed, other apps can start any activity
  - Camera app opens "Compose" activity to share photo
- Activities control and link processes
  - Ensure the current process is not killed
  - Link to calling activities and maintain their process
  - Model state in case process is killed
  - Model flow between apps

Services

Ca' Foscari University of Venice

- Entry point for running a background process
  - Playing music, sending files
- Does not provide a direct UI
- Can be started by an activity
  - Can maintain a notification to allow user interaction
  - Services without notifications can be killed if resources are needed by OS
- Bound services offer an API to the calling app
  - Maintained as long as needed, then killed

Broadcast Receivers

- Allows OS to deliver events when the app is not running.
  - Listen to system-wide broadcast announcements.
- Respond to events like a photo being taken.
- Often notify users that an event has occurred.
- Often minimal, used as a gateway to launch activities or services.

Content Providers

- Manages a shared set of app data.
- Other, allowed, apps can query or modify the data.
  - Android has a Content Provider for contact data.
- An entry point into an app for publishing data items.
  - Identified by a URI.
  - Owning app wakes up when a URI is accessed.
  - URIs provide a secure way to pass content.
    - Content is locked and accessed through temporary permission.

Intents

Ca' Foscari University of Venice

- Asynchronous messages that bind components at runtime.
  - Messengers that request actions from other components.
  - Start an activity, start a service, deliver a broadcast.
  - Can convey a result back to the caller.
- Explicit intents activate a specific component.
- Implicit intents activate types of components.

Explicit Intents

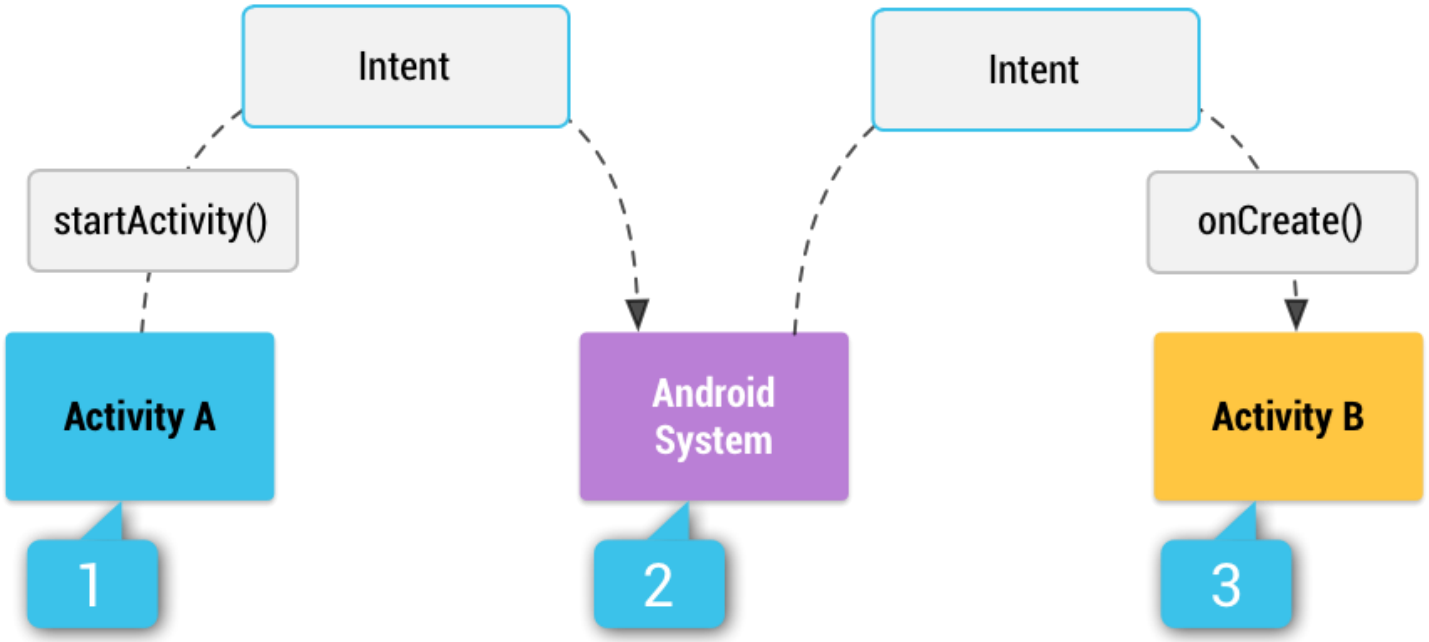
- Name a specific app (by package or component).
- Often used by one component to start another within the same app.

```
// Executed in an Activity, so 'this' is the Context
// The fileUrl is a string URL, such as "http://www.example.com/image.png"
Intent downloadIntent = new Intent(this, DownloadService.class);
downloadIntent.setData(Uri.parse(fileUrl));
startService(downloadIntent);
```

Implicit Intents

Ca' Foscari University of Venice

- Describe a type of action you want to perform.
- Allow the system to find components that can perform that action (selected by the user).
  - Done using IntentFilters



Implicit Intents

Ca' Foscari University of Venice

```
// Create the text message with a string
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// Verify that the intent will resolve to an activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(sendIntent);
}
```

This is a SEND action

It sends a text message

Android will search for all Activities that can handle a SEND action on plain text.

The Manifest

Ca' Foscari University of Venice

- Each android app needs an AndroidManifest.xml file
- Essential information
  - App name
  - Components
  - SDK version
  - Permissions
  - …

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="myapp.wm1819.grischaliebel.de.myapplication">

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="My Application"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        android:usesCleartextTraffic="true"
        tools:ignore="GoogleAppIndexingWarning">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".DisplayMessageActivity"></activity>
    </application>

</manifest>
```

Declaring Components

Ca' Foscari University of Venice

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
        </activity>
        ...
    </application>
</manifest>
```

Resource for an icon.

Class name of Activity

Declares an Activity

Label used to identify Activity

Declaring Component Capabilities

Ca' Foscari University of Venice

```
<manifest ... >
    ...
    <application ... >
        <activity android:name="com.example.project.ComposeEmailActivity">
            <intent-filter>
                <action android:name="android.intent.action.SEND" />
                <data android:type="*/*" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Intent Filters declare how app responds to Intents

This Activity is registered as an option for ACTION_SEND intents.

Declaring App Requirements

Ca' Foscari University of Venice

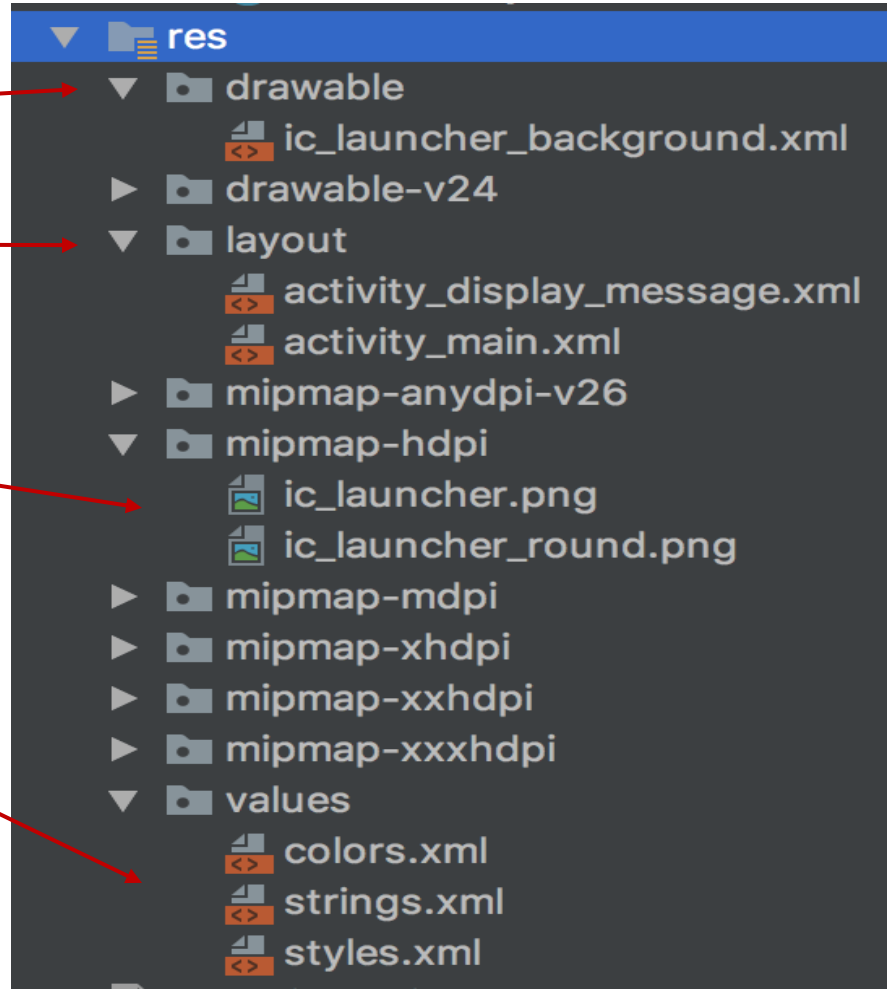
- App requires at least Android 2.1 and a camera.
- Setting "required" attribute to false indicates that the app uses the camera, but can function without.

```
<manifest ... >
    <uses-feature android:name="android.hardware.camera.any"
                  android:required="true" />
    <uses-sdk android:minSdkVersion="7" android:targetSdkVersion="19" />
    ...
</manifest>
```

App Resources

Ca' Foscari University of Venice

- Bitmaps/Pictures
- UI definitions in XML
- Launcher icons
- Text strings (incl. translations)



App Resources

- Each resource is assigned a unique ID.
  - Used to reference the resource from code or layout XML.
  - File *res/drawable/logo.png* -> resource ID named *R.drawable.logo*
- Can provide alternate resources for configurations.
  - UI strings can be used to swap one language for another
  - Qualifier appended to directory (*res/values-fr/*)
  - Many default qualifiers supported for different screen sizes, device types, orientations (layout vs portrait).
  - Allows automated responsiveness.

Permissions

- By default, apps are not allowed to use hardware, access data, access network.
- Permissions must be explicitly asked for in manifest

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.SEND_SMS"/>
```

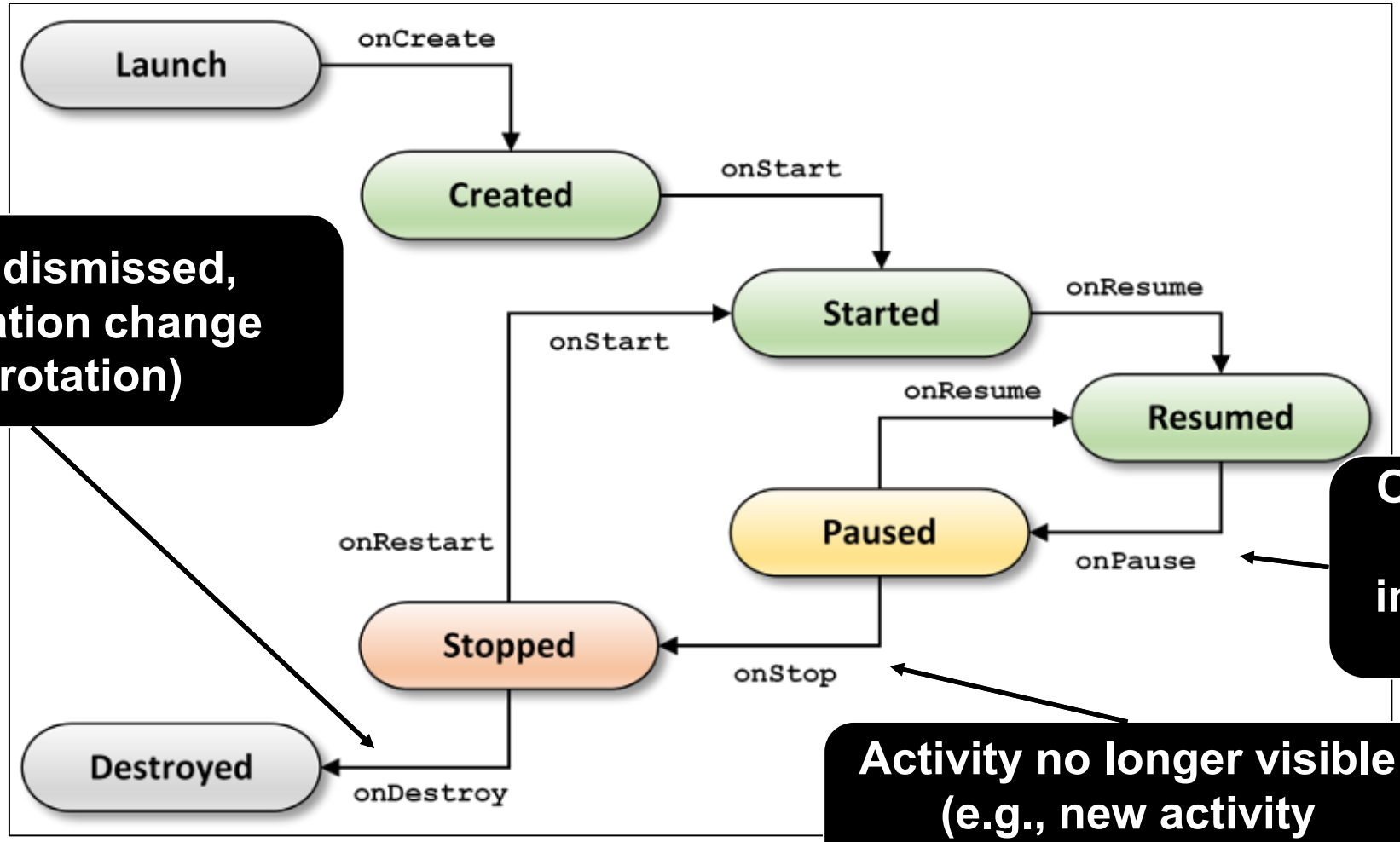
- Users must grant permission for dangerous requests (like the ones above)
  - Formerly, user had to agree to all requests to install app (< API 6.0 (23)).
  - Now, permissions granted individually.

The Concept of Activities

- Rather than interacting with apps as atomic units, Activities interact directly with Activities.
  - Any Activity can serve as an entry point to app interaction
- An Activity has a UI, and is usually a single screen.
  - An app may have a Settings Activity, a Select Photo Activity, ...
- One Activity is the "main activity".
  - Launches when you click the icon.
- Activities must have minimal dependencies on other Activities.

Activity Lifecycle

Ca' Foscari University of Venice

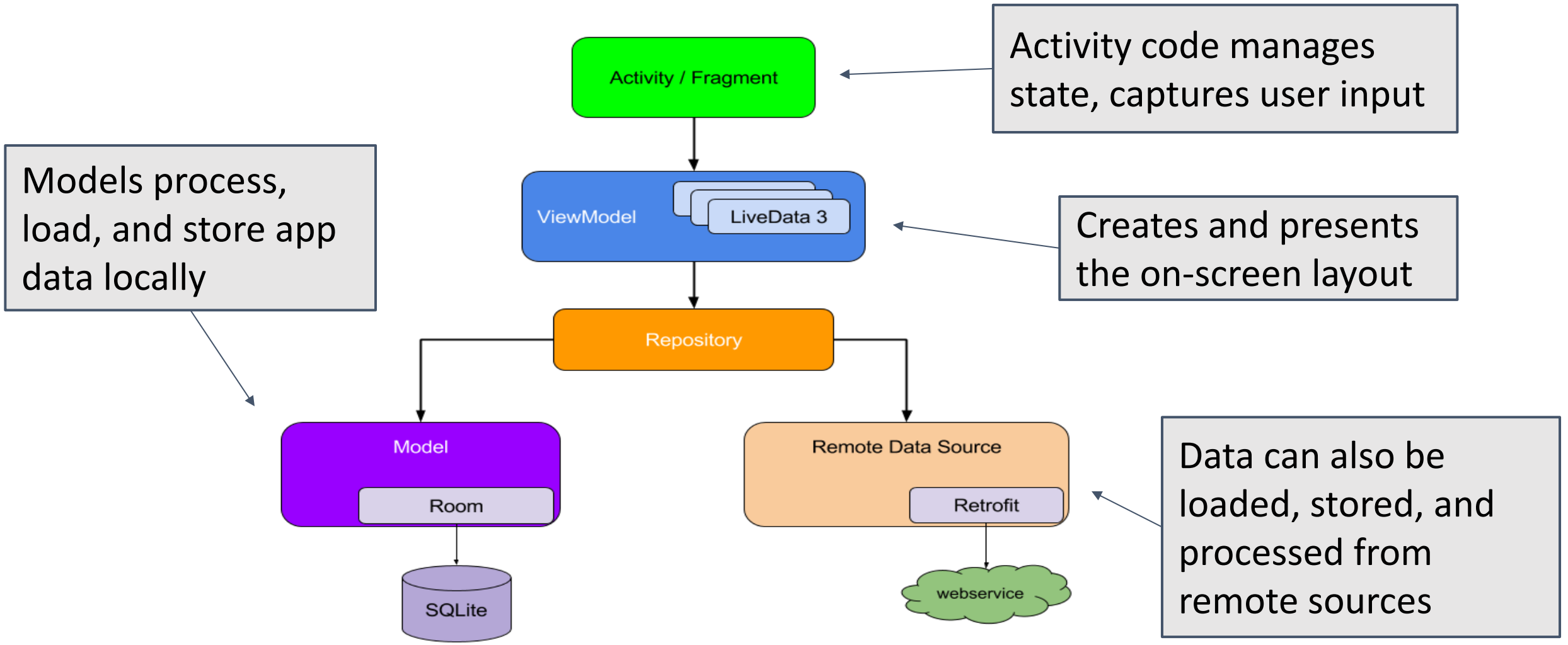


Activity dismissed, configuration change (e.g., rotation)

Other Activity in foreground, incoming phone call, ...

Activity no longer visible (e.g., new activity started)

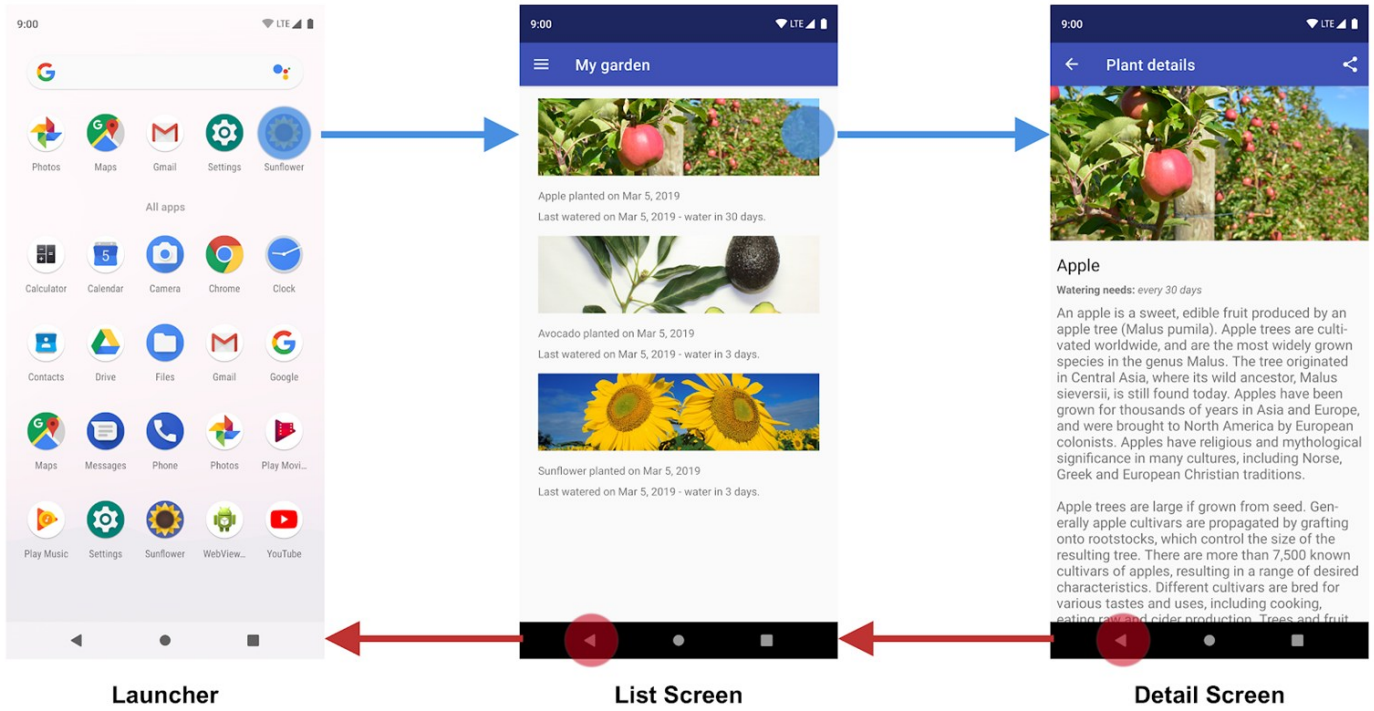
App Architecture

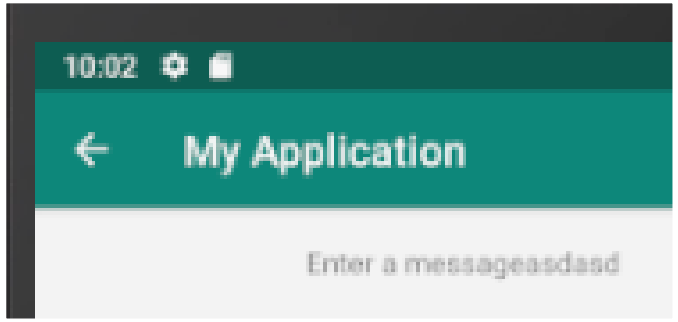


Navigation

Ca' Foscari University of Venice

- All apps have a fixed start location.
- A stack of Activities is maintained.
- When you press back, you pop from the stack.
- Also provide "up navigation".
  - (exit a chain to a set location)
  - Define a parent activity in the manifest.





Activity Best Practices

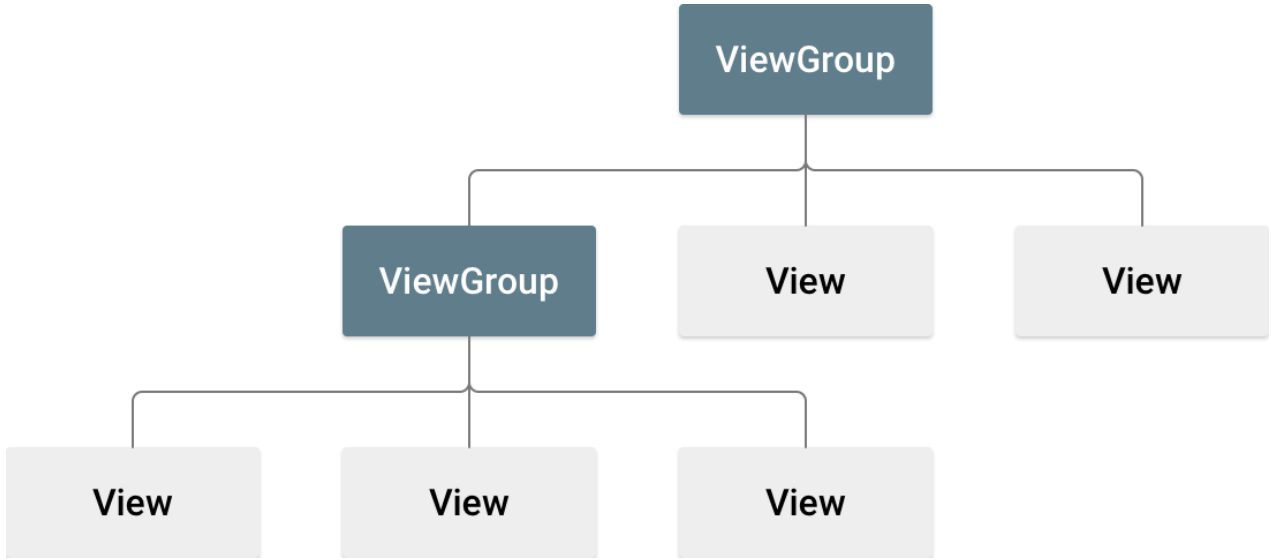
Ca' Foscari University of Venice

- Activities should coordinate with Data Models to retrieve a minimal amount of relevant data.
- Create independent, well-defined code modules.
- Make each module testable in isolation.
- Do not write code if an existing Activity does it.
- Use Models to persist fresh, relevant data.
- Assign one data source as the source of truth.

UI Design - Views and Layouts

Ca' Foscari University of Venice

- A layout (ViewGroup) defines the structure of the UI.
  - Containers that group one or more widgets (View).
    - A button, a text box.
- Many pre-defined types of layouts (LinearLayout, Constraint Layout).
- UI elements can be declared in XML or in code.



UI Design - XML

Ca' Foscari University of Venice

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
</LinearLayout>
```

Layout has two widgets, which have no constraints on each other's size or location

Text box containing a set string

Button with a set string

UI Design - Attributes

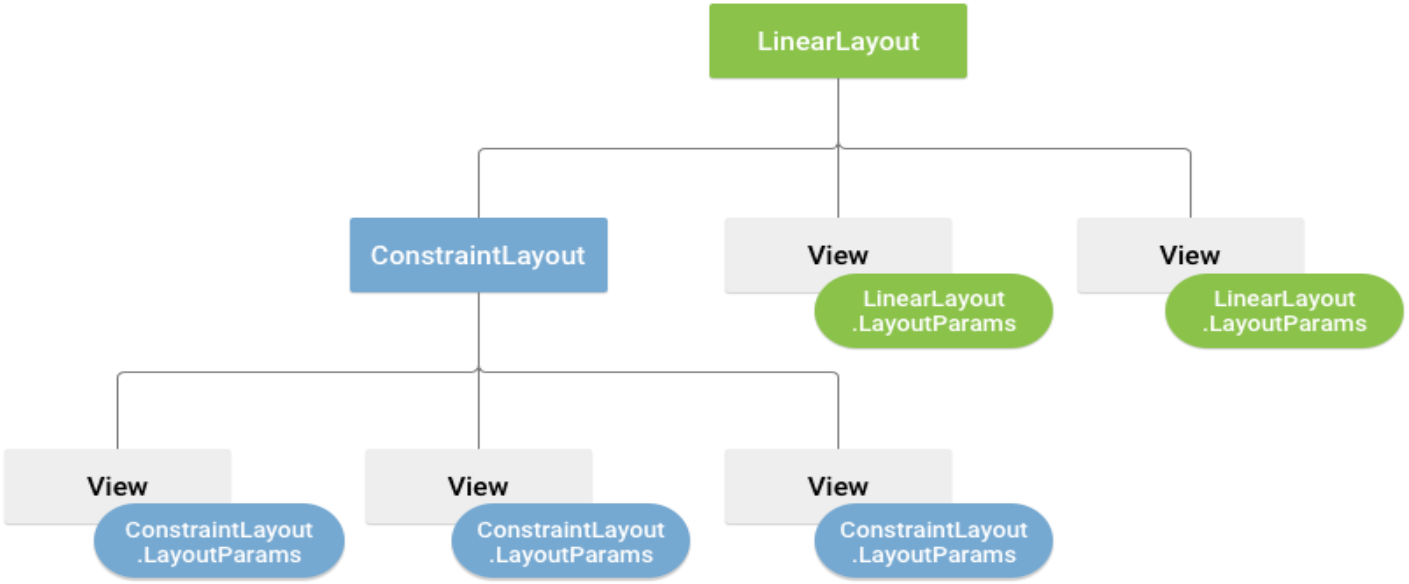
Ca' Foscari University of Venice

- All View objects have a unique identifier.
  - Integer assigned at compile time, mapped to a user-specified variable: android:id="@+id/my_button"

```
<Button android:id="@+id/my_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/my_button_text"/>
```

```
Button myButton = (Button) findViewById(R.id.my_button);
```

UI Design - Attributes



- LayoutParameters
  - Inherited by Views contained in that layout.
  - Define how Views appear.
- All ViewGroups have a width and height.
  - Each view must define width and height relative to this.
  - wrap_content sizes view to its content.
  - match_parent makes view as big as its parent ViewGroup allows.
  - Specify in dp (density-independent pixel units)

UI Design - Layouts

Ca' Foscari University of Venice

- Views are rectangles with left and top coordinates.
  - Can get location with getLeft() and getTop()
  - Defined relative to the parent.
- Size is defined in width and height.
  - Measured width/height are how big the view wants to be.
  - Drawing width/height are the actual size of the view on screen, after layout constraints.
  - These can differ.

Android resources

Ca' Foscari University of Venice

- Textbook Professional Android , 4th edition, Wiley
  - Download Code Examples from here for the teaching projects
  - https://www.wiley.com/en-it/Professional+Android%2C+4th+Edition-p-9781118949535
- Online official Android Developer Guide
  - https://developer.android.com/guide
- Download Android Studio
  - https://developer.android.com/studio
- Build your first app!
  - https://developer.android.com/codelabs/build-your-first-android-app