1. Lesson 27/02/23

Introduction to the Course topics

Slides 00

Goal of the Course:
The goal of Mobile Development course is to develop mobile applications for Android using Java or Kotlin.

Why sudy Mobile Development?
In order to have an idea on how to develop an Android App for a mobile phone since the use of them is widespread and the number of them in the world is increasing really fast.

Which Mobile platform?

- **12 - 15 years ago:** several competitors were on the market, the most important were Symbian and BlackBerry.
- **8 - 10 years ago:** several competitors were on the market, the most important were Symbian and Blackberry.

  Symbian: was used by many major mobile phone brands like Samsung, Motorola, Sony Ericcson and Nokia above all.

  Nokia: was a pioneer that established the smartphone industry, it was the most popular smartphone OS on a worldwide average in 2010.

  BlackBerry OS: was a proprietary mobile operationg system developed by the BlackBerry company for its line of smartphones.

- **More recently (last 5 years):** Android - iOS - Windows
  From 2014 the 90%+ of the market is Android + iOS the other platforms are unused.

- **Today:** Google Android - iOS jointly possess more than 98% of the global market share.

Mobile vs Desktop

- **Android:** is the most used in China, India, Africa, South America, Asia and several EU countries.
- **Windows:** is the most used in the Western world and America.

In general Android is the most used OS in thw world.

Why focus on Android?

We focus on Android for “Logistic reasons” in fact Android provides:

- **Free IDE** for the development (Android Studio)
- Possibility to **develop on several Operatin Systems** (Windows, macOS, Linux)
- **Free emulator** for executing the app
- Test of the app on your real Android phone

Link to the Google courses on Java and Kotlin for Android App development
https://developer.android.com/courses

Introduction to Android

Slide 01

Android Ecosystem

What is Android?

Android is a Mobile operating system (OS) based on a modified version of the Linux kernel and other oper source software.

- It is not Linux
- No native windowing system
- Does not include the full set of Linux utilities
- Dowsn’t include a graphical X server or the standard GNU libreries and so it can’t simply run Linux applications
- Only applications written specifically for Android can run
- Can’t run Android Apps on typical Linux distributions

Android provides a User Interface for touch screens

Android is used on the majority of all smartphones

Android also powers devices such as smartwatches, TVs and cars

Android is highly customizable for devices and is open source.

User Interaction

You can interact with Android devices with **touch gestures**, the most common are**:**

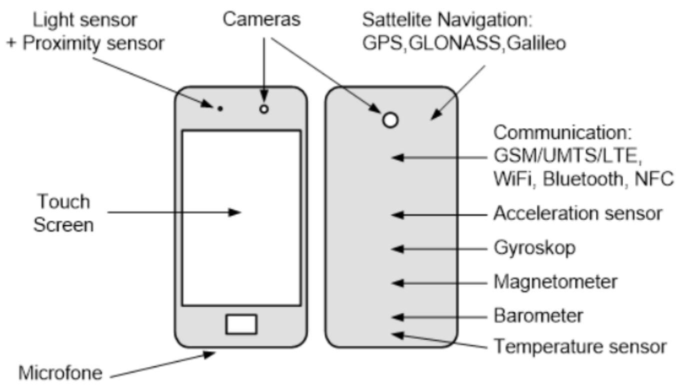
- Swiping
- Tapping
- Pinching

You can also interact using a **Virtual Keyboard,** for characters, numbers and emoji.
Depending on the type of input that you need you will have a different type of keyboard. If you need to insert a word you will use the Default Alphabet keyboard, if you need to insert a symbol or a number you will use the Default Numeric keyboard and if you need to insert a numeric sequence you will use the Keypad.

The user can interact also through the **Support for Bluetooth** or through **USB controllers** and **peripherals**.

Android and sensors

The sensors inside the devices can discover user actions and respond to them.



- **Accelerometer / Gyroscope:** through the gyroscope sensor of the device the content rotate as needed, switching from **portrait view** to **landscape view** and viceversa.
  Almost all the applications could be used both in portait mode and in landscape mode but some apps are suited only for one of the two.
- **GPS (GLONASS, Galileo):** used to adjust position on map by using three satellites (triangulation).

Android home screen

The home screen of an Android device have:

- Launcher icons for apps

- Self-updating widgets for live content
- Can be composed of multiple pages
- Folders to organize apps

Android Software Developer Kit (SDK)

The Android SDK (software development kit) is a set of development tools used to develop applications for the Android platform.
The Android SDK includes the following:

- Development tools (debugger, monitors, editors)
- Libraries
- Virtual devices (emulators)
- Documentation (developers.android.com)

Android Studio

It is the official Android IDE that permitts to develop, run, debug, test and package apps. It provides a project view and a visual layout editor. Android Studio also provides a virtual device where you can test virtually your app.

App Development

What is an Android app?

- One or more interactive screen
- Written using Java or Kotlin (logic) and XML (UI)
- Uses the Android Software Development Kit (SDK)
- Uses Android libraries and Android Application Framework
- Executed by Android Runtime Virtual Machine (ART)

Challenges of Android development

1. **Design UI elements that work on all devices.**
   Potentially the developed app will be used on devices with multiple screen sizes and resolutions so it must adapt to them.

Screen Property - Density:

If you use a fixed measures in you app (f.i. pixel) you can obtain a good result on a certain device but when the density of the screen changes the result too. So it’s possible that a good result on a certain device is a bad result in another one.
This depends on the density of pixel in the screen.

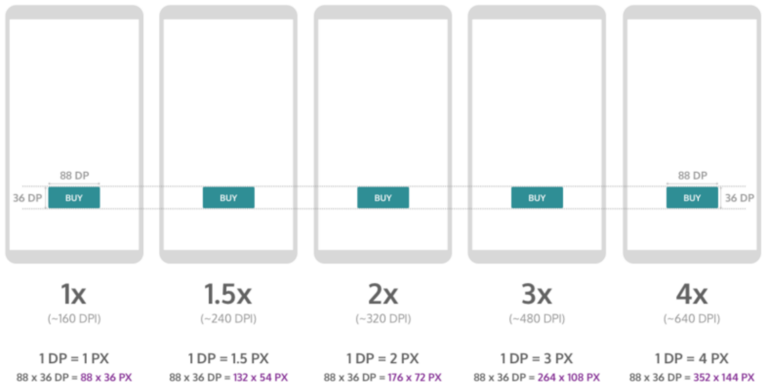
Screen Property - Size:

Different devices could have a different size of the screen or the same size of the screen with different number of pixels.

To preserve the visible size of your UI on screens with different densities you must design the app UI using density-independent pixels (dp) as unit of measurement.

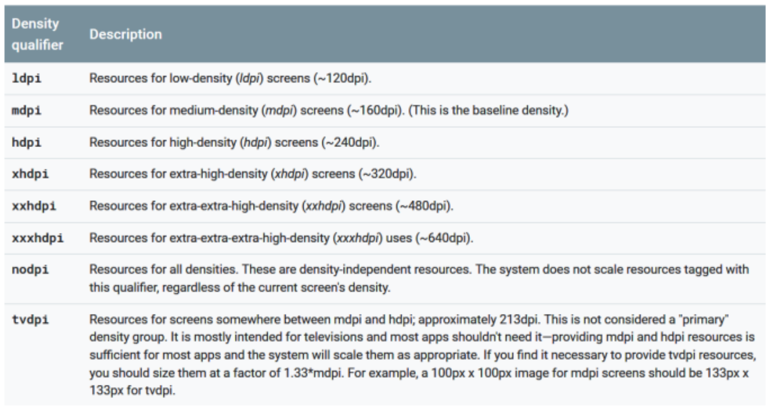
dp: is a virtual pixel unit that’s roughly equal to one pixel on a medium-density screen.

Android translates this value to the appropriate number of real pixels for each other density.



Screen Property - Qualifiers:

Qualifiers for different pixel densities that can be used for example to upload image asset on the project.



Density qualifier	Description
`ldpi`	Resources for low-density (*ldpi*) screens (~120dpi).
`mdpi`	Resources for medium-density (*mdpi*) screens (~160dpi). (This is the baseline density.)
`hdpi`	Resources for high-density (*hdpi*) screens (~240dpi).
`xhdpi`	Resources for extra-high-density (*xhdpi*) screens (~320dpi).
`xxhdpi`	Resources for extra-extra-high-density (*xxhdpi*) screens (~480dpi).
`xxxhdpi`	Resources for extra-extra-extra-high-density (*xxxhdpi*) uses (~640dpi).
`nodpi`	Resources for all densities. These are density-independent resources. The system does not scale resources tagged with this qualifier, regardless of the current screen's density.
`tvdpi`	Resources for screens somewhere between mdpi and hdpi; approximately 213dpi. This is not considered a "primary" density group. It is mostly intended for televisions and most apps shouldn't need it—providing mdpi and hdpi resources is sufficient for most apps and the system will scale them as appropriate. If you find it necessary to provide tvdpi resources, you should size them at a factor of 1.33*mdpi. For example, a 100px x 100px image for mdpi screens should be 133px x 133px for tvdpi.

2. **Performance: make your apps responsive and smooth.**
    - How fast the app runs
    - How easily the app connects to the network
    - Ho well the app manages battery and memory usage

    Performances are affected by some factors:
    - Battery level (low battery —> low performance)
    - Multimedia content (HD content reduces performances)
    - Internet access (concurrent streaming reduce networl performances)

3. **Security: keep source code and user data safe.**
    Take precautions to make the code, and the user' data as secure as possible:
    - Detect and remove unused classes, fields, methods, and attributes
    - Remove unused resources from the packaged app
    - Obfuscate the code: shortens the name of classes and members
    - Optimize the code
    - Protect critical user information such as logins and passwords
    - Secure your communication channel
    - Secure data at rest on the device

4. **Compatibility: run well on older platform versions.**
    - NOT focus only on the most recent Android version
    - NOT all users may have upgraded (or may be able to upgrade) their devices

Android Versions

Google provides several major incremental upgrades to the Android operating system in the last 15 years.

API Level: Android Studio allows to choose the minum API level to support.
You need to consider an API such that you have a balance between:

- Going low enough to support as many devices as possible
- Not going too low that you lose features your app needs to run

App building blocks

- **Resources:** layouts, images, strings, colors (XML and media files)
- **Components:** activities, services, and helper classes (Java or Kotlin code)
- **Manifest:** information about app for the runtime
- **Build configuration:** gradle config files