

Mobile Programming and Multimedia Simple (for real)



Gabriel Rovesti

Table of Contents

[2 Course Introduction 3](#_Toc160702342)

[3 Introduction to Mobile Development 6](#_Toc160702343)

[4 Frameworks for Cross-platform Development 9](#_Toc160702344)

[5 PhoneGap/Cordova Framework 14](#_Toc160702345)

[6 Corona/Solar 2D Framework 15](#_Toc160702346)

[7 Xamarin Framework 16](#_Toc160702347)

[8 Flutter Framework 17](#_Toc160702348)

[9 React Native Framework 18](#_Toc160702349)

[10 Store Deployment 19](#_Toc160702350)

[11 iOS Platform 20](#_Toc160702351)

[12 Android Platform 21](#_Toc160702352)

[13 Mobile Design 22](#_Toc160702353)

[14 Wearable Devices 23](#_Toc160702354)

[15 Multimedia Data Encoding 24](#_Toc160702355)

[16 Images 25](#_Toc160702356)

[17 Audio 26](#_Toc160702357)

[18 Video 27](#_Toc160702358)

**Disclaimer**

# Course Introduction

The course is based on two parts:

* *Mobile Programming*
  + this is the section that will change, given the nature of it
  + this year we will get to the detail of design of applications
* *Multimedia and Data*
  + this remains pretty much the same
  + this part is focused on understanding how to encode data
    - media can be particularly heavy on memory/battery/bandwidth usage
    - we want to save time in downloading and streaming the files
    - this of course because usage is very much mobile and not desktop anymore

Other general info:

* Slides will be gradually updated according to the need on the Moodle, which is free to access
  + There will be 4x4 (4 slides per page) and normal slides (1 per page)
* After 15 days, data will be collected in order to get account data of labs
* In case of need, an Apple PC is available in the library to be used to develop applications
* Recordings are available but only if attendance doesn’t go down

Quality of the application is measuring “how much it does what it’s supposed to do”. While in Web is important the number of clicks to get to the data, here it’s important how much data is asked to the user.

When creating an application, it’s important also to drive metaphors towards a common goal, giving the everyday use inside what is done inside of a product/application. Consider the desktop, which seems not a metaphor, but it is, actually.

When we are developing an application, consider this is done for smartphone, which do not have OS, but RTOS (Real-Time Operating System): for example, when we receive phone calls, an applications stops, has to save save state, then we get back to what we were doing before.

Immagine che contiene mammifero, grande felino, tigre, Grandi felini

Descrizione generata automaticamenteThe following is an example of compression; the difference is present but not very much:

To understand this, we will understand what we can or cannot perceive in light and image. This way, we can remove all the information we can’t see, without losing quality. It’s more important to know what the formulas do, instead of understanding for real the elaborations.

Representation and encoding revolves around:

* Immagine che contiene fiore, Viso umano, illustrazione, schizzo

  Descrizione generata automaticamenteHow human sight works
* Image properties: size, quality, transmission, visualization
* File formats
  + GIF
  + PNG
  + JPEG
  + JPEG2000
  + others

Also, it’s important to understand how *sound* is made:

* Audio properties: fidelity, ì transmission, playback
* Standard file formats:
  + WAV
  + MP3
  + others
* MIDI
* Compression (lossy – with loss of some data)
  + understanding what to perceive and what not

But also images with audio - *video*:

* Human vision with motion pictures
* Digital and analogic video
* Video properties: quality, representation, transmission
* Standard formats:
  + H261
  + H263
  + MPEG family
  + DivX, Xvid

We also talk about *data compression*, in particular:

* Reasons behind data compression
  + Storage space, transmission time
* Continuous and not-continuous media compression
* Lossless and lossy compression
* Lossless encoding
  + Entropy encoding methods
  + Semantic compression
* Lossy encoding
  + Image compression: JPEG
  + Video compression: MPEG1-2
  + Audio compression: MP3

Prof. says slides are not enough for the examination; Moodle material is suggested and also textbooks are needed here.

Fragmentation is present between platforms (Apple vs Android and all of its versions) and different values of settings according to devices and specific needs (e.g., brightness, virtual environment, singular devices with particular sensors/features, etc.).

This is the knowledge and skills targeted from the course:

* Mobile interface design
* Cross-platform development
* Emotional design
  + transform the user into a customer
* Wearable devices
* Market

The examination will require:

* A presentation of an argument with slides plus an oral examination

Or:

* A small group project with a final report plus a small oral examination
  + Develop an application
  + It should be made for all platforms
  + But it’s not mandatory, can be only for one specific system
  + Should be sent 5 days before the examination
  + Subscribed on Uniweb and poll of what examination was chosen
* In-depth analysis of an argument with a presentation and a small oral examination
  + 10-page essay presented on the end of the course
  + The argument must be defined by the end of April
  + Possible dates: 4th – 7th June 2024
  + It’s very important to choose topics and prepare/explain material seen during lectures

In any case, oral examination is made on two questions about all the class program and material. One can avoid these two oral questions if one attends in presence only the homeworks (exercises). These can be something like “solve a multimedia algorithm problem” or “understanding design flaws of something and explain it to the class”. If all are delivered (or at most you miss one – but at least half or more of the half), then this is considered.

Exercises will be evaluated anyway even if wrong and can be given ½ points, then asked in oral examination, when given feedback of course. It seems they will be on Tuesday afternoon.

# Introduction to Mobile Development

To create the right product, between web and mobile, we have to *study the user and understand his necessities and needs*. As we all know by now, smartphone market has been exploding since years and more and more users are active using a mobile device.

Some data we can give about them:

* Over 5 billion people are using smartphones, with 4/5 billions being active social media users/Internet users respectively
* There are at least 7 billion of mobile subscriptions worldwide
* China, India, and the United States are the countries with the highest number of smartphone users, but also Indonesia, Brazil, South America and Africa
* While desktop is mostly used inside USA, South America, Oceania
  + connections can be wireless easily not needing an infrastructure
* Operating systems are Android and iOS
  + who wants to create premium services or something that people will buy, the preferred choice is actually iOS
  + on Android we have a lot of fragmentation between features and various things
* Worldwide users are on smartphones and usage data is collected from developers themselves
  + this does not mean we have to forget desktop however
  + consider mobile devices are also tablets, not only smartphones
    - they weigh more, they cost more, we use them with two hands, and we sit down using them concentrating using it
    - this way the application must come with some way to handle the error situation, recovering from them
* There are differences between males and females
  + females use more mobile apps than desktop
* Smartphones beat TV for younger users (the younger, the more usage)
  + less gestures required, less fatigue, more content present
* There is a relationship daytime-device
  + low-to-middle use between morning and daytime for mobile
  + daytime to early evening for desktop
  + in evenings for tablets
* On average there are more than 2000 interactions with a smartphone on a day
  + consider the user can make a lot of errors because there are a lot of interruptions
    - good quality means good experience, even when errors happen
* Today we have smart\* (smartphones, smart watches, smart homes, etc.)

Mobile phones are not considered anymore as a simple device to make calls but incorporate a lot of different features. All of these ones are provided by apps, in whatever form.

* Messages, calls
* Internet navigation
* Sensor data collection and usage (app for training, biking, running, etc.)
* Agenda
* Entertainment (games, music, video, reading, etc.)

There are different *false myths*:

* Mobile app development is not expensive
  + A bad app is worse than no app
* Mobile app development is easy

On the contrary:

* Mobile app development requires big teams
* Mobile app development is not like winning at the lottery

The first step to determine if it is better to develop a mobile layout of your own website or a mobile application is to understand the differences between the two:

* Diversified content 🡪 content will be personalized remembering user preferences
* Native interface vs. Company brand
* Development time
* User interaction (ex: push notification)
  + Using gestures instead of point and click
  + User experience improves
* Access (icon) vs bookmark 🡪 the icon remembers you to use the app
* Target (loyalty vs. reach)

The only data we need from users comes from payment information, all the rest is needed *because the developer asks for it*. The website is the best way to get information in a quick way, apart from push notifications. Also, icons are suitable to do that the best way, because it *remembers the user what to do*.

Remember also *mobile e-commerce* is going strong and has a greater market share each time.

* Usually, transactions are simultaneous and can happen on multiple devices at a time and also multiple apps at the same time
  + bringing an environment together (without having to put all data again)
* Committed retailers capture more transactions on mobile apps rather than browsers
* In both platforms, iOS devices capture the majority of these retailers’ transactions
  + they will definitely spend more money

Other general statistics:

* There are millions of apps in the stores and a 25% of them are used only once
* A user usually spend 90% of the smartphone time using apps
* The 84% of the time is spent using 5 apps that change between users
  + which include, in this order, social apps, games, music, and video streaming
* It does not matter the number of downloads in the end, but the number of installations
* Study the user remember: screen time depends on different factors but also context

There is the *app vs mobile web*:

* A mobile application usually tends to encourage brand fidelity (icon on the desktop, notifications, etc.)
* A website with a mobile layout allows reaching the user in every situation, immediately

There are situations when it’s useful to create an app:

* A lot of graphics or calculations
* Camera, sensors, or microphone usage
* Gallery or contacts' access
* Push notifications or background service
* For games
* It is the only way to have access to the store

The number of installed apps changes depending on the device (and its operating system). According to Nielsen, the best approach is to interview the users to understand if they would accept to install the new app on their device:

* Storage space
* Purpose
* Loyalty

The development of a mobile app requires several *resources*:

* Interface design
* Development
  + E.g., Which operating system?
* Maintenance

There are different *advantages in web apps*:

* They require a very low knowledge base, HTML is popular
* HTML5 now provides access to almost all smartphone features
* More straightforward ‘’conversion’’ to different operating systems
* User *does not have to worry about the update* of the application
* It is not necessary to wait for application approval
  + Apple can require more than 2 weeks

Applications for mobile devices are different from desktop applications:

* Mobile operating systems are soft real-time operating systems:
  + An application can be suspended or terminated in every moment
  + The operating system manages context switch
* Only one application active
  + Not with iOS on iPad
* Limited space, it is not possible to open more windows at the same time
* Easy to install (or at least discourage less the use)
* Incredibly high number
  + To design and create an exciting app is extremely challenging
* Market fragmentation

Bibliography of this chapter: [here](https://www.business.com/articles/mobile-apps-vs-mobile-web-do-you-have-to-choose/) and [here](https://techcrunch.com/2017/05/04/report-smartphone-owners-are-using-9-apps-per-day-30-per-month/).

# Frameworks for Cross-platform Development

Once upon a time, whenever there was an idea for a new app, the main goal was to develop it for iOS.

* On iOS there are users that spend the most
* On Android there is the highest diffusion of users
  + Before 2010, there was still choice in OS other than these two
* To have a lot of income, it’s important to develop for both platforms
* In Android, there is a lot of fragmentation between devices
  + layout needs to be flexible and suitable for many of them
  + so many different manufacturers
* iOS overall is pretty much well-updated and organized
* For different OSes different languages are needed
* It is necessary to develop different apps (all the same) for several devices
  + Creating one for each operating system *by hand* is quite expensive

There are *different variables* to consider, which are independent between devices:

* Operating system
* Programming language
* Development tools (IDE, simulators, etc.)
* API
* Sensors/equipment
* Screen size
* Computational capacity

The goal is: *develop once, adapt for all*.

* Do not follow the principle of “code forking” (e.g., if iOS do this, if Android do that)
* Cross-platform frameworks for mobile development reduce market fragmentation
  + Allows to reduce negative effects
  + “Write *once*, distribute *everywhere*”

There are different *main features*:

* Application developed on time, using only one programming language
  + or a set of languages 🡪 with one language/one environment
* The chosen framework allows the distribution of the application in several applications stores
  + so, there are several applications deployed
* The frameworks usually provide support for native API

There are so many frameworks one can choose from: jQuery Mobile, jQTouch, Sencha, Sproutcore, xui, appcelerator, PhoneGap, appMobi, QuickConnectFamily, Worklight, netbiscuits, dragonRAD, pyxismobile, kony, MoSync, bedrock, LiveCode, Unity, Unreal, Adobe…. the list will go on if you will.

We move the problem from *choosing the right platform to choose the right framework*.

* Most applications are developed with frameworks, not natively
  + because of versatility and convenience) – e.g., Uber, Pinterest, etc.
* It they are developed natively, this happens because of performances reasons
* This is incredibly time-saving and cost effective for a developer
  + it needs to create effectively just one application
  + instead of one dedicated to each and every platform

Other data:

* There are billions of dollars achievable with cross-platform tools
* Different devices spread across all countries and nations
  + The most between North America, Europe, Pacific Asia, Central/South America
* The most known frameworks for this specific development paradigm are Flutter (supported by Google), React Native (supported by Meta) and Xamarin (supported by Microsoft)
  + other ones just to quote: Ionic, Corona, Sencha, Unity, etc.)

There are different *pros and cons to cross-platform development*:

* Pros
  + Wide market reach
  + Single codebase
  + Faster and cheaper deployment
  + Reduced workload
  + Platform consistency
* Cons
  + Possibly slower performance
  + UX and UI discrepancies (create widgets suitable for the platform)

There are also *pros and cons for native development*:

* Pros
  + Usually, a native application offers a better user experience, a faster and more high-performance interaction
  + Non-native applications are limited by the expressivity of the used framework
    - e.g., available APIs
  + An Apple computer is always needed
* Cons
  + Fragmentation = higher development costs
  + Problems with test

Good question: *How to choose the best framework?* App development involves 4 steps:

1. Idea analysis
2. Interface design
3. App development
4. Store deployment

Consider in particular:

* Store deployment is necessary every time there is an update and for each platform
* Native development requires repeating steps 2-3-4 for each platform

Class discussion with Wooclap: *What are the features that influence the choice of a cross-platform mobile development?*

* Documentation and good references
* Third party support
* Development cycle time and maintenance
* Cost of framework and license prices
* Energy consumption requirements
* Good learning curve for the language
* Support for native look and feel
* Community and good support
* Compatibility between different platforms

Frameworks’ classification is still an open problem. Raj and Tolety classification define 4 different classes (paper is inside the fundamental material – aka need to study/read it, [here](https://ieeexplore.ieee.org/document/6420693)):

* *Web* Approach
  + *General features*
    - Immagine che contiene testo, Cellulare, gadget, schermata

      Descrizione generata automaticamenteThis is not a mobile app, instead it’s a web application accessible via URL
    - Mainly developed using HTML, CSS, JS
    - It executes a web service to obtain a native version of an app
    - It does not require an installation, easier to update without manual intervention
  + Pros
    - Same interface (but not same experience) on all devices
    - No installation necessary
    - Easy update and maintenance-free
  + Cons
    - No store publishing
    - Network connection necessary
    - Difficult test
      * Cannot access mobile device hardware and software
      * Difficult to support different screen resolutions this way
    - Strongly connected to HTML5 support of the device
      * Widgets with native look and feel for applications
      * Less control over content rendering
      * Limited to leverage the gestures offered by the platforms
    - Non-native interfaces bring to low usability
    - More difficult to monetize
* This is the concept of Progressive Web App (PWA), which are web pages that behave like native applications
  + The term was coined by Steve Jobs in 2007, since apps using new functionalities like service workers and web app manifests needed to be categorized
* In particular:
  + They are developed using web technologies, therefore HTML5, CSS3, Javascript
  + It works independently from the browser
    - using *progressive enhancement* (according to the device equipment)
    - the more features the browser provides, the more features provides the app
  + It works even offline, but with limited support
  + Can be installed without using the store (but in this case, they are a sort of link)
  + Like every web page, these apps adapt themselves to device size (responsive)
  + Secure (HTTPS) and indexed by search engines
  + Easy to update
  + Support push notifications
  + No need for stores to publish the app
    - but there is no payments management
    - and there is no control of what is published
* Examples: Sencha Touch, Angular, React (note: not React Native), jQuery Mobile
* *Immagine che contiene testo, schermata, diagramma, design

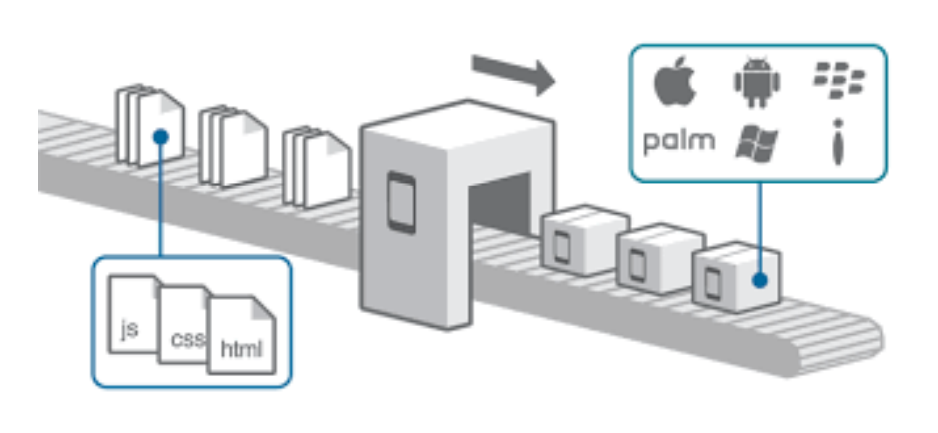
  Descrizione generata automaticamenteHybrid* Approach
  + This is developed using web technologies and gets executed inside native container on the mobile device
  + Uses the browser engine of device to render/display HTML content full screen
  + Separate in layers
    - Hybrid application
    - JavaScript Abstraction Layer
      * Allows to expose device capabilities
    - Native Library
    - Web Services
  + Pros
    - Store publishing available
    - Reusable UI
    - Usage of device components and powered by device computing capabilities
    - Usable for both server backed and standalone applications
  + Cons
    - Need to be installed on devices
    - Lower performances compared to native apps
    - Has cross-communication vulnerabilities because of JS
    - UI do not follow native Look and Feel hence styling is required
* Examples: PhoneGap/Cordova
  + The project started in 2008 trying to solve these problems
    - Development of mobile applications using web technologies
    - Solve the problem of low support of mobile browsers to HTML5
    - Allow access to unique features of the device
  + Actual support to HTML5 of the mobile browsers
    - and HTML5 evolution has partially solved these problems
  + In 2011 PhoneGap code was offered to Apache to continue the development
  + Apache Cordova is the engine below PhoneGap
    - like WebKit is the engine of several browsers

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

* Apache Cordova framework is a hybrid framework
  + Applications development works with HTML, CSS and JS, well known to web devs
  + It uses plugins to access hardware components of the smartphone (camera, GPS, etc.)
  + It provides tools for testing (emulators) and deployment of the final application
  + Figure on the right shows an example of *HelloWorld* in this framework
* Other frameworks/tools allow app development using Cordova:
  + Monaca – Framework7 – NativeScript – Ionic Capacitor – Progressive Web Apps
  + Really easy to use for web developers to develop mobile applications
  + Quite easy to customize interface using CSS
* Cordova usually is not used stand-alone
  + but as a support framework for other frameworks

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

* *Interpreted* Approach
  + Pros
    - Native Look and Feel
    - Store publishing
    - APIs for device components
  + Cons
    - Really difficult to reuse the UI
    - Available features depend on the framework
    - The interpreter can have low performances
* Immagine che contiene testo, schermata, Carattere, Rettangolo

  Descrizione generata automaticamente*Cross-compiled* Approach
  + Pros
    - Allows to use all the components available from native app
    - Native interface
    - Good performances
    - Store publishing
  + Cons:
    - Not reusable UI
      * Specific to the platform
    - Overly complex apps can have problems during the building process
    - Identifying/correcting cross compilation issues might be hard

(For your info: course started at the end of February, second week of lessons, prof. says she will release one of the homework assignments next week. I like people being organized, so I tell you straight away).

# PhoneGap/Cordova Framework

# Corona/Solar 2D Framework

This is made for animations/games pretty much close and not used that much outside of college environment.

# Xamarin Framework

# Flutter Framework

# React Native Framework

# Store Deployment

# iOS Platform

# Android Platform

# Mobile Design

# Wearable Devices

# Multimedia Data Encoding

# Images

# Audio

# Video