The Basics of iOS Development

A quick overview of the Apple way of building apps...

A bit on the evolution of iOS

- first iPhone launched in 2007
- AppleTV in 2007
- iPhone 4 in 2010
- developing UI using Storyboards in 2010
- Swift announced in 2014
- Apple Watch, in 2014
- Face ID 2017
- Combine and SwiftUI in 2019

The development environment

- you need a Mac to develop iOS apps
- XCode
  - almost everybody uses it
  - compiles, debugs, runs the tests
  - integrated with the instruments for analysing your app
- you can use an iOS Simulator if you don't have the device
- a bit difficult to distribute the app to other phones

Building the User Interface

- first there were the XIBs
- then came the Storyboards and Constraints
- the old way of doing the UI is using UIKit classes like
  - UIViewController
  - UITableViewController
  - UICollectionViewController
- then came SwiftUI (analog to Jetpack Compose)
- other frameworks commonly used:
  - CoreGraphics
  - CoreAnimation
  - Metal

Lifecycle and Navigation

- most commonly overridden methods in UIViewController:
  - viewDidLoad
  - viewWillAppear
  - viewDidAppear
  - viewWillDisappear
- navigation can be done by:
  - navigation controller, that manages the stack
  - presenting modally from the UIViewController
  - using Storyboard segues

Data persistence

- UserDefaults: for app preferences
- Documents directory: the equivalent of the internal storage
- Keystore: for encrypted data
- CoreData: for structured data
- there is no external storage, but you can use something like "File Sharing"

Testing

- very important, but sometimes overlooked
- unit testing, with XCTest
- UI testing
- snapshot testing, but not part of XCode
- can use other 3rd party frameworks

Comparison between iOS and Android

Objective-C	Java
Swift	Kotlin
UIKit, XIBs, Storyboards	Activities, XMLs
Swift Concurrency, async & await	Coroutines
Swift UI	Jetpack Compose
Combine	Flow
XCode	Android Studio
XCode	Gradle

Some differences between iOS and Android

- fewer devices on iOS -> easier to test
- most users update their devices to the latest OS -> can drop support for earlier versions
- no services in iOS, but you can achieve something similar with Background Modes
- stricter control from Apple, harder to install apps that are not in App Store

Architecture patterns

- by default, Apple uses a lot of the MVC pattern, and delegation pattern
- in practice there are other patterns like MVVM, VIPER
- also, reactive programming is very popular, with frameworks like RxSwift or Combine
- SOLID principles also apply to mobile development

Deploying your app

- you must pay the annual 99 USD fee
- for each app you create a provisioning profile
- each app has an unique App ID
- how to do it:

1. build the IPA
2. upload to testflight
3. wait for the approval
4. publish in App Store

Creating the first app

Doing a couple of the labs in XCode, with iOS.

- setting up the first project
- creating the hello world app
- doing some screen navigation
- adding a dependency
- creating a request

Where to go to for more information?

- Stanford's CS193p course
- Caio & Mike, good talks about architecture
- Bart Jacobs, with a couple of sample apps
- Swift by Sundell
- iOS dev weekly
- Apple's annual WWDC

- Robert Martin (aka Uncle Bob) about Computer Science, in general