Capitolo 1

Installazione

In questo capitolo

- **Flutter doctor**
- **Android**
- **iOS**
- **Emulazione**

Mettere in piedi un ambiente di sviluppo che consenta di utilizzare Flutter non è poi così difficile. La complessità di un'attività di questo tipo è strettamente collegata a diversi fattori: il tipo di applicativo che vogliamo realizzare, il sistema operativo che ci ospita, il sistema operativo target, tutto quello che ruota intorno al framework che stiamo utilizzando e tutte le dipendenze richieste. Un applicativo mobile, cross platform, non è di certo paragonabile a una landing page, ma rispetto ad altre soluzioni Flutter ci darà una mano anche in questo.

L'obiettivo di questo capitolo è spiegare piuttosto velocemente in che cosa consiste l'installazione dell'ambiente. Eviteremo di andare troppo nel dettaglio: per quello affidatevi, di certo senza sbagliare, al sito ufficiale, `https://flutter.dev/docs/get-started/install`.

Detto questo, iniziamo a sporcarci le mani.

Il primo passo da fare è procedere al download di flutter SDK.

NOTA

Software Development Kit (SDK) è una suite di tools realizzati per lo sviluppo di software.

NOTA

I possessori macOS con processori Apple Silicon, per esempio M1, necessitano di una dipendenza aggiuntiva (`/usr/sbin/softwareupdate --install-rosetta --agree-to-license`), oltre a una versione di SDK Flutter dedicata.

Terminato il download, procediamo a scompattare l'archivio e a registrare il path di Flutter affinché possiamo richiamarlo in semplicità da qualsiasi punto del filesystem.
Fate attenzione a eseguire il comando ponendo un livello sotto la directory `flutter` appena scompattata.
Per Mac e Linux:

```
# export PATH="$PATH:`pwd`/flutter/bin"
```

> **NOTA**
>
> Con il comando export impostiamo il path solo all'interno dell'istanza della console. Per rendere permanente il path, possiamo inserire il path assoluto nel file `/etc/paths`.

> **NOTA**
>
> Per Linux, nel caso in cui l'utente utilizzi bash, saprà che è sufficiente aggiungere il path di Flutter nel file: `$HOME/.bashrc`.

Per Windows:
Dopo aver scompattato il file archivio di Flutter, sarà necessario registrare il suo path tramite il comando *env* che può essere eseguito dal menu Start. È possibile perseguire l'aggiunta del path di Flutter tramite l'uso del tool: *Edit environment variables for your account*.

Flutter doctor

Terminata l'installazione, proseguiamo con la configurazione dell'ambiente di sviluppo. Il passo successivo si basa sull'uso del tool `doctor` che consente di verificare a che punto si trova la nostra installazione.

```
# flutter doctor
```

```
A new version of Flutter is available!

To update to the latest version, run "flutter upgrade".
```

```
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.7.1, on macOS 12.3.1 21E258 darwin-x64, locale it-CH)
[✗] Android toolchain - develop for Android devices
    ✗ Unable to locate Android SDK.
      Install Android Studio from:
      https://developer.android.com/studio/index.html
      On first launch it will assist you in installing the Android SDK
      components.
      (or visit https://flutter.dev/docs/get-started/install/macos#android-setup
      for detailed instructions).
      If the Android SDK has been installed to a custom location, set
      ANDROID_SDK_ROOT to that location.
      You may also want to add it to your PATH environment variable.

[✗] Xcode - develop for iOS and macOS
    ✗ Xcode installation is incomplete; a full installation is necessary for iOS
      development.
```

```
      Download at: https://developer.apple.com/xcode/download/
      Or install Xcode via the App Store.
      Once installed, run:
        sudo xcode-select --switch /Applications/Xcode.app/Contents/Developer
        sudo xcodebuild -runFirstLaunch
    ✗ CocoaPods not installed.
        CocoaPods is used to retrieve the iOS and macOS platform side's plugin
        code that responds to your plugin usage on the Dart side.
        Without CocoaPods, plugins will not work on iOS or macOS.
        For more info, see https://flutter.dev/platform-plugins
      To install:
        sudo gem install cocoapods
[!] Android Studio (not installed)
[!] VS Code (version 1.50.1)
    ✗ Flutter extension not installed; install from
      https://marketplace.visualstudio.com/items?itemName=Dart-Code.flutter
[!] Connected device
    ! No devices available

! Doctor found issues in 5 categories.
```

In questo caso, Flutter ci notifica che dovremmo procedere a un aggiornamento per via di una release più recente. Procediamo pertanto all'aggiornamento tramite il comando:

```
# flutter upgrade
```

Terminato questo step, è necessario preoccuparsi delle dipendenze mancanti. In questa specifica installazione ci viene notificato che mancano Android SDK e Xcode, entrambi necessari per sviluppare app per Android e iOS. Inoltre, dovremo anche gestire l'IDE che useremo, ma affronteremo il tema tra un attimo.

> **NOTA**
>
> Il comando `channel` consente di scegliere tra i canali di release: stable, beta e master. Il consiglio è verificare ed eventualmente scegliere il channel stable, che rappresenta la linea di sviluppo da usare per la produzione, in quanto è la più stabile.

Android

Come anticipato, Flutter ci notifica la mancanza della toolchain Android.

> **NOTA**
>
> Toolchain è un nsieme di software che lavorano in sinergia per lo sviluppo del software.

Il primo passo da fare è procedere al download di Android Studio (`https://developer.android.com/studio`). Android Studio è un IDE sviluppato da Google necessario allo sviluppo Android.

> **NOTA**
>
> IDE è un software che fornisce un ambiente di sviluppo per uno o più linguaggi di programmazione.

Come mostrato in Figura 1.1, il wizard di installazione procede per conto nostro, installando le dipendenze necessarie per il corretto uso dell' IDE Android Studio e di conseguenza per Flutter.
Dopo l'installazione di Android studio, proseguiamo nell'accettare la licenza legata ad Android tramite il comando flutter:

```
# flutter doctor --android-licenses
```

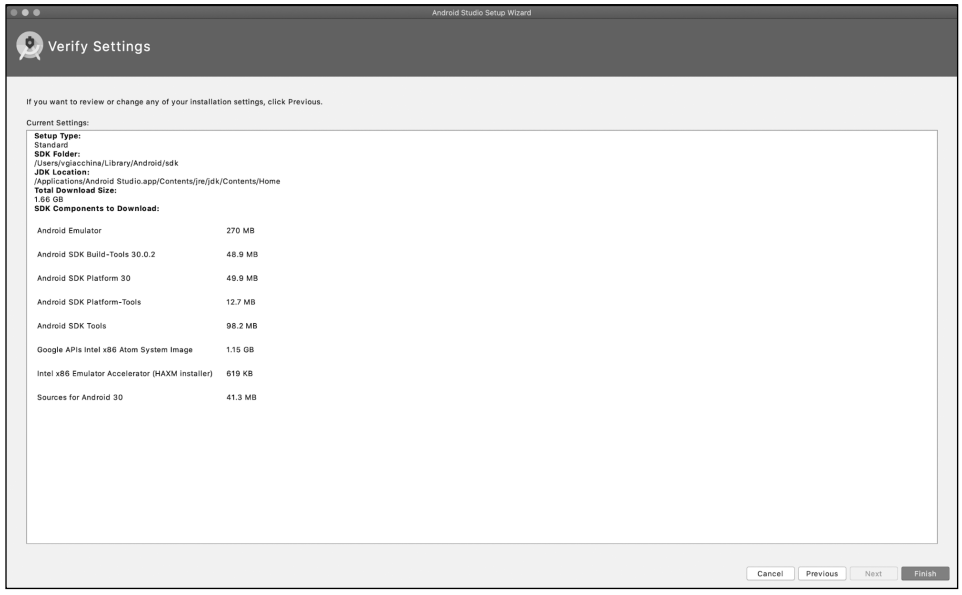

Figura 1.1 Android Studio Setup Wizard.

Nel caso in cui il comando dia un errore relativo a un tool mancante: sdkmanager, procediamo facendo attenzione che la command line tools dell'sdk sia installata. È possibile verificare questa dipendenza in Android Studio, nel tab SDK Tools nell'SDK Manager.
Ora, se rilanceremo il comando `flutter doctor`, potremo leggere che siamo pronti per lo sviluppo in Android, avendo installato la toolchain.

```
[✓] Android toolchain - develop for Android devices (Android SDK version 30.0.2)
```

Procediamo integrando Flutter, tramite plugin, con Android Studio per soddisfare i seguenti requisiti:

```
[!] Android Studio (version 4.1)
    ✗ Flutter plugin not installed; this adds Flutter specific functionality.
    ✗ Dart plugin not installed; this adds Dart specific functionality.
[!] VS Code (version 1.50.1)
    ✗ Flutter extension not installed; install from
      https://marketplace.visualstudio.com/items?itemName=Dart-Code.flutter
```

Per risolvere anche questa dipendenza, sarà necessario avviare Android Studio e procedere all'installazione del plugin Flutter e Dart.

Figura 1.2 Plugin Flutter per Android Studio.

Se volessimo utilizzare un IDE differente, Flutter supporta anche VS Code (Visual Studio Code), che è quello che attualmente uso con soddisfazione e vi consiglio caldamente. Nel mio caso, Flutter ha rilevato l'installazione di Visual Studio, ma attualmente non supporta Flutter.
Avviamo VS Code, oppure installiamolo se non è già presente nel nostro ambiente, (è disponibile all'indirizzo `https://code.visualstudio.com/`) e tramite le preference procediamo a installare le estensioni Flutter e Dart.
Adesso la procedura di installazione per Android risulta essere terminata e `flutter doctor` lo confermerà.

```
[✓] Flutter (Channel stable, 1.22.3, on Mac OS X 10.15.5 19F101, locale it-CH)
[✓] Android toolchain - develop for Android devices (Android SDK version 30.0.2)
[✓] Android Studio (version 4.1)
[✓] VS Code (version 1.50.1)
```

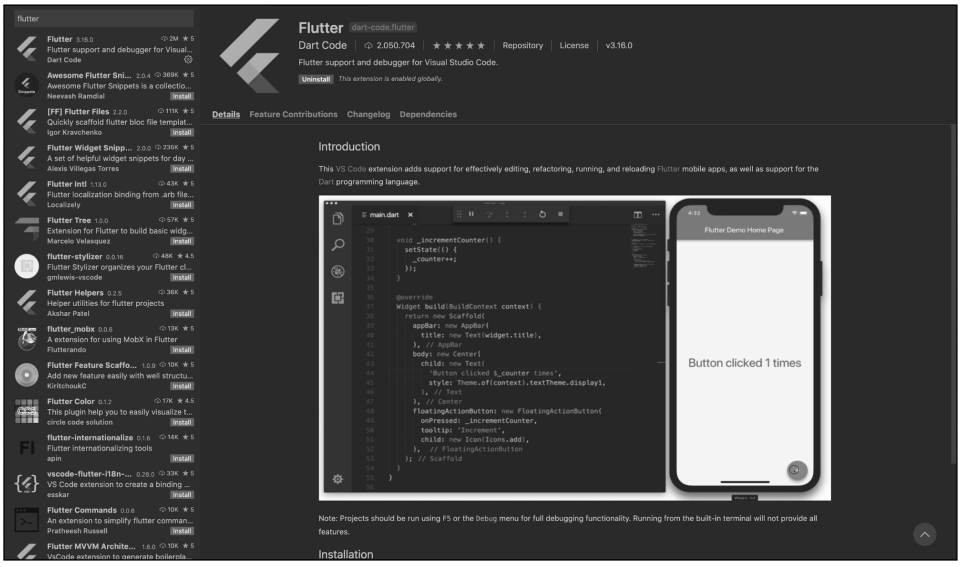

Figura 1.3 Plugin Flutter per VS Code.

NOTA

Per utilizzare il device fisico Android durante il processo di sviluppo è necessario procedere all'installazione del driver Google USB, reperibile a questo indirizzo: `https://developer.android.com/studio/run/win-usb`.

iOS

Come ben sappiamo, e se così non fosse potreste tranquillamente immaginarlo dalle politiche Apple, per sviluppare app per iOS è necessario avere un Mac, proprio perchè il corrispettivo di Android Studio per iOS è Xcode, e quest'ultimo è disponibile solo per Apple.
Procediamo all'installazione di Xcode tramite Apple Store (`https://developer.apple.com/xcode/`).
Una volta ottenuto l'IDE, procedere avviandolo per accettare la licenza e l'installazione delle dipendenze, che verrà fatta dal software automaticamente. Insieme a Xcode sarà necessario installare anche CocoaPods.
CocoaPods (`https://cocoapods.org/`) consente di gestire le dipendenze; nel caso di Flutter potrebbe tornare utile nel caso si usino plugin che necessitano di dipendenze.

```
# sudo gem install cocoapods
```

Una volta terminata l'installazione, eseguiamo Xcode.app (può essere eseguito nelle nostre applicazioni). Se prima dell'avvio ci venisse chiesto di aggiornarlo o scaricare delle dipendenze, proseguiamo pure dando il nostro consenso. Una volta avviato, ci apparirà una finestra come quella in Figura 1.4.

Figura 1.4 Xcode.

Emulazione

La connessione di un device mobile, o la sua emulazione, è parte fondamentale nella preparazione dell'ambiente di sviluppo; infatti, `doctor` lo segnala inequivocabilmente:

```
[!] Connected device
    ! No devices available
```

L'approccio alla creazione di un device virtuale, che altro non è che un simulatore, sarà differente per OS. Nel caso di Android, per semplicità procediamo alla configurazione tramite Android Studio. È possibile procedere anche tramite il tool offerto dall'SDK via command line.

ADV

È comunque possibile creare e gestire i device virtuali tramite il tool emulator offerto da Android SDK, installato precedentemente. Se il path dell'SDK Android vi fosse sfuggito durante il wizard di installazione di Android Studio, è comunque consultabile eseguendo Android Studio e selezionando l'SDK Manager nella lista delle *Preferences* (Figura 1.5). Tramite la GUI di Android Studio è possibile lanciare l'interfaccia Android Virtual Device Manager. Di default dovremmo già trovare un device; infatti, avviandolo tramite l'icona di start e procedendo a lanciare `flutter doctor`, noteremo che adesso il device viene rilevato con successo.

```
[✓] Connected device (1 available)
```

Nel caso incontraste problemi nell'avvio del device vi consiglio di controllare lo spazio sufficiente su disco, in quanto il software da installare non è poco.

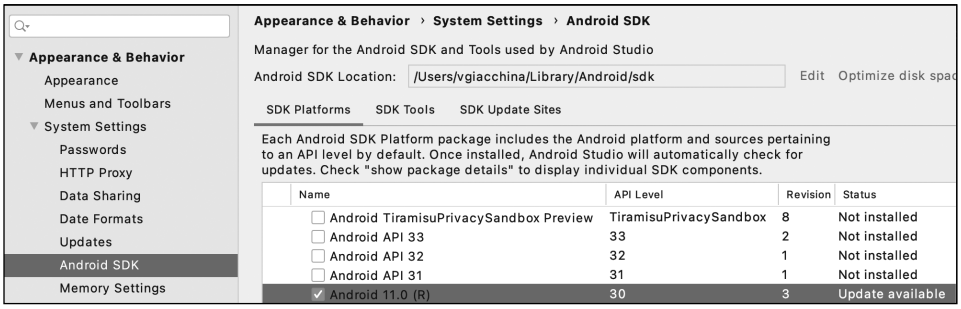

Figura 1.5 SDK Manager di Android Studio.

In questa istanza di installazione, procediamo pure al download e di conseguenza alla configurazione del device virtuale di Android API 33, l'ultima versione Android attuale. Le API Level di Android, come è possibile intuire dalla Figura 1.6, sono specifiche per ogni versione del sistema operativo Android. Se utilizzassimo le API 33, faremmo riferimento alle API offerte dal sistema operativo Android 13, conosciuto come Tiramisù. Le API dell'SDK Android offrono un innumerevole set di interfacce per interagire con il sistema operativo Android e garantiscono la retrocompatibilità, per questa ragione iniziare a sviluppare con l'ultima versione stabile delle API Level di Android è cosa giusta e consigliata.
Come mostrato in Figura 1.6, dall'interfaccia di AVD potete sbizzarrirvi a creare i device virtuali che preferite; faremo qualche altro esempio lungo il percorso.
Nel caso in cui si volesse approcciare la gestione dei device Android da command line, questo sarebbe possibile tramite il tool: `avdmanager`.

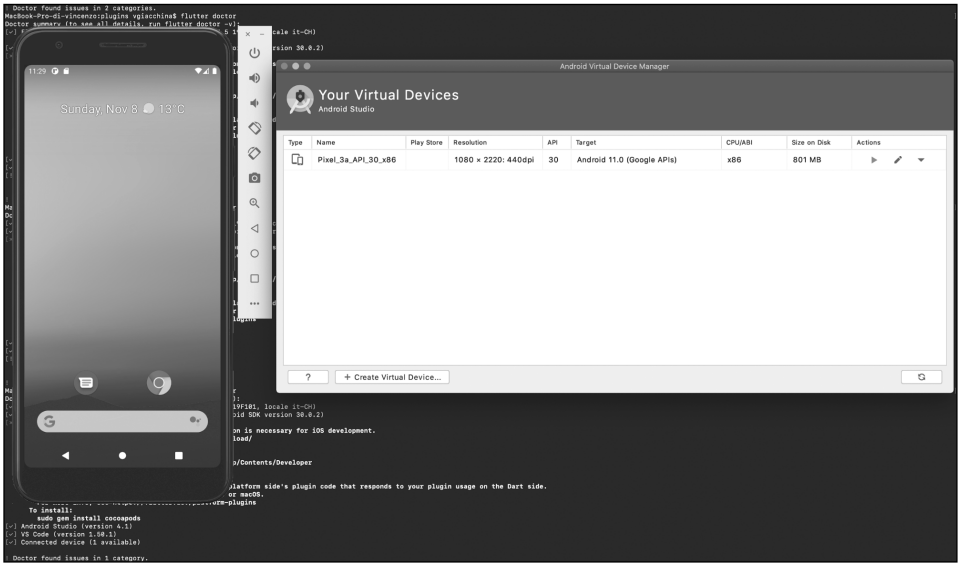

Figura 1.6 Device virtuale in esecuzione.

È importante, nel caso in cui invece si volesse utilizzare un device Android fisico, abilitare la modalità debug, affinché il device venga rilevato; ciò consente di eseguire l'applicazione Android sul vostro device. Potete visionare la documentazione ufficiale per abilitare questo supporto all'indirizzo `https://developer.android.com/studio/debug/dev-options`. Brevemente, in genere, si procede ad abilitare la modalità debug tramite il menu *Impostazioni,* su *Info telefono* e poi facendo clic ripetutamente sul numero di build di Android, fino a ottenere l'attivazione del menu *Opzioni sviluppatore*, dove sarà possibile attivare il debug USB. Fatto ciò, il vostro IDE sarà in grado di eseguire l'applicazione sul device fisico Android.

Simulator

L'installazione di Xcode si porta con sé il tool simulator. L'emulatore di Xcode è facilmente eseguibile una volta creato il nostro progetto all'interno di Xcode. In questo caso, opteremo per eseguire direttamente il simulatore, fuori lo scope di Xcode, proprio per verificare che tutto l'ambiente è disponibile.

> **NOTA**
>
> La differenza tra emulatore e simulatore è che il primo si prefigge lo scopo di emulare anche le risorse hardware, a differenza dei simulatori che simulano esclusivamente il sistema operativo. Pertanto risulta sempre fondamentale, anche in fase di sviluppo, il supporto del device fisico per effettuare i nostri test.

Cerchiamo nel nostro finder il tool Simulator.app ed eseguiamolo.Verrà immediatamente simulato un iPhone (vedi Figura 1.7). In ogni caso è possibile, molto semplicemente, scegliere quale device virtuale avviare.
Flutter, tramite `doctor`, potrà confermarci la connessione al device virtuale.

```
# flutter doctor -v
…
[✓] Connected device (3 available)
    • iPhone 13 mini (mobile) • 4C25CEB5-6DA1-4D4D-9038-D0AFD725DA41 • ios            •
com.apple.CoreSimulator.SimRuntime.iOS-15-0 (simulator)
```

Rispetto a quanto visto per Android, il deploy sul device fisico iOS segue una logica differente con passaggi differenti. Nonostante questo, è necessario, anche sul device Apple, attivare la modalità developer e fidarsi del certificato del Mac a cui stiamo collegando il nostro device. Aimè, è necessario procedere all'acquisto di un account sviluppatore Apple ed è anche necessario configurare i certificati di sviluppo e distribuzione associati al vostro account sviluppatore Apple.

> **NOTA**
>
> In fase di sviluppo, non è necessario acquistare un account da sviluppatore per lanciare l'app sul device fisico. Possiamo utilizzare semplicemente l'Apple ID creando con esso un account in Xcode, sezione *Account* in *Preferenze*. Fatto ciò, utilizzeremo questo account come Team nella sezione identity di Xcode.

Non tratterò questo tema per mancanza di tempo, e purtroppo vi anticipo che richiede un pò di impegno, in quanto Apple, rispetto ad Android, ha implementato una gestione

del deploy per il mondo Apple che non è semplice come la principale concorrenza. Vi affido a questa guida `https://docs.flutter.dev/deployment/ios` e al sito ufficiale per lo sviluppo Apple: `https://developer.apple.com/`.

Figura 1.7 IPhone 13 mini virtualizzato.