- Salvatore Ferranti, Francesco Ingrassia, Marco Passariello -

Sviluppare applicazioni per Apple Watch

con watchOS 2

Design e implementazione di app reattive e attraenti >>

Progetti di esempi in doppio linguaggio Objective-C/Swift >>

Tutte le novità di watchOS 2 >>

Best practice e trucchi avanzati >>

*pro DigitalLifeStyle

*pro
Digital*LifeStyle

Sviluppare applicazioni per Apple Watch con watchOS 2

Salvatore Ferranti
Francesco Ingrassia
Marco Passariello

EDIZIONI
LSWR

Sviluppare applicazioni per Apple Watch | Con watchOS 2

Autori: Salvatore Ferranti, Francesco Ingrassia, Marco Passariello

Collana: *pro DigitalLifeStyle

Editor in Chief: Marco Aleotti
Progetto grafico: Roberta Venturieri
Immagine di copertina: Fabio Prati
Realizzazione editoriale e impaginazione: Studio Dedita di Davide Gianetti
Redazione: Stefano Andreini

ISBN: 978-88-6895-282-2

EDIZIONI LSWR

Via G. Spadolini, 7
20141 Milano (MI)
Tel. 02 881841
www.edizionilswr.it

Printed in Italy

Finito di stampare nel mese di febbraio 2016 presso "LegoDigit" Srl., Lavis (TN)

(*) Edizioni Lswr è un marchio di La Tribuna Srl. La Tribuna Srl fa parte di LSWR GROUP.

Sommario

Prefazione

Che si ritenga l'Apple Watch l'inizio concreto del pervasive computing o un poderoso e inesplorato canale di digital marketing oppure, più semplicemente, che si pensi a Gizmo come un modo per vivere appieno il proprio smartphone, è indiscusso che ogni tecnologo – professionista o geek che sia – non possa non conoscerne il wording, le funzionalità e l'architettura software e hardware.
La trasformazione più straordinaria di questi device si è svolta solo negli ultimi due anni, sotto i nostri occhi non sempre attenti a questo fenomeno e, ora, il mondo dei wearable è qui per noi, pronto per essere esplorato.
Questo libro è nato proprio per questo scopo. Si tratta di un volume ricco di esempi, completo e attuale, in grado di guidare e rendere velocemente produttivi tutti coloro che volessero sviluppare applicazioni per Apple Watch.
Nato dalla "penna" di tre profondi conoscitori della piattaforma watchOS, il libro indica chiaramente pattern di sviluppo, interazione e design già testati su applicazioni business critical e distribuite a un vasto pubblico.
Seguendo il naturale filo conduttore del testo, lo sviluppatore potrà evitare i classici errori che si incontrano ogni qual volta ci si approccia a una nuova tecnologia e costruirsi una solida reference guide per affrontare in tranquillità ogni progetto.

Carlo Romagnoli
Digital Products Director
Information Technology
Sky Italia

Introduzione

Tim Cook, il 9 settembre 2014, ha svelato al mondo il primo dispositivo wearable di casa Apple: l'Apple Watch. Dalla pioggia scrosciante di applausi che l'amministratore delegato di Apple ha ricevuto al termine del filmato di presentazione si era già capito che il prodotto, una volta uscito sul mercato, avrebbe riscosso molto successo.
Un paio di mesi dopo, la "casa della mela" ha reso disponibile il kit di sviluppo di applicazioni per Apple Watch, che consisteva nel framework WatchKit e nel sistema operativo watchOS 1, entrambi distribuiti all'interno di una versione beta di Xcode.
Al momento della sua apparizione al pubblico, nell'aprile del 2015, lo sviluppo di applicazioni su Apple Watch era molto limitato e poteva avvenire solo se il dispositivo veniva collegato direttamente all'iPhone. I limiti della piattaforma sono subito apparsi evidenti tanto che, solo due mesi dopo, durante il WWDC 2015 è stato annunciato watchOS 2; esso apportava notevoli cambiamenti all'architettura di Apple Watch, rimuovendo le limitazioni della prima versione e aggiungendo la possibilità di:

- effettuare chiamate di rete direttamente dal dispositivo senza passare per l'iPhone (in watchOS 1 era sempre il telefono a comunicare con la rete e a trasferire i dati);
- eseguire il codice dell'applicazione quasi interamente sull'orologio (a meno di casi specifici che vedremo più avanti). In watchOS 1, invece, era l'iPhone stesso a occuparsi della logica implementativa e a comunicare all'orologio le modifiche da applicare all'interfaccia grafica;
- usufruire dei sensori per registrare il battito cardiaco e il movimento;
- esporre i dati dell'applicazione e permettere all'utente di includerli nelle watch face dell'orologio (una watch face non è altro che la vista in cui l'utente visualizza l'ora e altre informazioni).

A chi si rivolge questo libro

Il libro si rivolge agli sviluppatori di applicazioni iPhone che possiedono una conoscenza media della programmazione e degli strumenti del mondo iOS e che abbiano già usato uno dei due linguaggi di sviluppo, Swift (in particolare in questo libro faremo esempi usando Swift 2) o Objective-C, per la realizzazione di un'applicazione. Un'utenza, insomma, che voglia ampliare il proprio raggio d'azione, intraprendendo lo sviluppo di applicazioni per Apple Watch. Chi ha già sviluppato applicazioni per watchOS 1 può trovare, in questo libro, un'utile guida per il passaggio di piattaforma.

Struttura del libro

Il libro si compone di otto capitoli. In ognuno di essi approfondiremo un aspetto diverso delle funzionalità che Apple ha messo a disposizione degli sviluppatori per il proprio device Apple Watch.

- **Capitolo 1:** il Capitolo 1 fornisce una descrizione dell'architettura software di watchOS 2, delle sue differenze rispetto a watchOS 1 e presenta le nozioni base per poter intraprendere lo sviluppo di un'applicazione su piattaforma Apple Watch. Si mostra come creare un progetto da zero e vengono illustrati i punti più importanti da considerare nel processo di migrazione da un'applicazione su watchOS 1 a una su watchOS 2;
- **Capitolo 2:** nel Capitolo 2 costruiremo la nostra prima applicazione descrivendo in maniera puntuale i principali componenti offerti da watchOS 2 e ricreando lo scenario tipico di uno sviluppatore che, dopo aver completato la propria applicazione iOS, decida di fornire agli utenti anche una versione per Apple Watch;
- **Capitolo 3:** una Watch App non è l'unico modo che ha uno sviluppatore per interagire con i propri utenti. Notifiche customizzate e Glance rappresentano due ottime soluzioni per arricchire l'esperienza d'uso dei propri customer. Il Capitolo 3 analizza e approfondisce queste due opzioni;
- **Capitolo 4:** il Capitolo 4 esplora le regole di design che Apple consiglia di seguire per garantire all'utente un'esperienza consistente all'interno della propria applicazione. Inoltre definiremo ed elencheremo le risoluzioni degli elementi grafici che compongono l'applicazione;
- **Capitolo 5:** il meccanismo di comunicazione tra le applicazioni ha subito un grosso cambiamento su watchOS 2: è stato introdotto un nuovo framework (Watch Connectivity) che si occupa della comunicazione tra le applicazioni. In questo capitolo esponiamo le basi per instaurare una comunicazione tra un'applicazione iOS e una Apple Watch;

- **Capitolo 6:** come per ogni orologio che si rispetti, anche Apple ha deciso di integrare maggiori funzionalità sulle watch face, aggiungendo la possibilità di visualizzare il prossimo appuntamento o le informazioni meteo. Nel Capitolo 6 approfondiremo il sistema che sta dietro questo meccanismo di Complications e impareremo come far sì che l'utente resti sempre allineato alle modifiche dei dati della nostra applicazione;
- **Capitolo 7:** in questo capitolo vengono descritti i sensori presenti all'interno di Apple Watch e il modo in cui essi possono essere usati per ottenere le informazioni sui dati raccolti;
- **Capitolo 8:** in questo capitolo vengono mostrate alcune tecniche avanzate sull'utilizzo di alcuni dei componenti già presentati nel Capitolo 2; inoltre, vengono presentati nuovi elementi introdotti con watchOS 2.

Link del libro

Sito web del libro:
https://guidawatchos2.wordpress.com

Codice sorgente:
https://github.com/vdmAuthors/guidawatchos2-samples

Email:
vdmauthors@gmail.com

Twitter:
@guidawatchos2

Ringraziamenti

Francesco

A Francesca, per aver saputo dire la cosa giusta al momento giusto.

Marco

Ringrazio i miei genitori per essermi stati sempre vicini nei momenti di difficoltà e, soprattutto, per avermi incoraggiato durante la stesura di questo libro.

Salvatore
Ringrazio mia moglie Laura che mi ha supportato (e sopportato) e mi è stata vicino durante la stesura di questo libro. Un grazie particolare va a mia sorella Roberta, senza la quale tutto questo oggi non sarebbe stato possibile. Grazie Roberta.

Un ringraziamento speciale va a Stefano Sanna e Roberto Fraboni, senza i quali questo libro non sarebbe mai stato possibile. Ci teniamo a ringraziare anche Matteo Bonifazi per averci aiutato e guidato nella stesura di questo testo. Grazie a tutto il team mobile di Open Reply che, giorno dopo giorno, alimenta la nostra passione per quello che facciamo.

1

Architettura di watchOS 2

Con il rilascio di watchOS 2 Apple ha cambiato le carte in tavola per gli sviluppatori, andando a modificare l'architettura di base per lo sviluppo delle applicazioni su Apple Watch. In questo capitolo parleremo delle differenze tra le prime due versioni di watchOS, soffermandoci in particolare sull'architettura della versione 2 e sulle novità apportate rispetto alla versione 1.

Prima di concentrarci sulle modifiche architetturali presentate da Apple con la nuova versione del sistema operativo di Apple Watch, è necessario fare un rapido accenno all'architettura della prima versione di watchOS; in questo modo si potranno comprendere appieno i vantaggi apportati dalla nuova versione.

Architettura di watchOS 1

Come per la prima versione di iOS, anche la prima di watchOS era molto limitata. Mancavano all'appello componenti fondamentali come l'accesso ai sensori o la possibilità di interagire con la corona digitale dell'orologio.
Di seguito presentiamo una panoramica dell'architettura della prima versione di watchOS per aiutare il lettore a comprendere perché Apple abbia deciso di rilasciare un sistema operativo che, di fatto, non ha saputo soddisfare pienamente le aspettative degli sviluppatori.

Con l'uscita di iOS 8, Apple ha introdotto il concetto di *Extension*, ovvero una modalità che permette agli sviluppatori di rendere disponibili all'esterno della propria applicazione le funzionalità dell'applicazione stessa. L'architettura di watchOS, in entrambe le versioni, si basa proprio sul concetto di *Extension*: l'applicazione presente su Apple Watch è un'estensione dell'applicazione presente su iPhone.

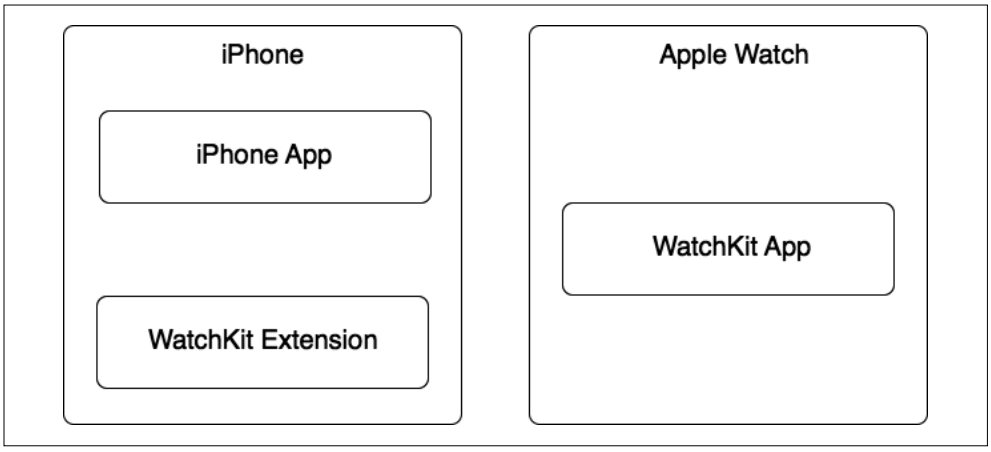

Figura 1.1 – Architettura di un'applicazione Apple Watch in watchOS 1.

In Figura 1.1 notiamo che un'applicazione Apple Watch è composta da tre elementi:

- **iPhone App:** è l'applicazione contenitore presente su smartphone; si occupa di effettuare le chiamate di rete e di fornire all'applicazione Watch i dati che questa richiede;
- **WatchKit Extension:** è l'estensione dell'iPhone App per l'orologio; in questo componente è presente tutta la logica applicativa;
- **WatchKit App:** contiene al suo interno tutto ciò che riguarda l'interfaccia grafica dell'applicazione: asset e storyboard (in watchOS non è possibile creare file di interfaccia di tipo XIB o gestire il tutto programmaticamente; è necessario lavorare con gli storyboard).

In watchOS 1, quindi, tutta la logica applicativa dell'applicazione risiede sull'iPhone; watchOS e iOS si occupano di gestire il dialogo tra la WatchKit Extension e la WatchKit App.
Ciò comporta tempi di risposta non ottimali (soprattutto se paragonati ad iOS) e latenza di comunicazione. Una volta che l'utente esegue un'azione, l'orologio la comunica al telefono, il quale esegue la logica applicativa e restituisce il risultato. Ricevuta la risposta, Apple Watch si occupa dell'aggiornamento dell'interfaccia grafica.

Architettura di watchOS 2

Al WWDC 2015 Apple ha presentato, a soli due mesi di distanza dall'uscita del suo device wearable, la seconda versione del sistema operativo dedicato a quest'ultimo. Il più grande cambiamento apportato riguarda proprio l'architettura.

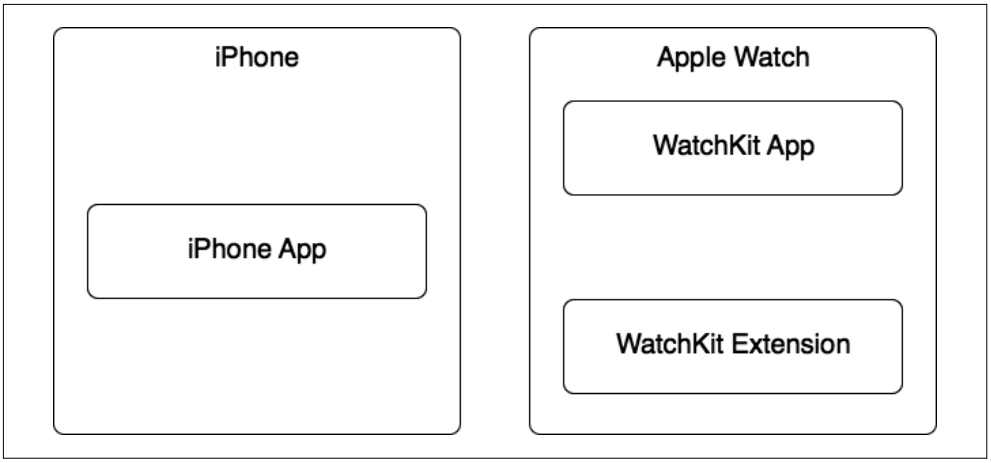

Figura 1.2 – Architettura di un'applicazione Apple Watch in watchOS 2.

La Figura 1.2 mostra come la componente *Extension* non sia più presente sull'iPhone, ma eseguita direttamente su Apple Watch; questo comporta notevoli benefici e aggiunte in termini di funzionalità:

- le applicazioni diventano indipendenti (in termini di esecuzione) dalla controparte iPhone;
- è possibile effettuare chiamate di rete direttamente dall'Apple Watch in Wi-Fi (il Watch, in mancanza di un iPhone, si collega alle reti Wi-Fi conosciute);
- eseguendo la logica direttamente sull'orologio, le prestazioni delle applicazioni vengono incrementate;
- con watchOS 2 viene data la possibilità agli sviluppatori di interfacciarsi direttamente con i componenti di Apple Watch: accelerometro, sensore per battito cardiaco e corona digitale.

Resta tuttavia il vincolo di gestire il tutto attraverso il modello di *Extension*: a oggi non è (ancora) possibile realizzare un'applicazione per Apple Watch senza la sua controparte iPhone.

Interagire con Apple Watch

Apple Watch è in grado di gestire diversi tipi di interazione:

- **Gesture:** come sui device iOS più grandi, le gesture sono il principale strumento di interazione con l'orologio. A livello utente sono supportati i tap, lo swipe laterale e lo scrolling verticale. Tali gesture non possono essere intercettate ma la loro gestione è completamente trasparente allo sviluppatore. A oggi non sono supportate gesture multitouch come il pinch to zoom; d'altronde, le ridotte dimensioni dello schermo non ne consentirebbero un uso corretto;
- **Force Touch:** è una nuova tipologia di interazione presentata da Apple, ovvero il sensore di pressione. Esso è in grado di capire quando l'utente effettua una pressione sullo schermo del device, distinguendo due livelli di forza applicata. Attualmente Apple non permette di intercettare in modo programmatico questa distinzione. Il tutto viene gestito automaticamente da watchOS 2 che, al riconoscimento della pressione da parte dell'utente, mostra un menu associato a una particolare vista. Parleremo più approfonditamente di questo aspetto nel prossimo capitolo;
- **Corona digitale:** la corona, da sempre presente sugli orologi, è stata mantenuta da Apple come punto di interazione tra l'utente e le applicazioni. Muovendo la corona digitale si può eseguire uno scrolling verticale della pagina o delle tabelle e cambiare selezione nei picker (vedere il paragrafo dedicato nel capitolo successivo). Come per il Force Touch, allo stato attuale non è possibile intercettare in modo programmatico il movimento della corona digitale. Essa è particolarmente utile in quanto permette all'utente di muoversi nella schermata senza coprire il piccolo schermo con le dita;
- **Dettatura vocale**: su Apple Watch non è presente una tastiera di sistema; le uniche modalità per inserire input testuali sono scegliere tra una lista (programmabile) di testi suggeriti oppure usare la voce per dettare ciò che si vuole scrivere;
- **Accelerometro**: con l'accelerometro è possibile intercettare il movimento del polso dell'utente e adeguare, di conseguenza, l'interfaccia grafica.

Risoluzioni e dimensioni

Il primo modello di Apple Watch è presente sul mercato in due dimensioni e due risoluzioni diverse:

- il modello da 38 mm ha una risoluzione dello schermo pari a 272 pixel in larghezza e 340 pixel in altezza;
- il modello da 42 mm ha una risoluzione dello schermo pari a 312 pixel in larghezza e 390 pixel in altezza.

Entrambi i device hanno una densità di risoluzione pari a 326 pixel per pollice e quindi lo schermo può fregiarsi del titolo di Retina Display.
I due modelli condividono l'aspect ratio dello schermo: entrambi hanno una ratio di 4:5. Il fatto di supportare tutte e due gli schermi, quindi, non richiede un eccessivo lavoro di sviluppo o di design.

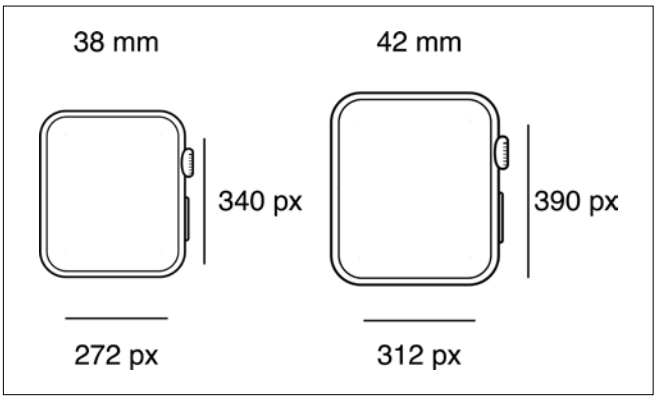

Figura 1.3 – Dimensioni e risoluzioni dello schermo.

Storyboard e layout

A differenza di un'applicazione iPhone o iPad, in cui è possibile scegliere tra gli Storyboard e gli XIB, per un'applicazione Apple Watch si è obbligati a usare gli Storyboard per il disegno dell'interfaccia grafica; inoltre, non esiste il concetto di Auto Layout. Interface Builder si occupa di posizionare correttamente gli oggetti all'interno della vista a mano a mano che questi vengono aggiunti.
Auto Layout è guidato da un sistema di vincoli e relazioni tra la dimensione e la posizione delle view. WatchKit introduce un sistema del tutto nuovo: anziché creare il layout partendo dai vincoli, posiziona gli elementi dell'interfaccia in funzione della dimensione dei contenuti e della spaziatura.

Componenti grafici

In questo paragrafo offriamo una rapida carrellata degli oggetti di interfaccia presenti in watchOS 2. Tutti questi componenti sono sottoclassi della classe `WKInterfaceObject`; non sono view ma degli oggetti proxy verso le interfacce presentate dalla Watch App. La comunicazione fra l'oggetto proxy e la corrispondente interfaccia grafica avviene in una sola direzione, dalla Watch Extension verso l'Apple Watch. I `WKInterfaceObject` sono quindi degli oggetti write-only: è possibile modificarne lo stato ma non recuperarlo.

NOTA

In una `WKInterfaceLabel` è possibile, per esempio, impostare il testo visualizzato a schermo attraverso il selettore `setText` ma non recuperarlo con un metodo, in quanto non è presente un selettore di lettura. Questa gestione è stata introdotta per motivi di performance e latenza. Spetta allo sviluppatore mantenere lo stato di quanto impostato su ogni oggetto visualizzato.

I componenti grafici di watchOS 2 sono:

- **`WKInterfaceLabel`**: corrispettivo Watch delle `UILabel` di iOS;
- **`WKInterfaceButton`**: corrispettivo di `UIButton` di iOS;
- **`WKInterfaceTable`**: tabelle a una sola dimensione (a differenza di iOS); non esiste il concetto di sezione ma solo di riga;
- **`WKInterfaceMap`**: versione limitata rispetto alle mappe di iOS. In WatchKit è possibile inserire *pin* e *annotation* ma le mappe non sono interattive. L'unica operazione possibile è che al tap su una mappa si venga portati direttamente sull'applicazione Mappe di Apple Watch;
- **`WKInterfaceImage`**: corrispettivo di `UIImageView` di iOS. È possibile, inoltre, impostare una serie di immagini e animarle;
- **`WKInterfaceDate`**: componente testuale specializzato nella visualizzazione di date, una classe unica di WatchKit;
- **`WKInterfaceGroup`**: classe esclusiva di WatchKit, si occupa di gestire i raggruppamenti e il layout degli altri oggetti di interfaccia;
- **`WKInterfaceSeparator`**: in iOS quando si vuole inserire un separatore si può lavorare con le immagini o far ricorso a una `UIView` configurata ad hoc; in WatchKit c'è un oggetto dedicato ai separatori sia orizzontali sia verticali;
- **`WKInterfaceSlider`**: simile a `UISlider` di iOS ma meno customizzabile;
- **`WKInterfaceSwitch`**: come `UISwitch`, con la differenza che è possibile impostare anche il testo della label dedicata;
- **`WKInterfaceTimer:`** come per `WKInterfaceDate`, anche l'oggetto `WKInterfaceTimer` è dedicato alla visualizzazione del tempo. Si tratta sostanzialmente di un componente testuale che si occupa di visualizzare un timer verso una specifica data (sia in futuro sia nel passato);
- **`WKInterfacePicker`**: aggiunto da Apple in watchOS 2, è il corrispettivo di `UIPickerView`; come per iOS può essere un picker che mostra una serie di informazioni con uno scrolling verticale, oppure (solo in watchOS) una sequenza di immagini. Il controllo dei picker avviene tramite la corona digitale;
- **`WKInterfaceMovie`**: altro oggetto inserito in watchOS 2 è la vista, che si occupa di riprodurre un video o un audio su Apple Watch; il contenuto audio/video può essere presente sull'orologio oppure scaricato dalla Rete.