TESINA PER IL CORSO DI
PROGRAMMAZIONE DI SISTEMI MOBILE

Corso di laurea in **Ingegneria e scienze informatiche**
Mattia Capucci 655780

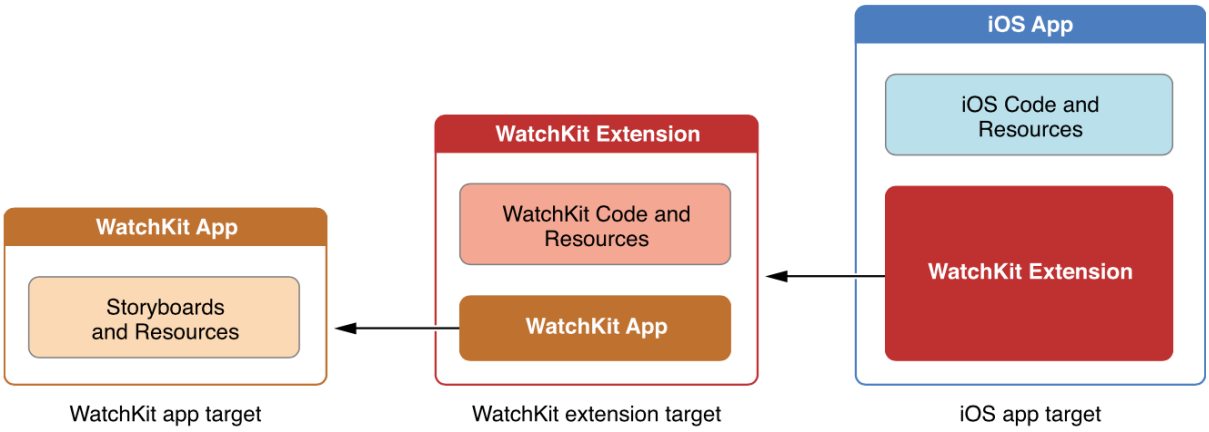
PRINCIPI DI PROGRAMMAZIONE SU APPLE WATCH

1. Introduzione

Attualmente, con una versione di watchOS inferiore alla 2.0, le applicazioni di terze parti richiedono la presenza di un iPhone per essere eseguite. L'applicazione, quindi, si comporrà di due parti principali:

- La ***WatchKit App:*** che gira su Apple Watch e contiene solo la storyboard ed eventuali risorse associate all'interfaccia utente
- La ***WatchKit Extension***: che gira su iPhone e contiene il codice per modificare l'interfaccia utente dell'app e per rispondere alle interazioni dell'utente.

La struttura dell'app è mostrata in questo schema:

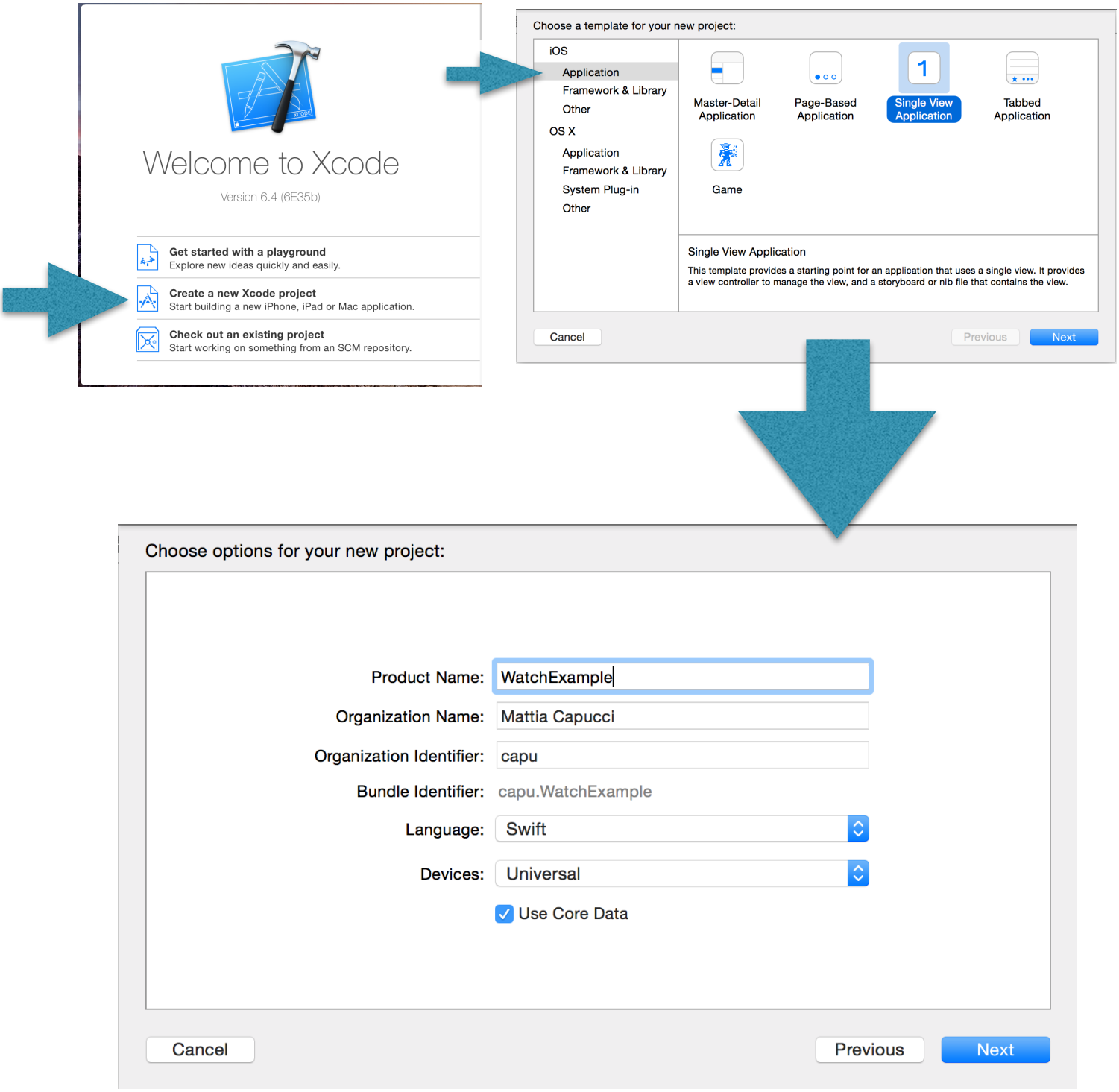


La WatchKit App è inclusa all'interno della WatchKit Extension, la quale a sua volta è inclusa all'interno dell'applicazione iOS.

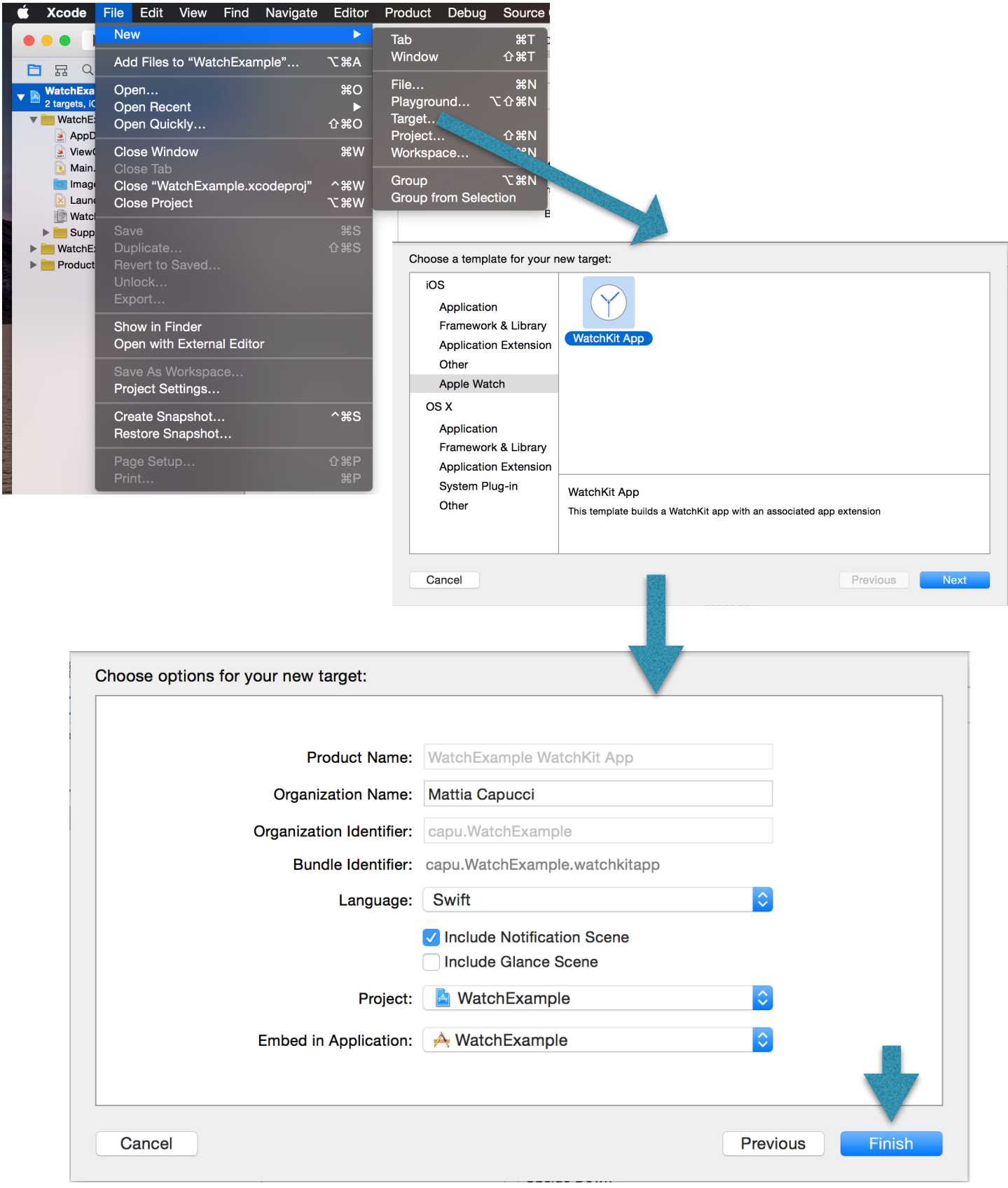
2. Creare un progetto Apple Watch

Per sviluppare una app per Apple Watch occorre seguire i seguenti passaggi:

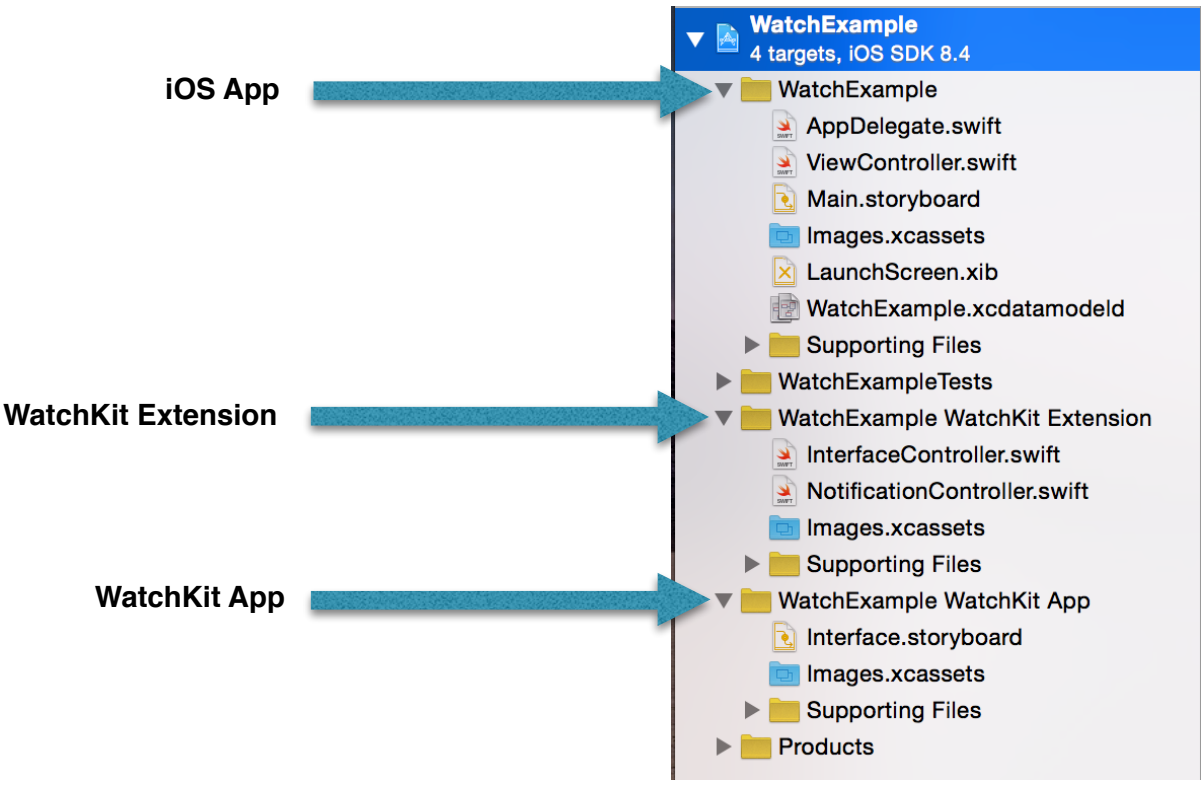
- Aprire Xcode e creare un nuovo progetto per iOS



- Aggiungere il target Apple Watch all'applicazione

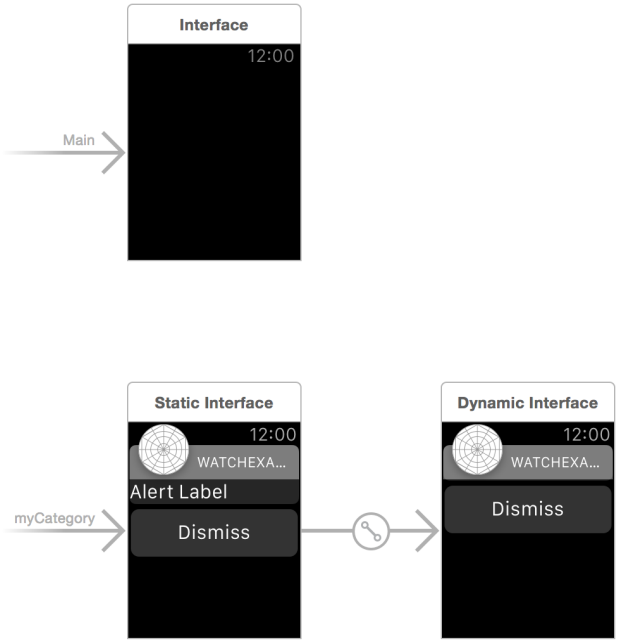


Una volta completati questi passaggi, saranno visibili fra le cartelle di progetto gli elementi che caratterizzano la struttura dell'app vista sopra.

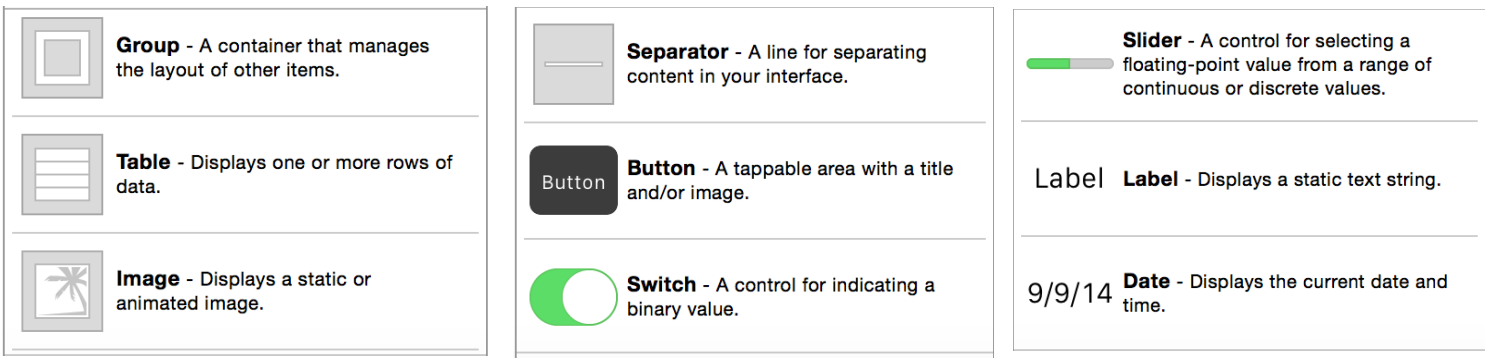


3. La storyboard in Apple Watch

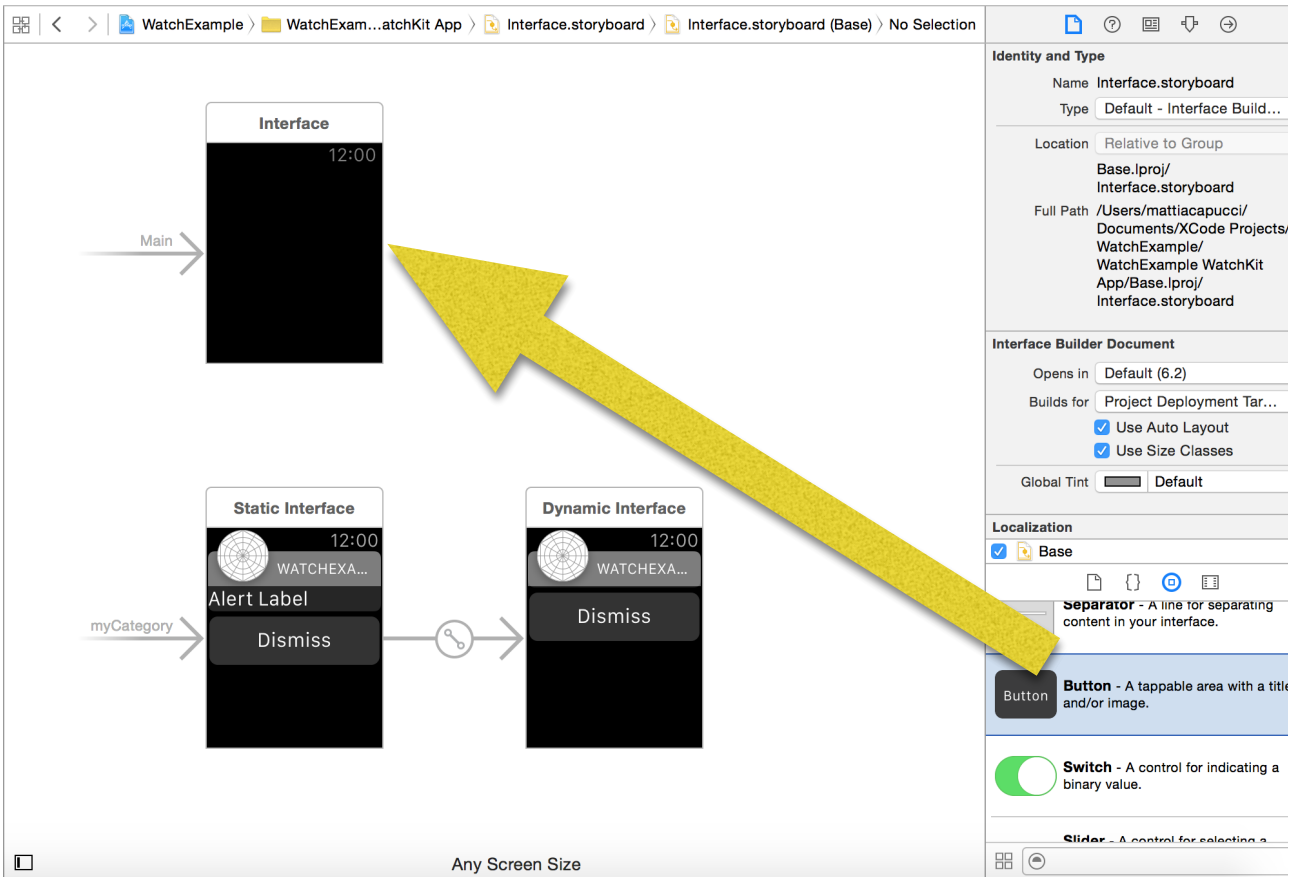
La storyboard in Apple Watch si compone di due parti: la parte superiore è l'interfaccia vera e propria dell'applicazione mentre la parte inferiore permette di costruire interfacce grafiche per le notifiche.

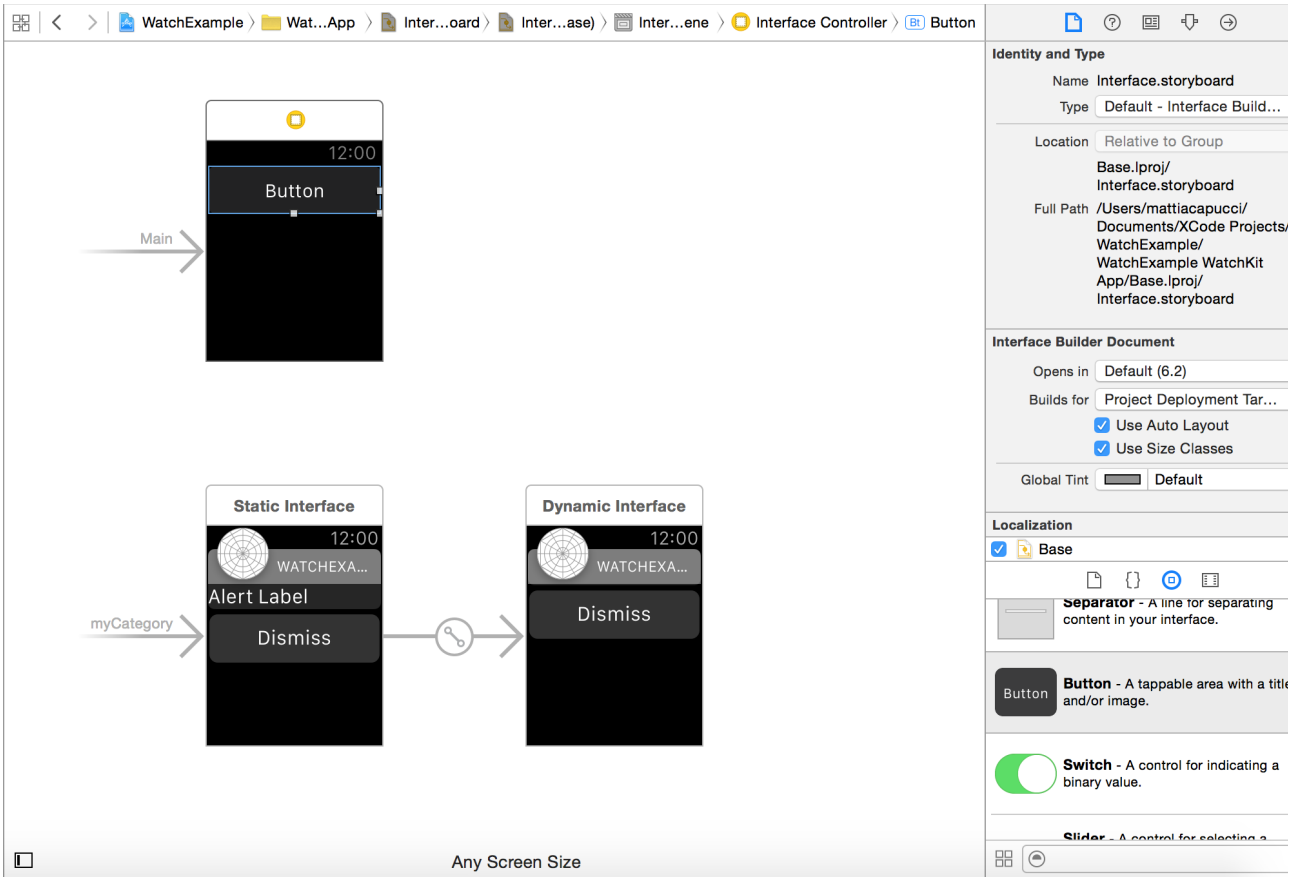


Così come per iOS, in basso a destra abbiamo una serie di oggetti grafici che possiamo aggiungere alla nostra interfaccia utente. Molti di essi sono comuni a iOS, altri sono totalmente nuovi.

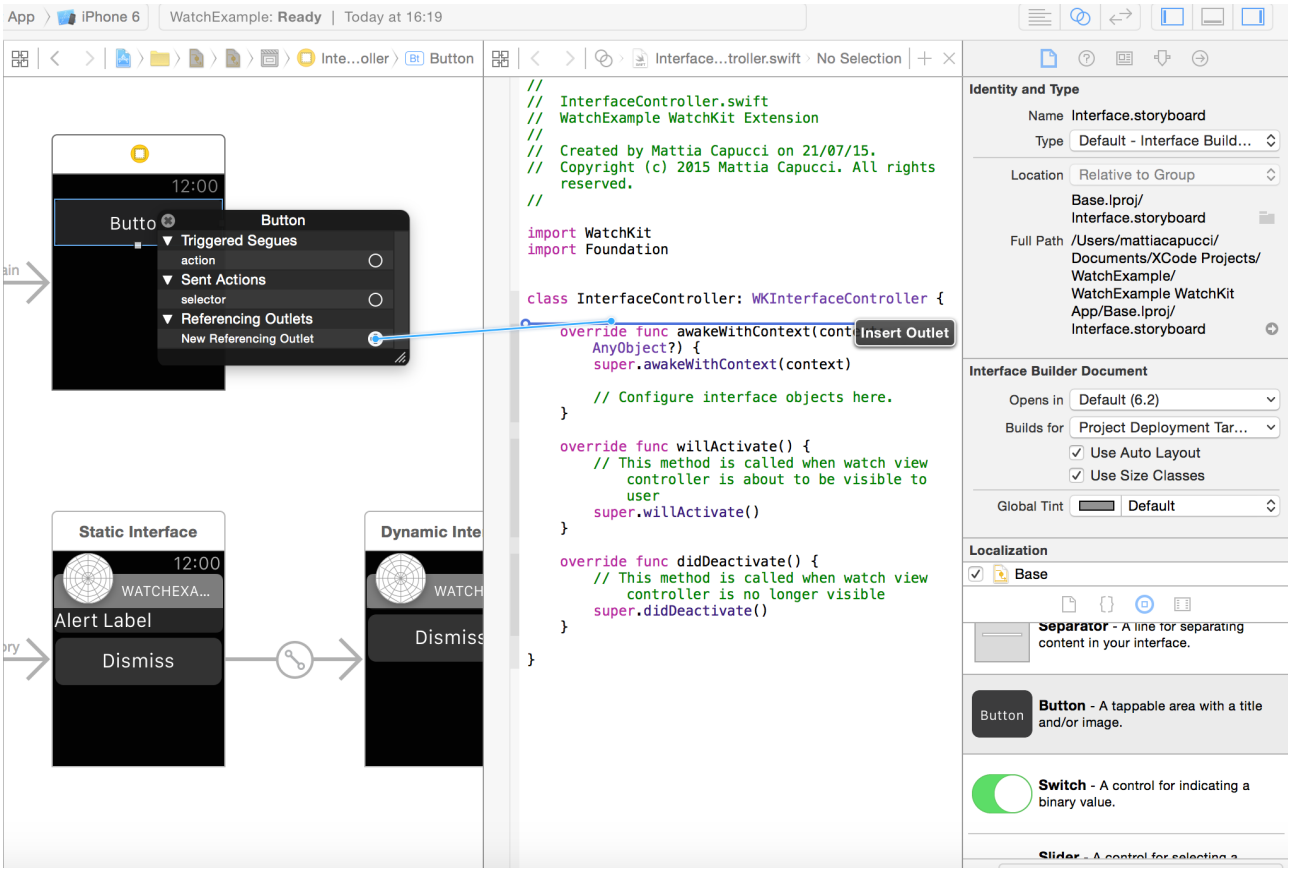


Il **Group**, per esempio, permette di creare gruppi di elementi mentre **Table** (equivalente di Table View in iOS) permette di visualizzare una o più righe di dati. Per posizionare un elemento nella View basterà trascinarlo:

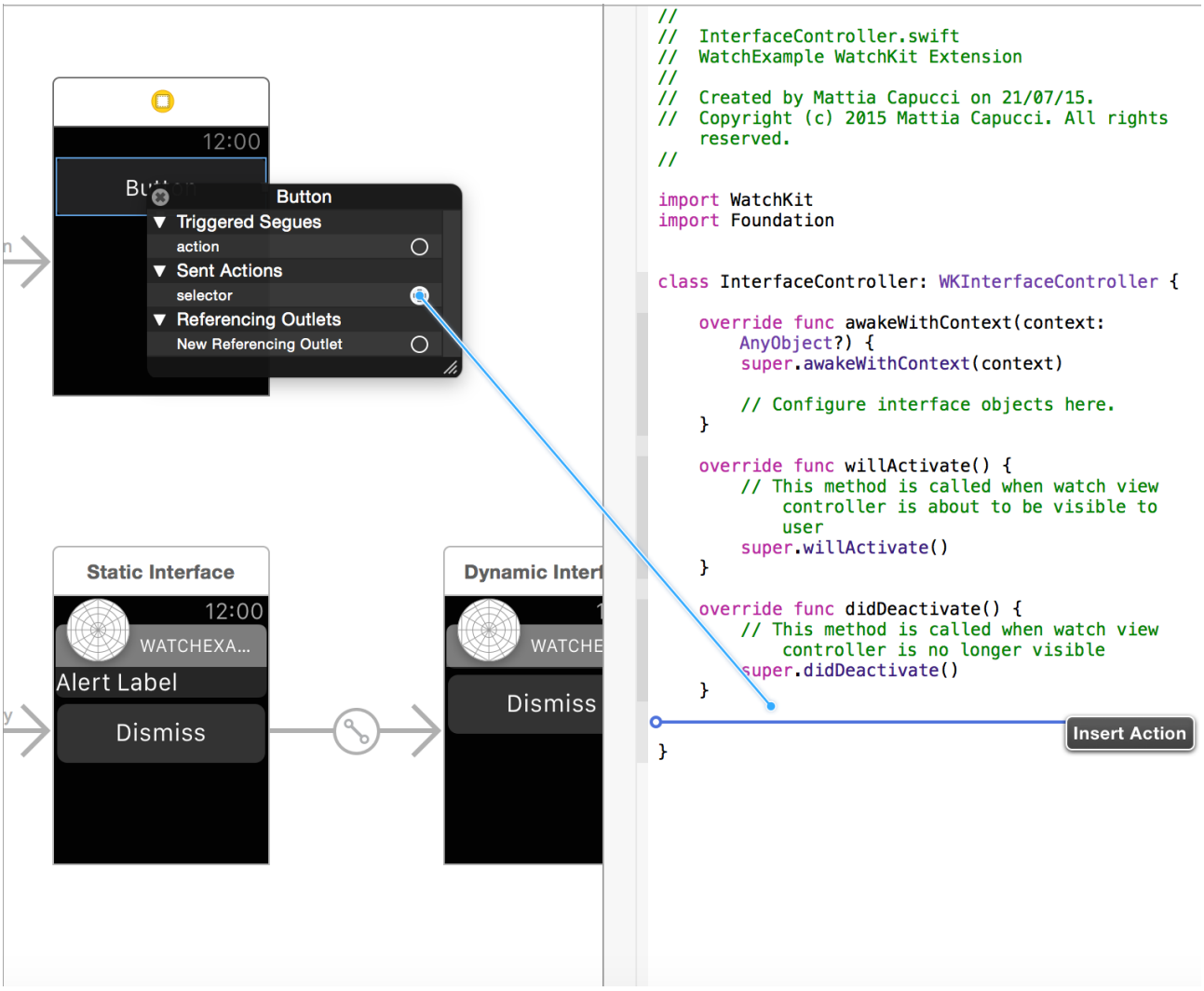




Similmente a come si procede per la programmazione in iOS, per creare un collegamento fra l'elemento grafico e l'Interface Controller si crea un nuovo **outlet**. Per farlo posizionamici dapprima nel tab *Assistant Editor*, successivamente facciamo un click destro sull'elemento grafico e infine trasciniamo l'elemento *New Referencing Outlet* nella classe di interesse.

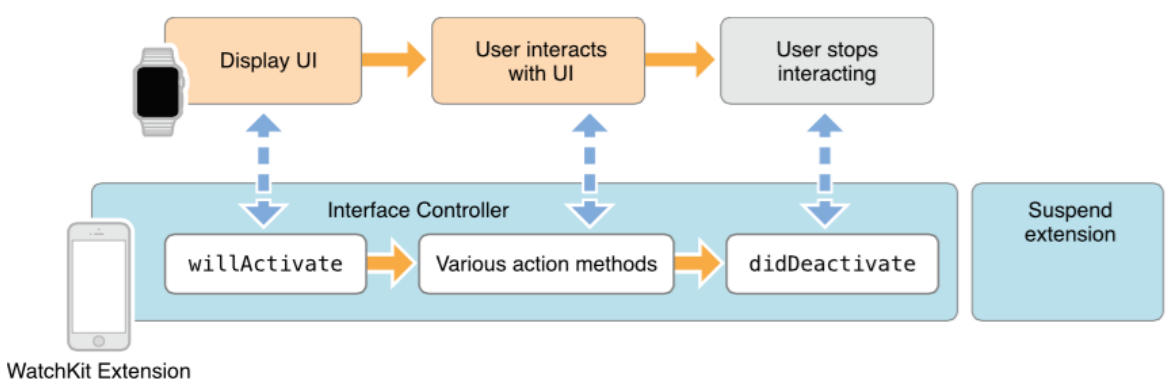


Prendendo in esempio il **Button**, possiamo aggiungere un'azione, alla pressione di questo, trascinando *selector* nella classe di interesse.



4. Interface Controller

Ogni scena rappresentata in Apple Watch è gestita da una classe che è un'istanza di **WKInterfaceController**, l'equivalente del View Controller in iOS. Il suo ciclo di vita è possibile schematizzarlo come segue:



I metodi di rilievo sono i seguenti:

awakeWithContext: questo metodo permette di configurare l'Interface Controller, preso in ingresso un qualsiasi context data. Occorre usare questo metodo per caricare dati e/o aggiornare gli oggetti grafici nella scena.

willActivate: questo metodo specifica che l'interfaccia sarà presto visibile all'utente. Viene utilizzato solo per piccoli cambiamenti nell'interfaccia.

didDeactivate: questo metodo viene utilizzato per ripulire l'interfaccia e porla in uno stato di inattività. Per esempio può essere utilizzato per fermare delle animazioni. Non è possibile impostare alcun valore agli oggetti di interfaccia in questo metodo.

5. Le immagini

Per inserire delle immagini nelle nostre scene possiamo seguire due strade:

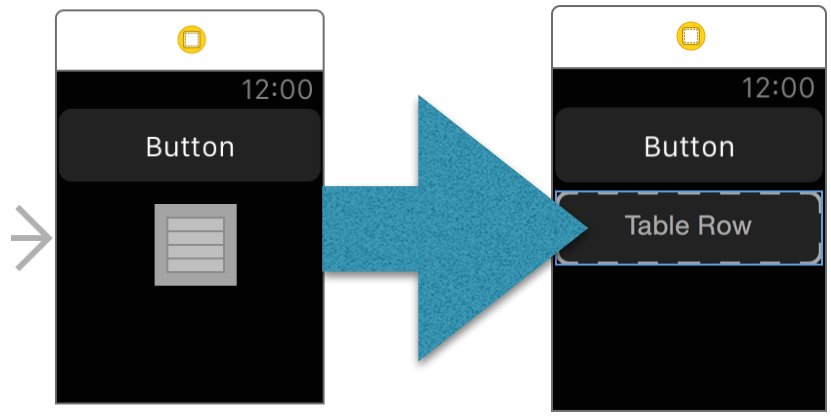
- Mostrare le immagini come elementi a se stanti, utilizzando la classe **WKInterfaceImage**
- Impostare l'immagine come background di un bottone, di un group…

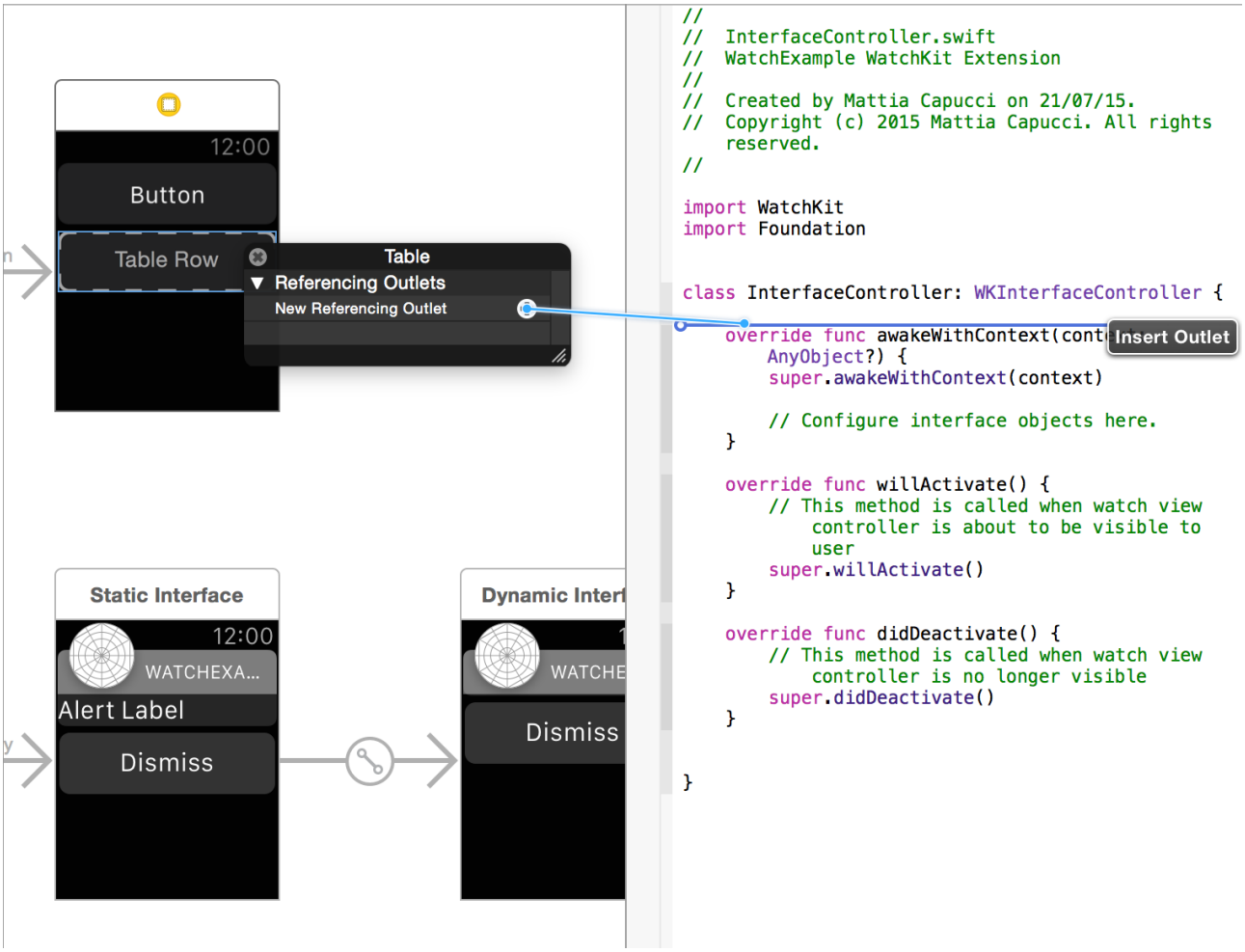
Allo stesso tempo ci sono diversi metodi per impostare un'immagine:

- **setImageNamed** e **setBackgroundImageNamed** impostano un'immagine che è già presente nella WatchKit App o nella cache.
- **setImage**, **setImageData**, **setBackgroundImage**, **setBackgroundImageData** impostano un'immagine dopo averla trasferita via wireless dalla WatchKit Extension alla WatchKit App. A causa del trasferimento, l'invio di molte immagini di grande dimensione possono causare un rallentamento nella visualizzazione della scena. E' quindi buona norma utilizzare immagini della dimensione corretta oppure comprimerle prima che siano inviate alla WatchKit App.

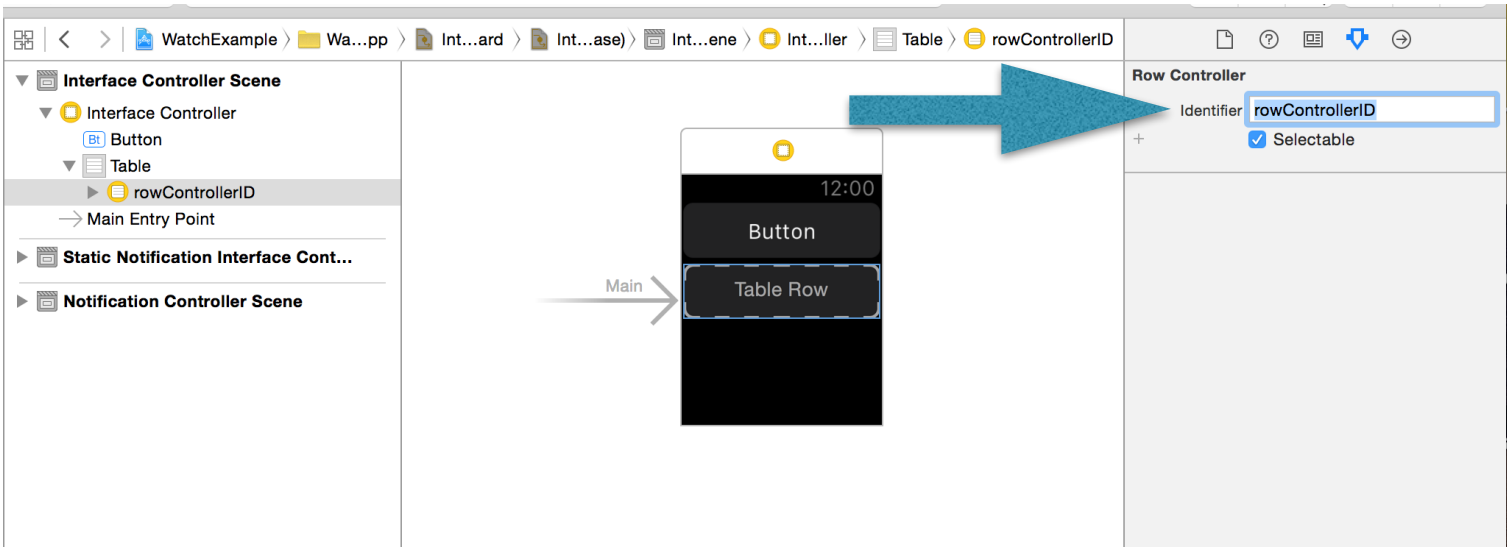
6. Le Table

Come detto in precedenza, le table permettono di visualizzare un insieme di righe di informazioni. Per poter utilizzare una table, per prima cosa aggiungiamone una nella nostra interfaccia e creiamo un outlet nella classe di riferimento.

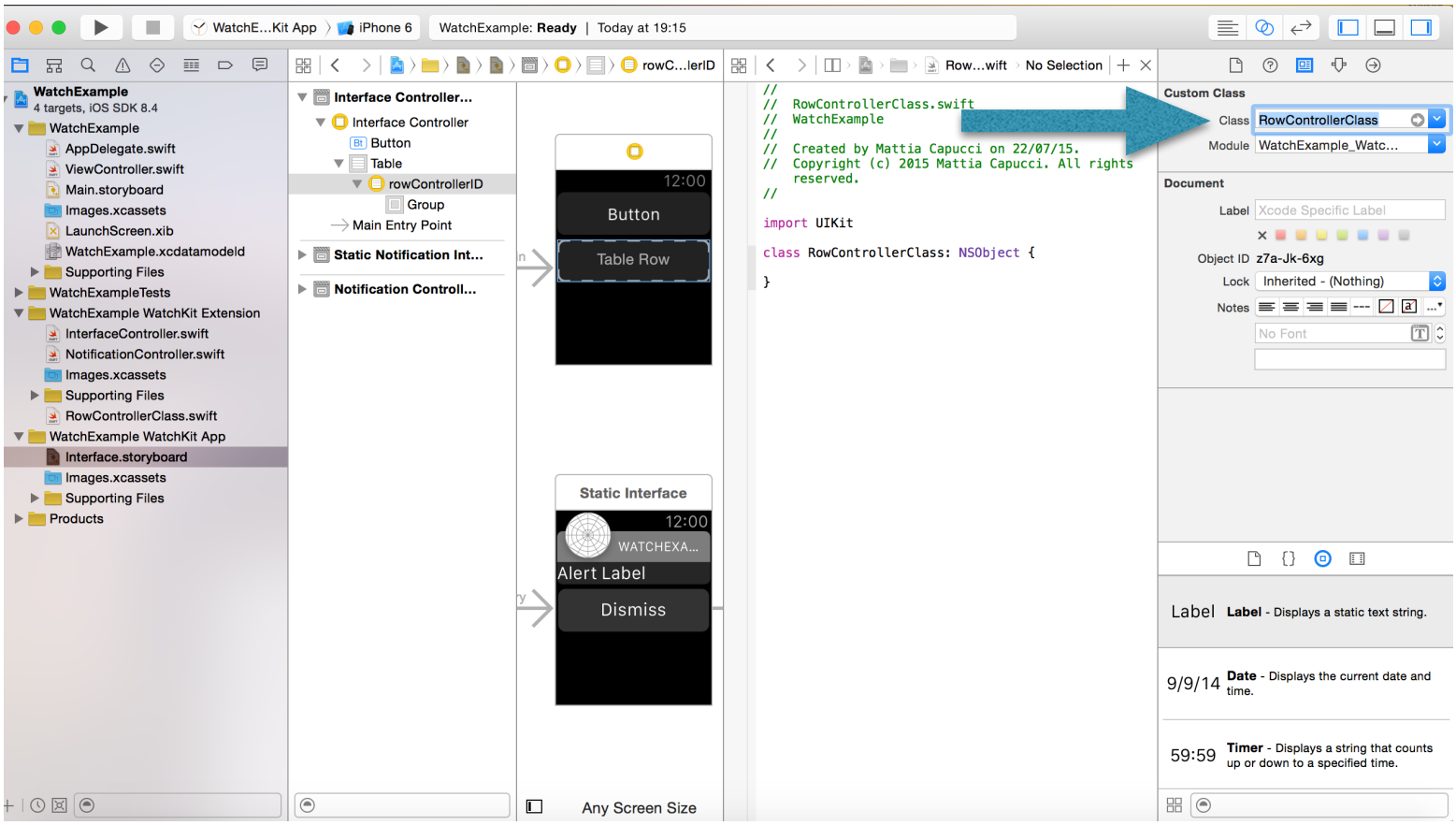




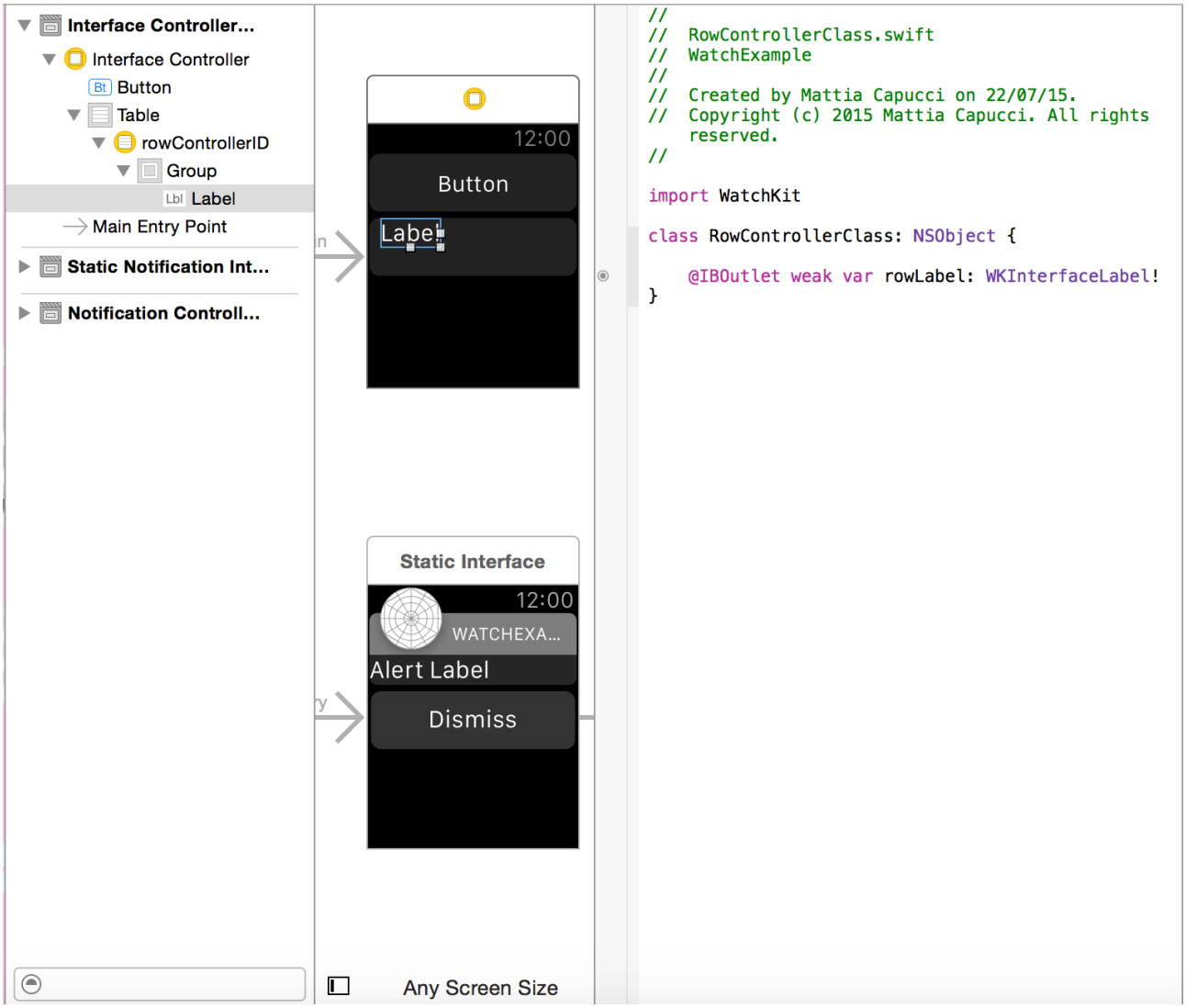
Successivamente dobbiamo creare un **Row Controller** che ci permetta di definire la singola riga della tabella. Xcode in automatico ne crea uno di default: selezioniamolo e assegniamogli un ID.



In seguito dobbiamo creare una classe che definisca il nostro Row Controller. Questa classe conterrà tutti gli elementi che si vuole avere in una riga della tabella, per esempio label, immagini … Per ogni elemento che si vuole mostrare, si creerà un outlet su questa classe, che deve essere una sottoclasse di **NSObject**. Una volta creata la classe, la prima cosa da fare è assegnarla al Row Controller. Per farlo, spostarsi su *Identity Inspector* e sotto *Class* impostare la classe appena creata.



Supponiamo che in ogni riga della tabella vogliamo visualizzare una label. La classe del Row Controller si presenterà così:

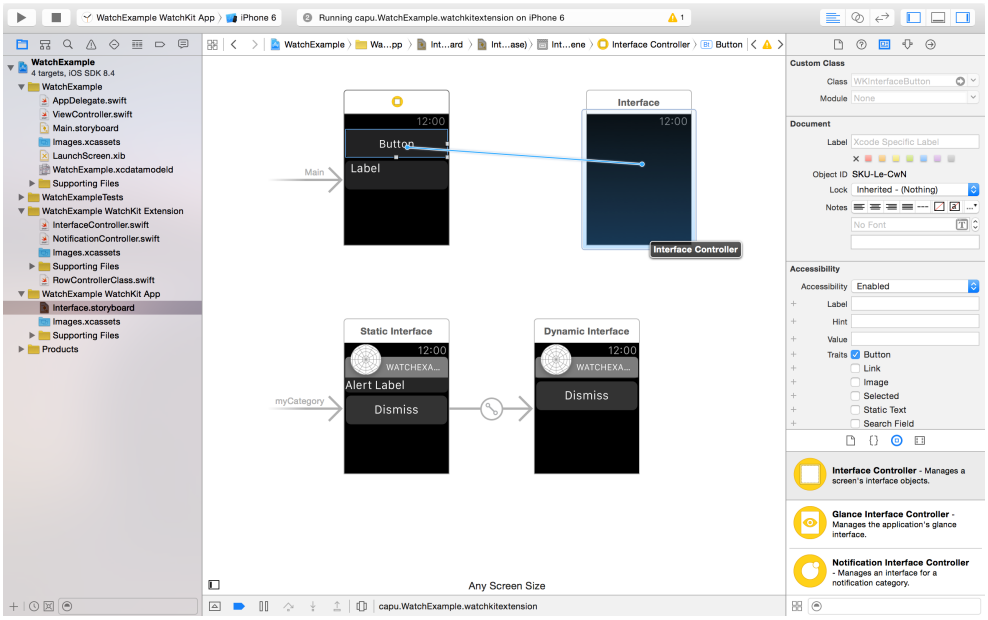


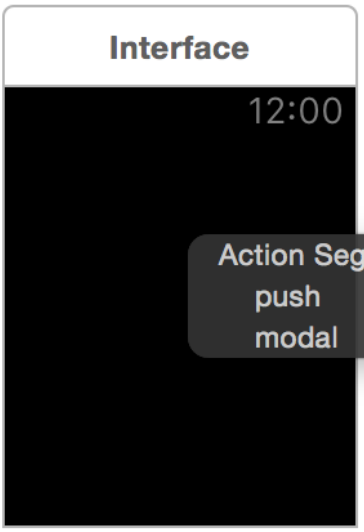
Ora, tramite codice, possiamo gestire la nostra tabella. I metodi di rilievo sono i seguenti:

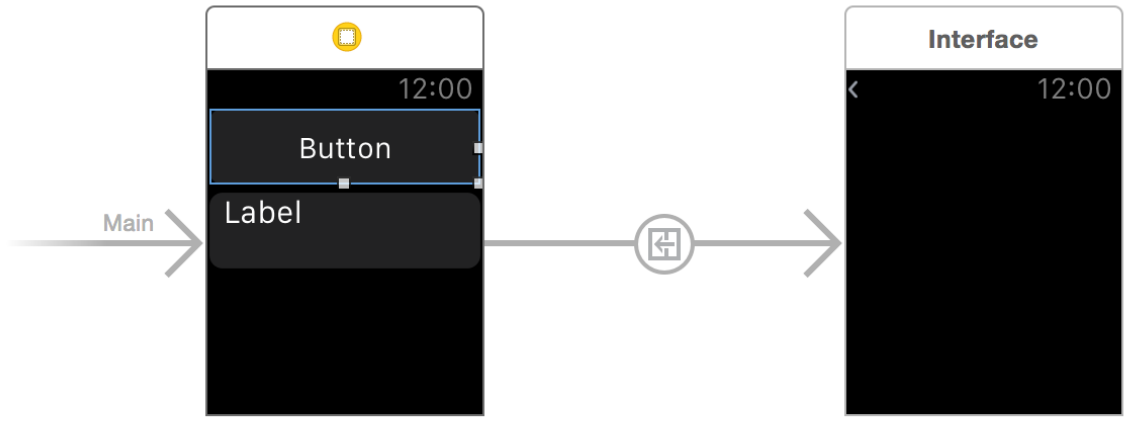
- **setNumberOfRows**(numberOfRows: Int, withRowType: String) permette di impostare il numero di righe di una tabella. Inoltre richiede in ingresso il tipo di riga, ovvero l'identificativo del Row Controller precedentemente impostato
- **insertRowsAtIndexes**(rows: NSIndexSet, withRowType: String) permette di inserire una serie di righe, prendendo in ingresso il set di indici delle righe e il tipo di riga, ovvero l'identificativo del Row Controller.
- **rowControllerAtIndex**(index: Int) permette, preso in ingresso l'indice di riga, di aver accesso al Row Controller corrispondente e a tutti gli oggetti che esso contiene.

7. I Segue

Così come in iOS, anche in Apple Watch è possibile passare da una scena ad un'altra tramite **segue**. Per creare un segue tra un interface controller e un altro basterà tenere premuto il tasto *CTRL* e scegliere la tipologia di segue. Per esempio, supponendo che il passaggio da una interfaccia all'altra avvenga tramite la pressione di un bottone, avremo:



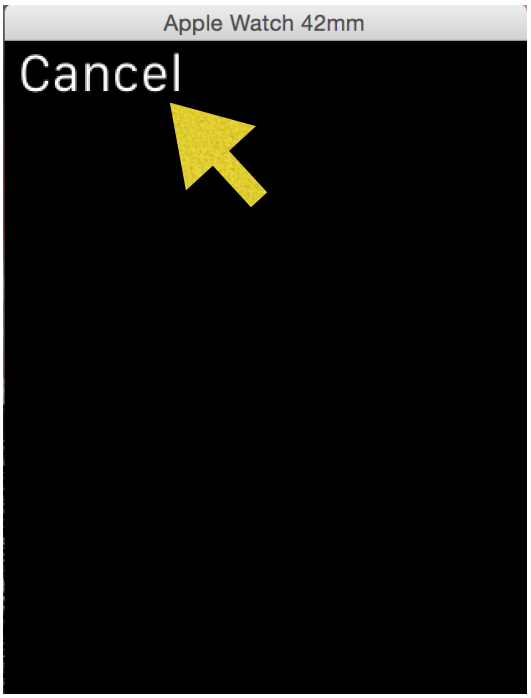




I segue si differenziano in:

- **Push**: il passaggio alla successiva scena avviene con un'animazione da destra verso sinistra e possiamo tornare alla scena precedente tramite la freccia che viene creata in automatico in alto a sinistra.

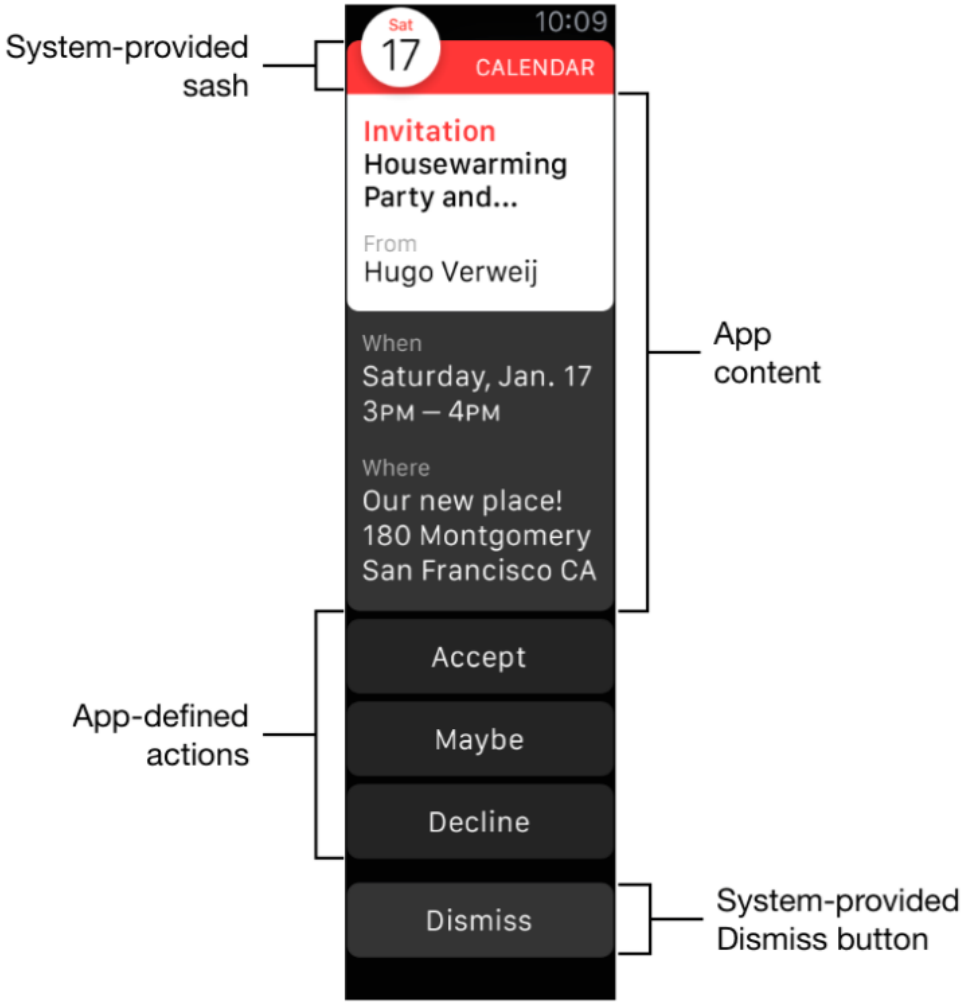
- **Modal:** Il passaggio alla successiva scena avviene con un'animazione dal basso verso l'alto e possiamo tornare alla scena precedente tramtite il bottone *Cancel* posto in alto a sinistra.



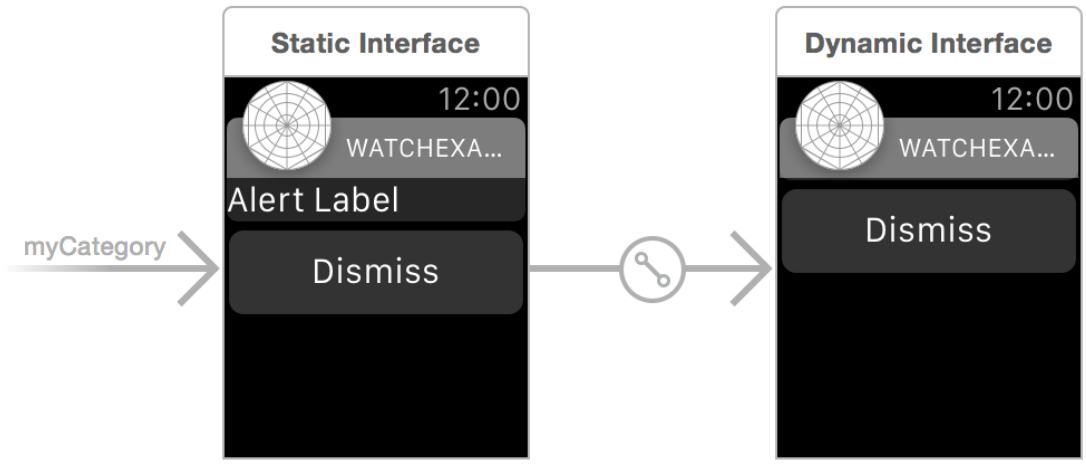
8. Le notifiche

Possiamo suddividere le notifiche in Apple Watch in due categorie: notifiche **short-look** e **long-look**. All'inizio l'utente vede la prima categoria di notifiche. Essa si presenta con un template predefinito dal sistema:

Successivamente, se l'utente continua a visualizzare la notifica, il sistema transita alla seconda categoria di notifiche definita come segue:



Possiamo modificare una notifica di tipo long-look, visibile nella storyboard:



Essa si suddivide in interfaccia **statica** e **dinamica**. L'interfaccia statica è un semplice modo per visualizzare la notifica e contiene tutti quegli elementi che possono essere configurati a tempo di design. L'interfaccia dinamica, invece, è facoltativa e permette di modificare la visualizzazione del contenuto delle notifiche. WatchKit, se non presente l'interfaccia dinamica, mostra quella statica. Se presente invece, mostra quella dinamica.