Notification Center

Widget e Interactive notification

“the Notification Center framework provides ways to customize the editing and searching experience in a widget”

App Extension

- Permette di estendere la funzionalità di un app ed il suo contenuto, in modo che l'utente abbia la possibilità di accedere a quei contenuti mentre sta utilizzando altre app.

- ogni area del sistema che supporta una determinata app extension, è chiamata **extension point**. Es: l'EP relativa al notification center è chiamata **Today**

- una today extension è chiamata **Widget**

- altri tipi di EP: Share, Action, Photo editing, custom keyboard…

Attenzione

- Un app extension è **diversa** da un app. nonostante abbiamo bisogno dell'utilizzo di un app che contenga un extension, quest' ultima è un **binario separato che "vive" in maniera indipendente rispetto all'app che lo contiene**

- Un app che contiene un extension è chiamata **app containing**

Creare un app Extension

- Per la creazione abbiamo bisogno di aggiungere un **nuovo target** nell'app containing.

- Un extension target specifica i settaggi ed i file utili alla compilazione nell'intero progetto

- Utilizzare i template messi a disposizione da Xcode. Ognuno contiene:

    - file d'implementazione dell' AE

    - settings

- Verrà infine prodotto un binario separato che viene aggiungo al bundle della containing app

Distribuzione di un app extension

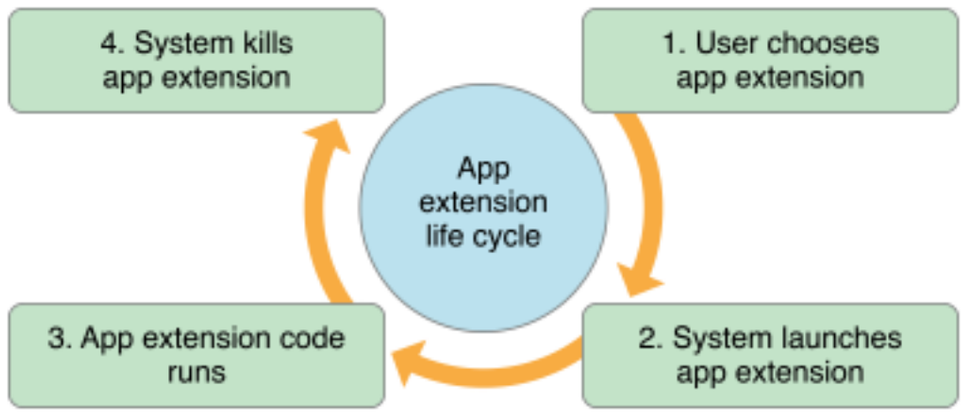
- Non è possibile distribuire un app extension sull'App store senza che quest'ultima sia dentro una containing app.

- L'app store rifiuta ogni app extension che utilizza uno o più framework che utilizzano API non disponibili per le AE.

Ciclo di vita di una AE

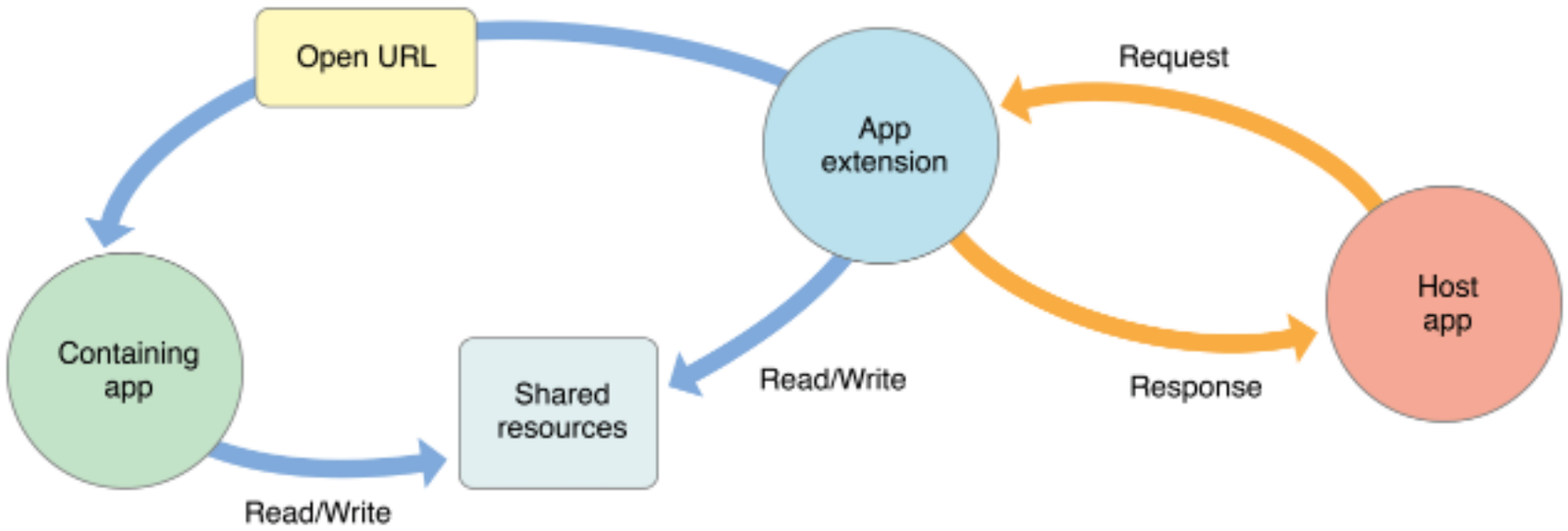
- Spesso lanciate quando un utente sceglie l'AE o dalla UI o da un activity controller

- un app che l'utente utilizza per scegliere l'AE è chiamata **host app**



Comunicazione

- non c’è diretta comunicazione tra l’AE e la Containing App



- lo sviluppatore non deve preoccupare della logica di tale meccanismo in quanto utilizzerà le API di alto livello fornite dalle extension app ed il sistema.

- tipicamente la containing app non è in esecuzione quando lo è l’extension app

API non disponibili

- Alcune api non sono disponibili nell'utilizzo delle app extension:
  - Accesso all'oggetto sharedApplication
  - HealthKit e EventKit UI framework
  - accesso alla camera o microfono su iOS device

Today widget

Widgets give users quick access to information that’s important right now

Today widget

- On both platforms, a Today widget should:
    - Ensure that content always looks up to date
    - Respond appropriately to user interactions
    - Perform well (in particular, iOS widgets must use memory wisely or the system may terminate them)

Design dei widget

- nei widget l'interazione con l'utente è veloce e limitata, di conseguenza il design deve essere semplice ed in modo che vengano valorizzate le notizie che interessano all'utente.

- In generale è buona idea limitare il numero di elementi con il quale interagire.

- i widgets **non supportano** la tastiera

plist di un widget

```
<key>NSExtension</key>

<dict>

  <key>NSExtensionPointIdentifier</key>

  <string>com.apple.widget-extension</string>

   <key>NSExtensionPrincipalClass</key>

    <string>TodayViewController</string>

</dict>
```

- il valore *TodayViewController* è ovviamente la classe controller (conforme al protocollo NCWidgetProviding) relativa al widget

NCWidgetController

- Import NotificationCenter
- *definisce un oggetto widget e fornisce alcuni metodi per specificare se il widget ha contenuto da visualizzare o meno*

Declaration

SWIFT

```
      func setHasContent(_ flag: Bool,
forWidgetWithBundleIdentifier bundleID: String!)
```

OBJECTIVE-C

```
- (void)setHasContent:(BOOL)flag
forWidgetWithBundleIdentifier:(NSString *)bundleID
```

- *così da fare in modo che la containing app possa determinare la visualizzazione o meno del widget a seconda che esista o no del contenuto da visualizzare*

NCWidgetProviding

- tale protocollo *permette di customizzare la grafica ed il comportamento di un widget*
- *possibilità di modificare i margini della view del widget*
- *possibilità di aggiornamento del contenuto e di ridisegnare la view. Quando i contenuti sono aggiornati, viene richiamato il blocco*

```
SWIFT
optional func widgetPerformUpdateWithCompletionHandler(_ completionHandler:
((NCUpdateResult) -> Void)!)

OBJECTIVE-C
- (void)widgetPerformUpdateWithCompletionHandler:(void (^)(NCUpdateResult
result))completionHandler
```

- diverse costanti(enum) specificano il risultato dell'aggiornamento del widget
  - NCUpdateResultNewData
  - NCUpdateResultNoData
  - NCUpdateResultFailed

Comunicazione tra containing app e widget

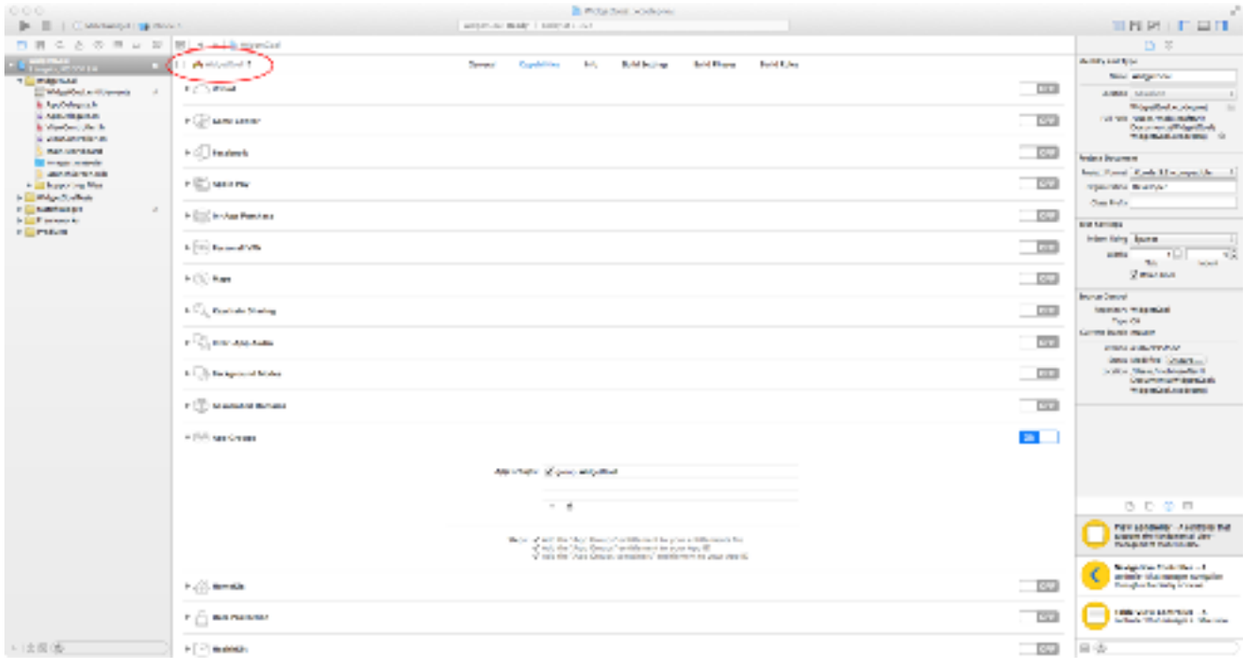
- Shared data

- possibilità di aprire la containing app dal widget

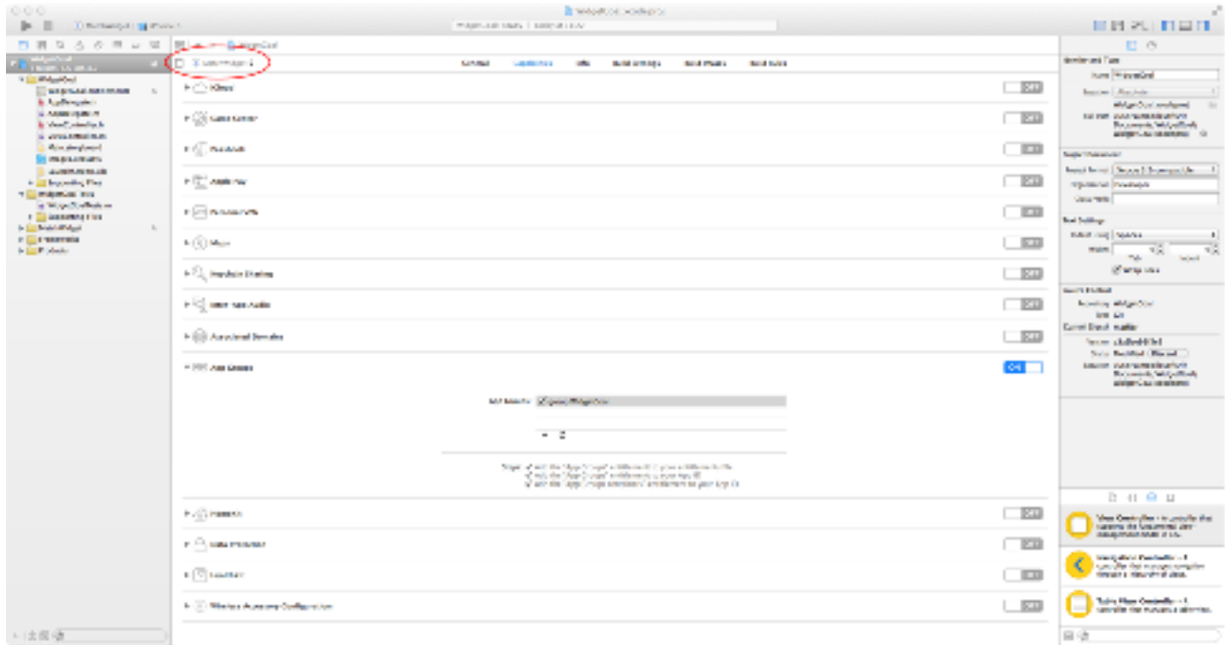
Shared Data

- nonostante l' app extension sia contenuta nell'app bundle, il suo security domain è distinto dalla containing app.

- Per abilitare il data sharing abbiamo bisogno di aggiungere le due app in uno stesso **App Group.**



Containing app

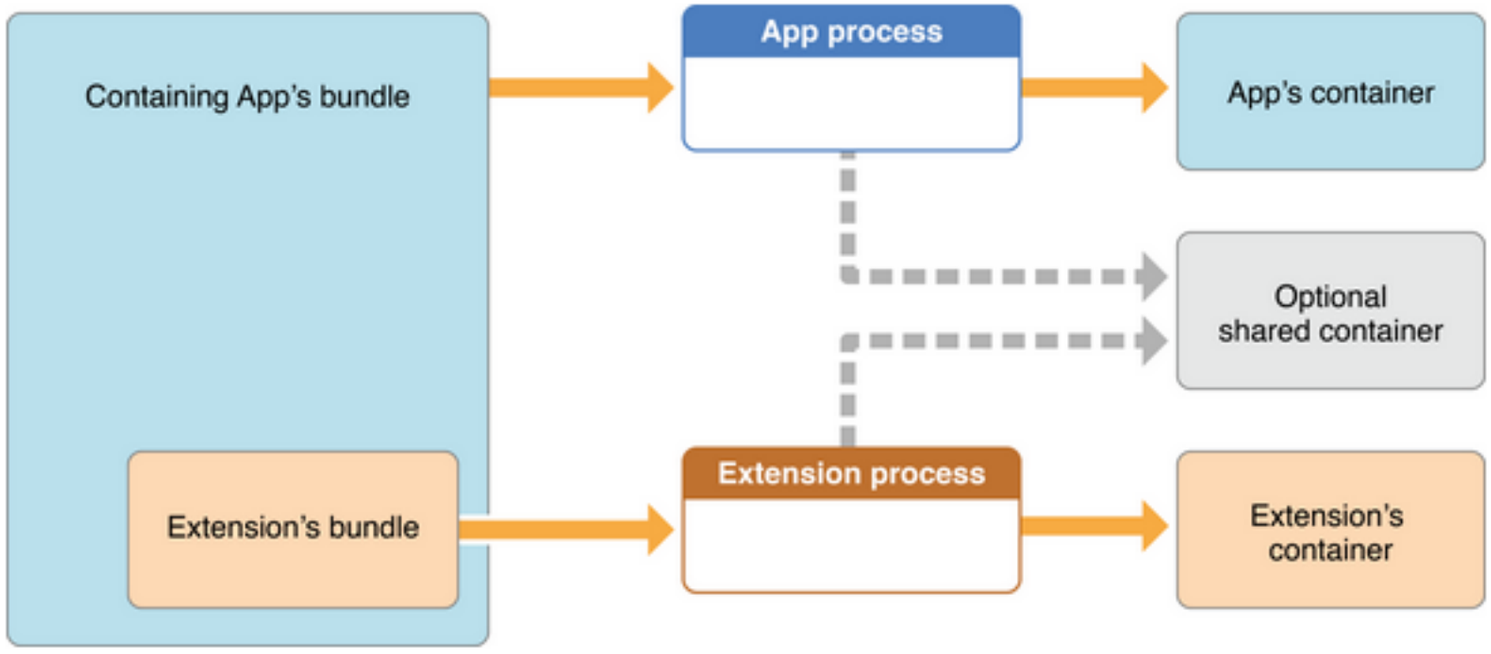


extension app

- quando si cercherà di attivare l'App Groups, Xcode ci chiedere di autenticarci con l'account developer

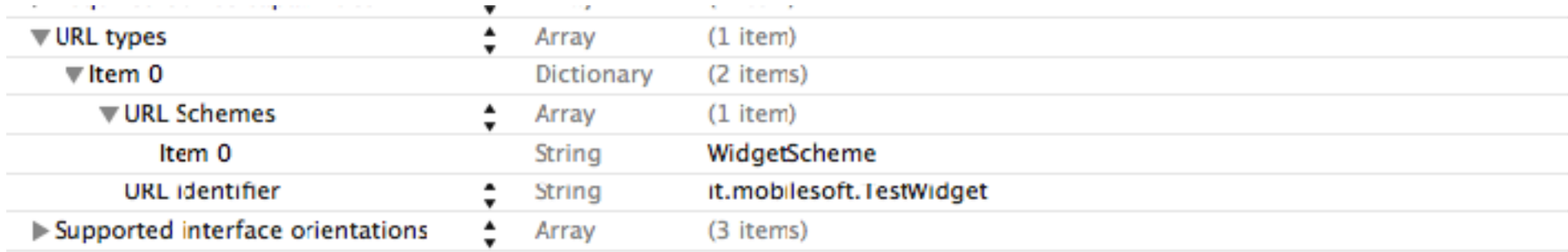
Shared Data

- A questo punto è possibile comunicare utilizzando la classe **NSUserDefaults** per la condivisione dei dati.
- Per rendere attivo lo sharing richiamo il metodo: NSUserDefaults *sharedDefaults = [[NSUserDefaults alloc] initWithSuiteName:@“group.appID”];
- [sharedDefaults setObject:[self.textField.text] forKey:@“keySettings"];
- [sharedDefaults synchronize];



Apertura della containing app dal widget

- setto i parametri (scheme url) della containing app



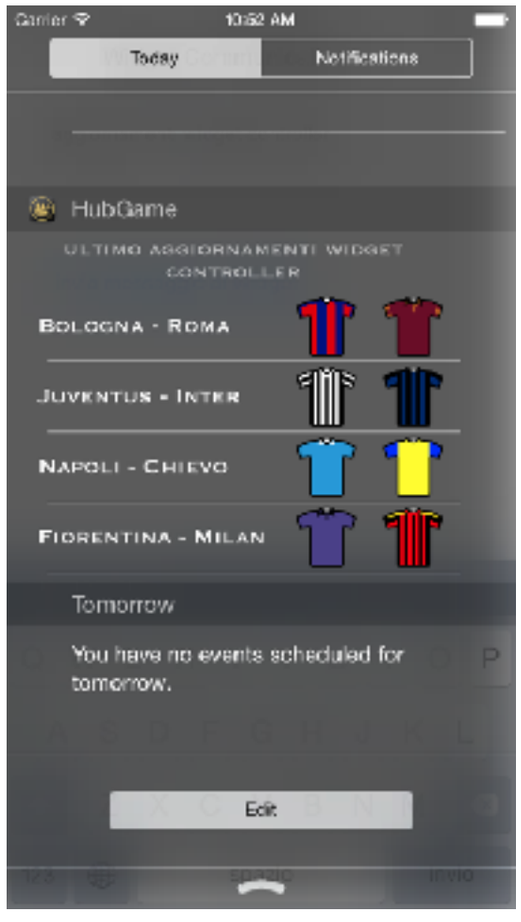
- e successivamente ad una determinata azione sul widget associo tale codice:

```
• NSString *pathSchemeUrl = @"WidgetScheme://it.mobilesoft.TestWidget?risultati=YES";
 NSURL *urlToOpen = [[NSURL alloc] initWithString:pathSchemeUrl];
[myExtensionContext openURL:urlToOpen completionHandler:^(BOOL success) {
    ;
}];
```

Controller Widget

- Il controller del widget gode di vita propria, estende UIViewController, di conseguenza possiede tutti e metodi di caricamento, visione della vista che vengono scatenati allo “swipe” del notification center

- Vediamo un esempio pratico di un progetto per la visualizzazione delle partite calcistiche da giocare, visualizzate in un widget contenente una tabella



Interactive Notification

What’s new in iOS notifications

Interactive Notification

- Permettono all'utente di interagire con la notifica stessa. Es: rispondere "si", "no" ad una notifica d'invito

- possibilità di eseguire in background l'azione scelta dall'utente

Interactive Notification

- Per definire una notifica interattiva abbiamo bisogno di:
  - definire delle ***Action***
  - definire delle ***category*** (insieme di action)
  - definire il ***type*** di una notifica
  - assegnare a tale notifica le ***settings*** (risultanti dai precedenti parametri)

UIMutableUserNotificationAction class

- tale classe permette di definire l'azione associata ad ogni action nella notifica. Le property da valorizzare per tale oggetto sono:

    - *identifier*
    - *activationMode:*
      rappresenta il modo con il quale l'action deve essere eseguita (background / foreground)
    - *authenticationRequired:*
      indica se l'utente deve sbloccare il device prima che l'action venga eseguita. Se activationMode=foreground, tale property è = YES
    - *destructive*
      quando viene eseguita una action con destructive=YES allora la notifica viene distrutta.

UIMutableUserNotificationCategory class

- possiede anch'esso un ***identifier***

- una category rappresenta un insieme di action per una determinata notifica.

- ad ogni category impostiamo le action in base al context (default o minimal)

UIUserNotificationSettings class

- un oggetto di questo tipo contiene il tipo di notifica (alert, sound, etc…) e l’insieme di category per ogni notifica

```
UIUserNotificationSettings *settings = [UIUserNotificationSettings
        settingsForTypes:userNotificationTypes categories:categories];
        [application registerUserNotificationSettings:settings];
```

handling notifications

- L’app chiama questi metodi contenuti nell’ *app delegate al tap dell’utente su una determinata action.*
  - *application:handleActionWithIdentifier:forRemoteNotification:completionHandler:*
  - *application:handleActionWithIdentifier:forLocalNotification:completionHandler:*
- *tali metodi prendono come parametro l’identifier dell’ action così da permettere allo sviluppatore di agire diversamente in base all’ action scelta dallo user.*

invio notifiche

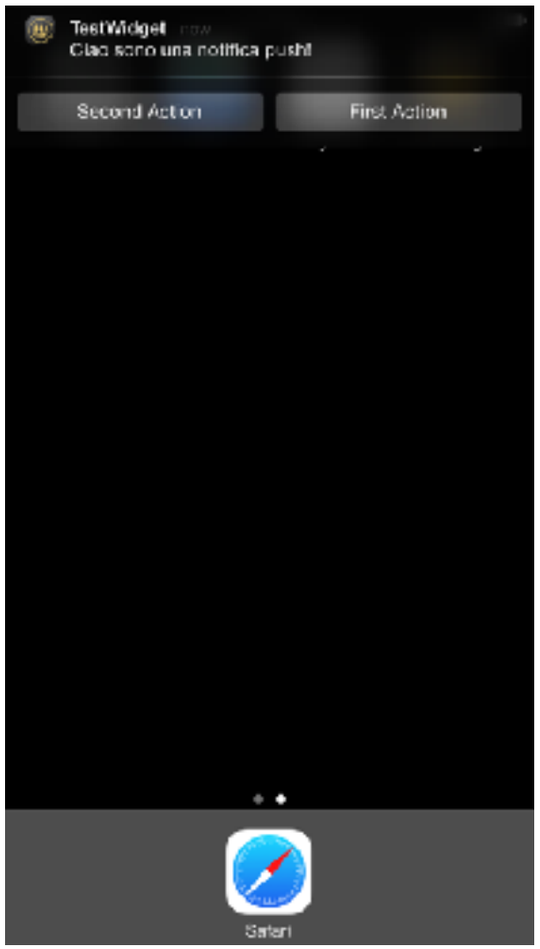
- localmente:

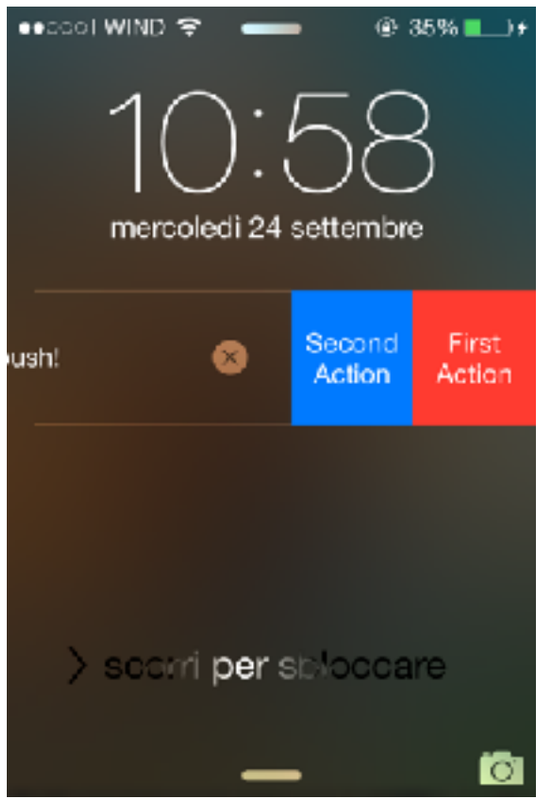
```
UILocalNotification *notification = [[UILocalNotification alloc] init];
notification.category = @"FIRST_CATEGORY";
notification.alertBody = @"Ciao sono una notifica push local!”;
```

- via server, basta inserire nel payload, tale json:

```
{
    "alert": "ciao sono una interactive push",
    "badge": "Increment",
    "sound": "chime",
    "title": "From parse",
    "category" : "FIRST_CATEGORY"
}
```

Design


TestWidget
Ciao sono una notifica push!
Second Action
First Action
Safari


WIND
35%
10:58
mercoledì 24 settembre
Second Action
First Action
scorri per sbloccare


WIND
11:20
58%
TestWidget
Ciao sono una notifica push!
Apri
First Action
Second Action
Third Action
Chiudi
240
Telefono
Calendario
Mail
Safari