



UNIVERSITÀ DEGLI STUDI DI UDINE

Dipartimento di Scienze Matematiche, Informatiche e Fisiche

Corso di laurea in Tecnologie Web Multimediali

**PROGETTAZIONE E SVILUPPO DI
UN'APPLICAZIONE PER LA
FIDELIZZAZIONE E IL REWARDING
DEGLI UTENTI**

RELATORE
Prof. STEFANO BURIGAT

LAUREANDO
RICCARDO CARANFIL

ANNO ACCADEMICO 2019/2020

Questa pagina è lasciata intenzionalmente bianca.

*Ringrazio la mia famiglia e
tutti i miei amici per avermi accompagnato lungo questo
percorso*

IV

Citazione

- *Tizio Caio* -

Ringraziamenti:

Da scrivere

Indice

Introduzione	1
1 Progettazione iniziale	3
1.1 Requisiti essenziali	3
1.2 La navigazione dinamica	4
1.2.1 Tipologie di navigazione	5
1.2.2 Il Coordinator Pattern	6
1.3 Il QIX Shake	8
1.4 Le animazioni interattive	8
2 Dettagli implementativi	9
3 Conclusioni	11
4 Riconoscimenti	13

Questa pagina è lasciata intenzionalmente bianca.

Introduzione

L'obiettivo principale del tirocinio presso **Urbana Smart Solutions srl** [1] è stata la creazione di un'applicazione iOS atta alla fidelizzazione e al rewarding di un target di utenti specifico. I principali clienti di QIX sono delle FMCG¹, ossia delle compagnie che vendono beni di consumo a basso costo e molto velocemente.

Tali compagnie attraverso i loro prodotti possono creare diverse tipologie di eventi e gli utenti dell'applicazione possono accedervi e vincere dei QIX coins, ossia dei punti con cui comprare o avere degli sconti sui beni venduti.

L'elemento cardine dell'app è il "QIX Shake" ossia l'attivazione di un particolare servizio e la possibilità di vincere dei QIX coins agitando lo smartphone. Tale funzionalità si divide in diverse tipologie:

1. **TV Shake:** un qualsiasi utente di QIX potrà tentare di vincere dei punti
2. **Read Shake:** i QIX coins vengono consegnati una volta letto una sorta di questionario
3. **Video Shake:** Dopo aver guardato un video
4. **Scan shake:** dopo aver scannerizzato il barcode di un prodotto delle FMCG
5. **Receipt Shake:** dopo aver scannerizzato uno scontrino
6. **Stadium Shake:** Ascoltando della musica negli stadi con una watermark non udibile dall'uomo

¹Fast Moving Consumer Goods

L'obiettivo principale è stato quindi quello di progettare un prototipo iniziale che rispettasse determinati requisiti

Capitolo 1

Progettazione iniziale

Il mio compito nello sviluppo dell'applicazione è stato quello di creare un prototipo iniziale avendo a disposizione un mock up creato con proto.io [2] e una serie di requisiti essenziali.

1.1 Requisiti essenziali

La base di partenza di QIX sono state delle funzionalità essenziali e sostanzialmente molto difficili da inserire in una versione dell'app già avanzata. È stato quindi deciso di creare un prototipo di partenza avente i seguenti requisiti:

- **Navigazione dinamica:** L'applicazione deve gestire dei cambiamenti di contesto dinamici: dev'essere possibile mostrare all'utente contenuti dinamici indipendentemente dal contesto in cui si trova.
- **QIX Shake:** L'utente deve poter agitare lo smartphone in qualsiasi sezione dell'applicazione e il risultato deve essere basato sul contesto attuale o su delle direttive dettate da delle Rest API;
- **Animazioni interattive:** L'intera applicazione dev'essere progettata in modo tale da presentare all'utente delle **animazioni interattive** in stile CardView [3] disponibili in qualunque sezione o vista in cui si trovi l'utente e definite dal contesto attuale;

Le animazioni in questione devono essere progettate in pagine, in cui ogni pagina può contenere più CardView. L'utente vedrà in un determinato momento una e soltanto una pagina.

Ogni CardView deve essere trascinabile dall'utente e deve interagire con le altre CardView della pagine. Quando l'utente usa una forza di trascinamento superiore a un valore di soglia tutte le viste devono cadere per gravità;

Tale gravità finirà con la fine dell'animazione o l'apparizione di una nuova pagina se presente;

- **Autenticazione:** L'applicazione deve supportare tre diversi stati o modalità di autenticazione:
 1. **Trial Mode:** l'utente è anonimo, esiste solo un id per tenere traccia dei suoi QIX coins.
 2. **Signed Mode:** l'utente ha inserito il numero di telefono e il suo genere;
 3. **Pro Mode:** l'utente aggiunge dei dati su se stesso o collega il suo account a dei social media come Facebook, Google o Instagram;

Si nota facilmente che non esiste uno stato in cui l'utente non è registrato: questo perché per tenere traccia dei suoi QIX coins e di altri dati utili è necessario avere un riferimento all'utente;

- **DeepLinks:** L'applicazione deve poter essere avviata dinamicamente attraverso dei **Deep Links** [4]; E deve essere in grado di gestirli in base al contesto dell'utente;

1.2 La navigazione dinamica

Prima di entrare nel merito della soluzione al problema, elenco brevemente gli standard di navigazione delle app iOS.

Ogni applicazione può avere degli **UINavigationController** [5], ossia dei contenitori di **UIViewController** [6] che vengono utilizzati per mantenere lo stack di navigazione e gestire le transizioni tra due UIViewController.

Nella figura 1.1 si nota facilmente come il Navigation controller gestisce un'array di View Controller e una sola navigation bar. In iOS sono infatti innate molte animazioni di navigazione che è utile sfruttare, piuttosto di creare componenti custom poi difficili da rendere interattivi.

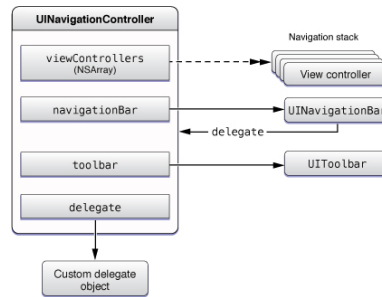


Figura 1.1: Navigation controller scheme

1.2.1 Tipologie di navigazione

Esistono tre tipologie base di navigazione:

1. **Push**: un UINavigationController avente un navigation controller può rendere visibile un altro ViewController attraverso la funzione "push-ViewController" di un Navigation controller

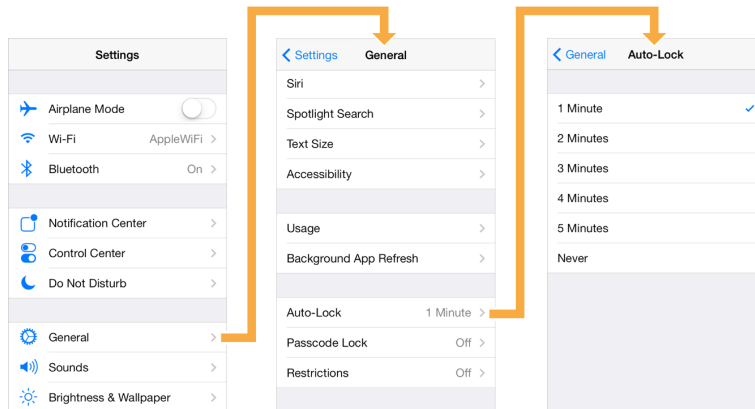


Figura 1.2: Presantazione di un ViewController tramite push

2. **Modal**: un ViewController può presentare un altro ViewController senza necessariamente avere un Navigation Controller, l'animazione standard è dal basso verso l'alto come in figura 1.3

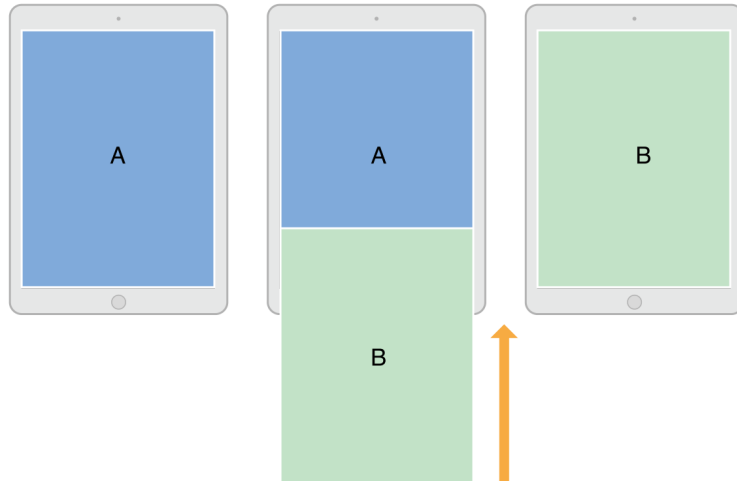


Figura 1.3: Presentazione di un ViewController tramite modal

3. **Segue**: Una segue non è altro che un link tra due view controller attraverso un'interfaccia grafica. In base alla tipologia cambia il tipo di navigazione (Modal o Push)

Avendo definito i principali metodi di navigazione tra ViewController torniamo al problema iniziale: *Come possiamo rendere dinamica la navigazione?*

A seguito di uno studio approfondito di varie tecniche di navigazione iOS ho scelto di utilizzare il **Coordinator Pattern** [7].

1.2.2 Il Coordinator Pattern

Generalmente in iOS l'intera logica di un ViewController viene scritta nel ViewController stesso, creando spesso file di grosse dimensioni e disordine generale. Il Coordinator Pattern è nato proprio per rendere le applicazioni più scalabili e leggere.

Ogni ViewController infatti delega tutte le decisioni al suo Coordinator che in base a determinate logiche deciderà i passi successivi.

Ogni Coordinator può controllare un ViewController o più Coordinator, questo rende le viste indipendenti tra di loro e rende ogni ViewController totalmente invisibile agli altri.

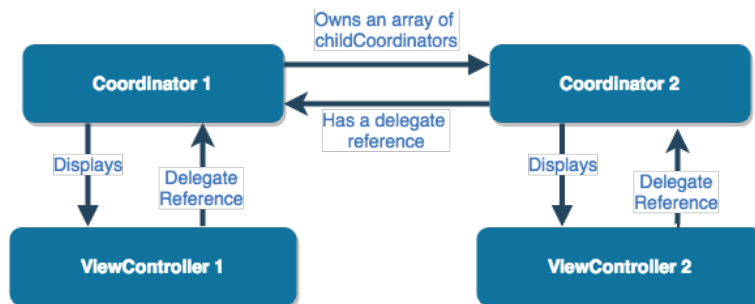


Figura 1.4: Il Coordinator Pattern

La responsabilità dei coordinator è infatti la navigazione, come un navigation controller gestisce i suoi View Controller, un coordinator gestisce i suoi figli e questo rende ogni vista o flow di navigazione totalmente indipendente dal resto dell'applicazione.

Per navigare tra i view controller vengono generalmente usate le tipologie di navigazione descritte nella sezione 1.2.1, tranne le segue, che essendo definite da vista grafica renderebbero la navigazione statica e fissata su determinati ViewController.

Di seguito in figura 1.5 presento uno schema dell'utilizzo di due coordinator per la gestione di una lista di prodotti e il carrello.

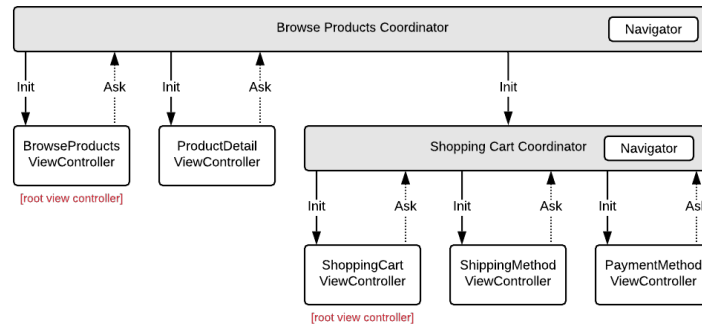


Figura 1.5: Esempio di coordinator pattern

Come si evince dall'immagine è presente in entrambi i coordinator è presente un oggetto **navigator** che sarà in gestione di un UINavigationController

1.3 Il QIX Shake

In iOS ogni UIViewController risponde a degli eventi. L'evento designato per lo shake è

```
func motionEnded(_ motion: UIEvent.EventSubtype,
  with event: UIEvent?)
```

In caso di shake infatti motion sarà uguale a .motionShake Per rendere disponibile l'evento "shake" in un qualunque ViewController la soluzione è stata abbastanza semplice: è bastato l'utilizzo di un ViewController Genitore e attraverso l'ereditarietà ogni view controller è in grado di eseguire la stessa funzionalità.

In questo caso è stato optato l'utilizzo delle notifiche locali: quando avviene uno shake i view controller inviano una notifica globale e solo gli observer vi hanno accesso.

1.4 Le animazioni interattive

Capitolo 2

Dettagli implementativi

Questa pagina è lasciata intenzionalmente bianca.

Capitolo 3

Conclusioni

Questa pagina è lasciata intenzionalmente bianca.

Capitolo 4

Riconoscimenti

Questa pagina è lasciata intenzionalmente bianca.

Bibliografia

- [1] Urbana Smart Solutions srl. <https://www.urbanasolutions.net>.
- [2] Proto.io. <https://proto.io>.
- [3] CardView. Definiamo cardview una vista rettangolo con un border radius e del contenuto di testo e immagini variabile.
- [4] iOS Universal Links. <https://developer.apple.com/ios/universal-links/>.
- [5] Apple Inc. UINavigationController. <https://developer.apple.com/documentation/uikit/uinavigation>.
- [6] Apple Inc. UIViewController. <https://developer.apple.com/documentation/uikit/uiviewcontroller>.
- [7] Soroush Khanlou. Introdotto nel 2015 alla nsspain conference.

Questa pagina è lasciata intenzionalmente bianca.

Elenco delle figure

1.1	Navigation controller scheme	5
1.2	Presentazione di un ViewController tramite push	5
1.3	Presentazione di un ViewController tramite modal	6
1.4	Il Coordinator Pattern	7
1.5	Esempio di coordinator pattern	8