

# 15 Puzzle game

---

Adriano Cardace

25 luglio 2017

## 1 INTRODUZIONE

In questo progetto si è realizzato il gioco dello spaccaquindici per IOS. L'applicazione prevede una schermata iniziale in cui l'utente deve effettuare il login per poter accedere alla home screen. Il gioco consiste nel riordinare 16 caselle disposte in una griglia 4x4 per ricomporre un'immagine. Attraverso le preferenze, l'utente ha la possibilità di scegliere l'immagine del gioco, utilizzare un'immagine del web fornendone semplicemente l'url, oppure attivare/dissattivare l'audio di gioco. Grazie all'utilizzo di Firebase, un realtime database sviluppato da Google, vengono salvate anche le migliori sessioni di gioco per ciascun utente registrato.

## 2 USER INTERFACE

Dopo il lancio dell'applicazione viene mostrata una pagina di login; se l'utente non dispone di un account può facilmente crearne uno cliccando sull'apposito bottone. Se necessario, è possibile anche effettuare il reset della password. Una volta eseguito il login viene presentata la home screen, dalla quale l'utente può effettuare tre azioni, iniziare il gioco, fare logout oppure accedere alle impostazioni. L'oggetto principale della schermata di gioco è ovviamente la griglia da riordinare, che viene disordinata tramite un numero prefissato di mosse random, in modo da far sì che sia sempre possibile ricostruire l'immagine originale. Delle 16 caselle solo una è vuota, e cliccando su una delle celle adiacenti è possibile effettuare una mossa. In questa stessa view, il player può visualizzare il suo best score in termini di tempo o nel numero minimo di mosse per una partita vinta, e contemporaneamente può vedere le mosse e il tempo trascorso nella partita corrente grazie all'utilizzo di un timer. Dalla home screen,

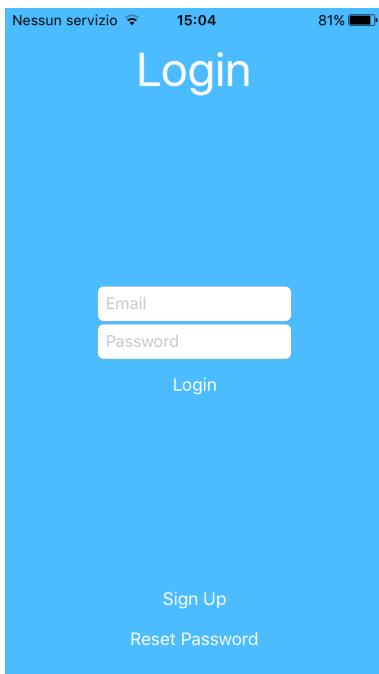


Figura 2.1: Flower one.



Figura 2.2: Flower two.

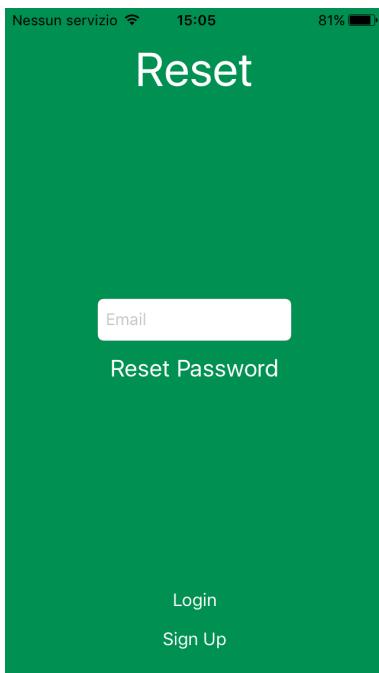


Figura 2.3: Flower one.

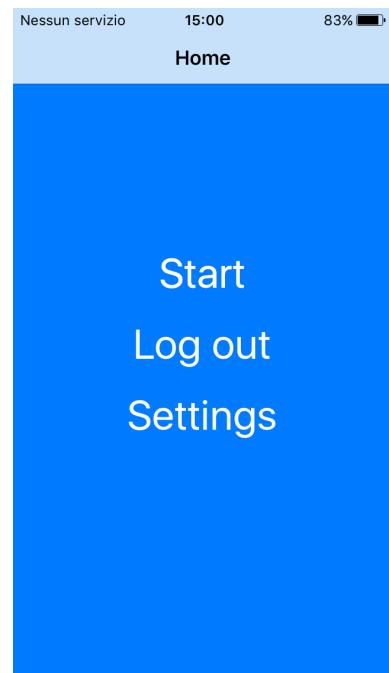


Figura 2.4: Flower two.

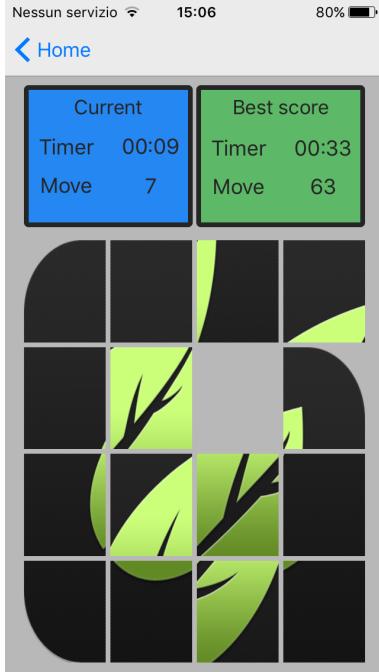


Figura 2.5: Game screen

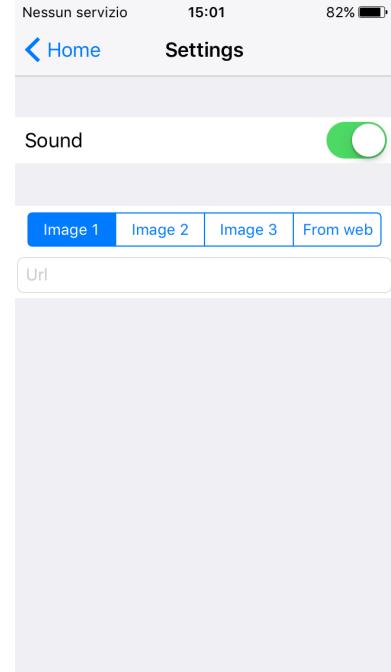


Figura 2.6: Settings.

cliccando sul bottone *Settings* è possibile accedere alle impostazioni di gioco, che consistono nell'attivare/disattivare il suono, oppure nel selezionare una delle immagini predefinite o inserire l'url per un'immagine scelta dalla rete.

Per poter utilizzare il gioco sia su iPhone che su iPad è stato utilizzato uno *split view controller* in combinazione con due *navigation controller*. In particolare uno *split view controller* possiede due oggetti principali, il primo, detto *master*, consiste nella schermata principale e viene mostrata come un'unica schermata su iPhone, mentre compare nella sidebar laterale su iPad. Il secondo elemento è detto *detail*, e si tratta delle schermate visualizzata sulla destra in un iPad, mentre viene vista ancora una volta come un'unica vista su iPhone a causa delle limitate dimensioni. Grazie all'utilizzo dello *split view controller* insieme ai due *navigation controller* dunque, nel caso dell'iPhone il player vede come schermata iniziale la home screen, dopodiché può accedere alle impostazioni o al gioco cliccando sui bottoni relativi e tornare alla home screen con un semplice swipe verso destra.

Per quanto riguarda l'iPad invece, l'utente vede su tutto lo schermo la schermata di gioco e in una sidebar la home screen (figura 2.7), dalla quale può accedere alle impostazioni che prenderanno il posto della schermata di gioco. Grazie all'*Interface builder* di Xcode inoltre, sono state definite alcune regole particolari che permettono di disporre in maniera differente gli oggetti della schermata di gioco in base alla modalità d'utilizzo dell'iPhone, ovvero in modalità portrait o landscape. In quest'ultima posizione infatti, nel caso di iPhone, la dimensione dello schermo in altezza non sarebbe stata sufficiente per mostrare contemporaneamente lo score della partita e la griglia di gioco. Per questo motivo, sfruttando il fatto che tutti i dispositivi Apple sono catalogati come *compact* o *regular*, sono state applicate delle regole per far

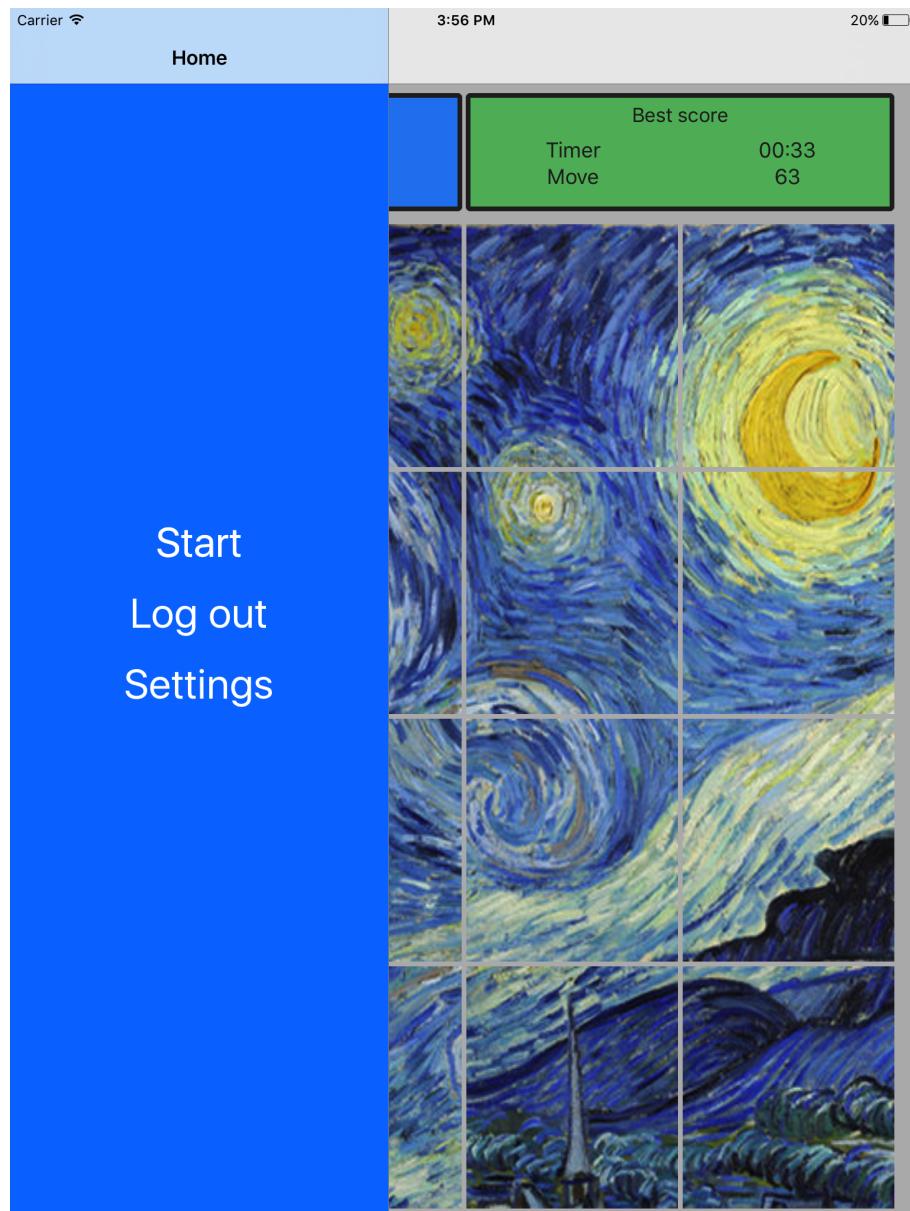


Figura 2.7: Game screen su iPad

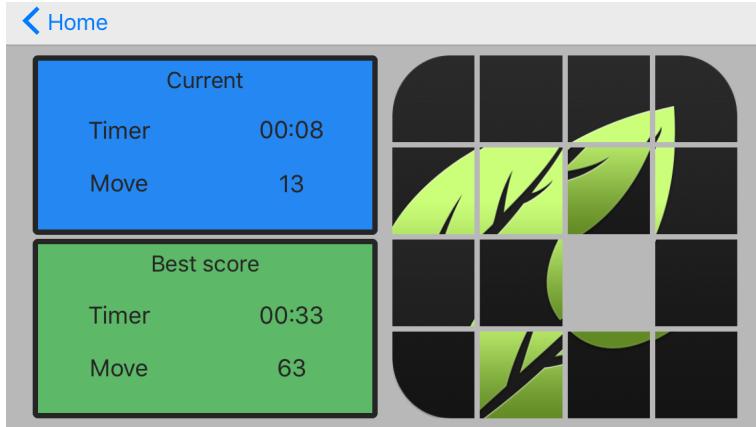


Figura 2.8: Visualizzazione in landscape su iPhone 5

sì che su iPhone (in modalità landscape tutti gli iPhone dalla versione 7 a quelle precedenti sono catalogati come *compact*) lo score attuale sia visualizzato sulla sinistra, mentre la griglia viene disposta sulla destra dello schermo (figura 2.8).

### 3 STRUTTURA

La struttura generale del progetto è illustrata in figura 3.1. Come si può vedere dall'immagine, di fianco alla pagina di login view, c'è una piccola freccia ad indicare che questa schermata è l'entry point della applicazione. La pagina di login, signup, reset sono collegate tra loro sfruttando dei *segue*. Per gestire in modo più efficace queste tre schermate sono state create tre sottoclassi della classe *UIViewController* chiamate *LoginViewController*, *SignUpViewController*, *ResetPasswordViewController* rispettivamente. In tutte queste tre classi si utilizzano le API di Firebase per collegarsi al realtime database e offrire le funzionalità di base all'utente. Ad esempio in *LoginViewController* se l'utente ha inserito correttamente username e password si utilizza la funzione *signin* per effettuare il login, che porta l'utente a visualizzare la sua home screen. Viceversa se il login per qualche ragione non va a buon fine, viene mostrato un alert per avvertire il player. Da notare è che nel file chiamato *AppDelegate*, è stata inserita una funzione chiamata *logUser* che controlla se è presente una sessione corrente, in questo modo non è necessario rieffettuare il login ogniqualvolta si lancia l'applicazione. Se è presente una sessione corrente, viene lanciato la schermata corrispondente al tag *Home*, che corrisponde allo *SplitView Controller*. Come detto precedentemente, questo è collegato a due *Navigation controller*, il primo considerato come master, è quello che contiene la home screen, mentre il secondo considerato come detail che contiene la schermata di gioco. Un problema riscontrato con questa struttura è che di default su iPhone viene caricato come prima cosa la pagina di detail e non quella del master, questo significa che al momento del login l'utente vedrebbe direttamente la schermata di gioco saltando la home screen. La strategia utilizzata per risolvere questo problema è quella di rendere la classe *MasterViewController*, ovvero il controller della home screen, il *Delegate* dello *SplitView Controller* nella funzione *awakeFromNib()*, poiché

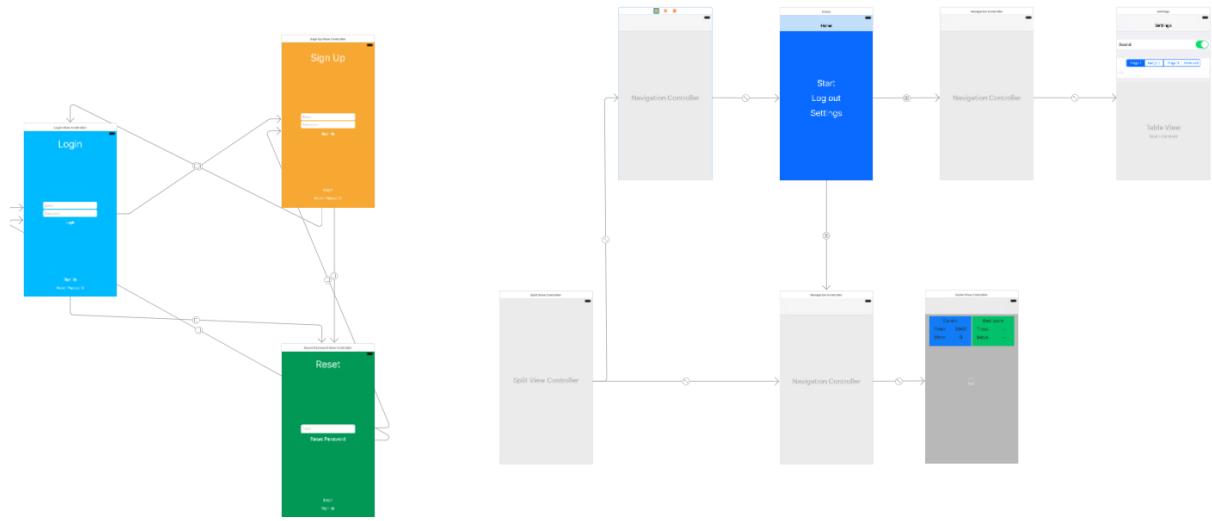


Figura 3.1: Struttura generale

questa funzione viene invocata molto presto. Dopodiché è necessario fare overriding della funzione `splitViewController(splitViewController: UISplitViewController, collapseSecondary secondaryViewController: UIViewController, onto primaryViewController: UIViewController)` e sostanzialmente non fare nulla, in modo da evitare che la `secondaryViewController` (la schermata detail) non rimpiazzi la `primaryViewController` (la home screen).

Dalla home screen l'utente ha tre possibilità:

- Cliccare Start e dunque lanciare un *segue* che porta al *Navigation controller* contenente il controller del gioco.
- Cliccare su logout ed invocare quindi la funzione `signOut()` delle API di Firebase.
- Cliccare su Settings ed effettuare un *segue* che porta ad un terzo *Navigation controller* che contiene le impostazioni.

La view delle impostazioni è gestita dal controller `SettingsTableViewController`. Il compito di questa classe è quello di aggiornare le impostazioni a seconda delle scelte dell'utente. Per fare questo si utilizza una classe chiamata `UserDefaultsManager` che contiene una *computed property* con i relativi metodi getter/setter per ciascuna impostazione. Ad esempio nel momento in cui l'utente accede alle impostazioni e attiva il suono, viene invocato il metodo `updateSwitchSound` (l'action corrispondente al cambio di stato dello switch del suono), in cui si utilizza il metodo `set` della computed property relativa definita in `UserDefaultsManager`. L'ultima classe da analizzare è `GameViewController`, si tratta del controller del gioco e qui dentro ne è contenuta tutta la logica. Il modello consiste principalmente in:

- Un array chiamato `cellState` che contiene 16 valori da 1 a 16; inizialmente le caselle sono tutte ordinate, di conseguenza la casella 1 sarà in posizione 0, la cassella 2 in posizione 1 e così via.

- Una variabile di tipo `UIImageView` chiamata `image` che rappresenta l'immagine da ricomporre nel gioco.
- Una variabile `current` che rappresenta la cella vuota nella griglia 4x4 (inizialmente si tratta della casella numero 16 in basso a destra).
- Una variabile `nsteps` che rappresenta il numero di passi random da eseguire inizialmente.
- Due variabili booleane chiamate `gameStarted` e `timerStarted` che rappresentano lo stato del gioco.

Inizialmente sia `gameStarted` che `timerStarted` sono entrambe false, e nella funzione `viewDidLoad()`, quando tutti gli `Outlets` sono già stati caricati, si preleva l'immagine scelta dell'utente in base alle impostazioni correnti. Questa operazione richiede particolare attenzione se l'utente ha scelto di utilizzare un'immagine del web, in tal caso infatti, per evitare di bloccare l'interazione con l'utente, è necessario creare un nuovo thread asincrono che si occupi del download dell'immagine della rete. Questo viene fatto tramite l'istruzione `DispatchQueue.global()` e implementando al suo interno una closure da eseguire asincronamente. Un secondo problema è che tutto ciò che riguarda l'user interface deve essere eseguito nella coda d'esecuzione principale, per questo motivo all'interno della closure discussa precedentemente è importante ricordarsi di effettuare il set della variabile `image` all'interno di `DispatchQueue.main.async`, questo perché non appena l'immagine viene settata viene invocato il metodo  `didSet()` che deve eseguire operazioni sulla view. Il compito di questo metodo è quello di spezzare l'immagine in 15 parti, assegnare ciascuna di esse ad uno dei bottoni e successivamente eseguire `nsteps` mosse casuali. Su ciascun bottone è definito un action chiamata `moveAction()` che viene invocata quando l'utente clicca su uno dei bottoni. Come prima cosa si controlla se il bottone toccato è uno degli adiacenti alla casella vuota (mantenuta in `current`), dopodiché si effettua lo swap delle due celle nell'array `cellState` e si scambiano le immagini di background dei due bottoni. A questo punto il gioco è iniziato, dunque si setta a true le variabili `gameStarted` e `timerStarted`. Dopo aver effettuato una mossa si invoca la funzione `checkWin()`, per verificare se si è tornati nello stato iniziale, ovvero l'array `cellState` contiene la casella 1 in posizione 0, la cassella 2 in posizione 1 e così via, in tal caso un alert viene mostrato al player per notificare la fine del gioco ed eventualmente il nuovo record viene memorizzato nel database online. La scelta di utilizzare un database online è dovuta al fatto che in questo modo l'utente può giocare da qualunque device Apple.

## 4 CONCLUSIONI ED ESTENSIONI

In questo report si è analizzato la struttura del gioco dello spaccaquindici realizzato per IOS. Per rendere il gioco disponibile su tutte le piattaforme è stato necessario utilizzare diversi componenti come uno *Splitview controller* e un *Navigation controller*. Una eventuale estensione futura potrebbe consistere nel dare la possibilità agli utenti di condividere sui social i loro migliori risultati.