Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Scienze di Internet

Un'applicazione iOS per la gestione delle designazioni arbitrali

Relatore:
Chiar.mo Prof.
Luciano Bononi

Correlatore:
Chiar.mo Dottor.
Luca Lipparini

Presentata da:
Fabrizio Barrano

Sessione II
2013/2014

a mio fratello

Indice

Introduzione

L'obiettivo della tesi è quello di sviluppare un'applicazione mobile per *iOS* al fine di creare un servizio all'utente che semplifichi il processo di consultazione della designazione della gara. Ho iniziato la mia esperienza arbitrale nell'Ottobre del 2008, quando per curiosità mi sono iscritto al corso gratuito organizzato dalla sezione di Bologna; da quel momento fare l'arbitro è diventata una passione. L'idea di sviluppare un'applicazione mobile è nata nel momento in cui ho riscontrato difficoltà, insieme ai miei colleghi, nel consultare il portale *Web* ogni qualvolta si presentasse la necessità di visualizzare la designazione arbitrale. In mancanza di un'applicazione mobile che permettesse l'accesso al portale *Web* dell'associazione, denominato *Sinfonia 4 You*, ho preso in considerazione la possibilià di sviluppare questo servizio. Per rendere concreta l'idea, ho avuto bisogno del supporto informatico dell'*Associazione Italiana Arbitri*.
Nel corso della mia carriera da arbitro, ho avuto modo di conoscere i responsabili informatici nazionali dell'associazione, così da renderli partecipi della mia idea. Intuendo le potenzialità del lavoro che volevo sviluppare, ho avuto la possibilità di essere invitato, attraverso il responsabile informatico della sezione di Bologna, nonché mio correlatore Luca Lipparini, di essere invitato a Coverciano (FI) per confrontarmi con il comitato informatico dell'associazione. All'incontro, ho avuto modo di far vedere una simulazione del prototipo che fino ad allora avevo sviluppato. Da quel momento, la collaborazione è stata concreta e fine ad un obiettivo comune. I responsabili hanno esposto la loro idea di volere già sviluppare un'applicazione mobile simile per le piattaforme *Android* e *Windows Mobile*. Tuttavia, la mia grande passione per il mondo *Apple* mi ha permesso di focalizzarmi sullo sviluppo dell'applicazione su *iOS*.
Il primo capitolo, ***Scenario di Riferimento***, descrive la storia di *Apple Inc.*, mette in evidenza i particolari della prima creazione dell'*iPhone*, i suoi sviluppi e la rivoluzione tecnologica che ne è conseguita. È stata inoltre approfondita la questione sul mercato delle applicazioni mobile di *Apple Inc.* e su come quest'ultima, mediante l'invenzione di *App Store*, sia riuscita ad imporsi nel mercato. Successivamente si descrivono in maniera dettagliata l'organizzazione dell'*Associazione Italiana Arbitri* e la formazione dell'organigramma per rendere più chiara la gestione delle designazioni arbitrali. Di seguito viene illustrato il portale *Web*, denominato *Sinfonia 4 You*, che è stato sviluppato in passato dagli organi informatici e che tutt'ora viene utilizzato da tutte le sezioni arbitrali d'Ita-

lia.
Nel secondo capitolo, ***Analisi degli Aspetti Tecnologici***, sono elencate e spiegate dettagliatamente tutte le tecnologie utilizzate nel progetto. La tecnologia più importante utilizzata è l'ambiente di sviluppo per la creazione e la simulazione delle applicazioni mobile, *Xcode*.
Il terzo capitolo, ***Progettazione App***, è il più corposo poiché tratta di tutte le fasi della progettazione. Il primo passo è stato quello di scegliere cosa progettare, ed in seguito sviluppare, dal portale *Sinfonia 4 You*. Ho scelto di sviluppare, nell'applicazione mobile, la sezione riguardante le designazioni arbitrali, in quanto contiene le informazioni che più si utilizzano durante la fase successiva alla designazione effettuata dagli organi competenti. Utilizzando il software *Balsamiq Mockups* è stato possibile creare una bozza di quanto successivamente sviluppato in *Xcode*. In seguito viene descritta l'architettura del progetto e come questo sia stato strutturato attraverso il *Model-View-Controller*. L'integrazione con il *Backend* è di fondamentale importanza per rendere l'applicazione mobile il più funzionale possibile perché si è potuto, tramite i *JSON*, caricare i dati da remoto nelle *View*. Per fare ciò mi sono servito dello strumento *CocoaPods*, che mi ha permesso di gestire le dipendenze con i *framework* utilizzati nel progetto.
Il quarto capitolo, ***Implementazione***, parla della implementazione dell'applicazione. Viene spiegata la creazione di tutte le classi *Objective-C* e la loro interazione con il mondo esterno insieme alla *Storyboard* che guarda invece alla navigazione dell'utente.
Nel quinto capitolo, ***Testing***, viene spiegata la funzionalità dell'App. Attraverso gli screenshot viene mostrato, in maniera dettagliata, come l'utente può interagire con l'App nelle *View* che visualizzerà.

Capitolo 1

Scenario di Riferimento

1.1 Apple

Apple Inc. è un'azienda informatica statunitense che produce sistemi operativi, computer e dispositivi multimediali con sede a Cupertino, nello stato della California.
È stata fondata da Steve Jobs, Steven Wozniak e Ronald Wayne nel 1976 ed è conosciuta in tutto il mondo, dai primi anni ottanta, grazie ad una delle prime produzioni di computer *Macintosh* e negli anni successivi grazie ad *iTunes*, *all'iPhone* e *all'iPad*.
A partire dall'Agosto 2011, *Apple Inc.* è una delle più grandi aziende al mondo per capitalizzazione azionaria e di maggior valore al mondo oltre ad essere, ad oggi, il secondo marchio più importante dietro soltanto a *Google*.

1.1.1 iPhone

Il 9 Gennaio del 2007, Steve Jobs[12], presenta al *keynote*[1] un dispositivo che racchiude tre prodotti in uno, *l'iPhone*. Un *iPod* con controlli touch, un rivoluzionario telefono mobile e un dispositivo per navigare in internet. L'allora *CEO*[2] della *Apple Inc.* ha dichiarato che è molto raro come un'azienda possa aver potuto rivoluzionare un settore della tecnologia, in questo caso della telefonia.
Prendendo il nome di "Melafonino", l'*iPhone* fu descritto dallo stesso inventore come il *Multitasking* che permetteva di passare, ad esempio, dalla musica alla mail con estrema facilità. Grazie al solo utilizzo delle dita tutto questo era possibile. L'*iPhone* sconvolse non poco il mercato della telefonia suscitando molta curiosità e critiche. Passarono sei mesi dalla presentazione all'uscita nel mercato della telefonia, ma fu disponibile soltanto in quattro nazioni con sole quattro compagnie telefoniche.

[1] Evento organizzato da *Apple Inc.* per la presentazione di nuovi software o device
[2] Chief Executive Officer

1.1.2 Rivoluzione del Multi-touch

L'idea del *Multi-touch*[8], che aveva in mente Steve Jobs, era quella di creare un dispositivo che non avesse bisogno né della tastiera fisica, né dello stilo. In pratica la capacità di elaborare simultaneamente impulsi diversi. Nel giro di pochi mesi ex *CEO* della *Apple Inc.* formò un team per la progettazione e la creazione del *multi-touch*. Fu subito elaborata l'idea dello scrolling inerziale. Una funzionalità che permettesse all'utente di passare le dita sullo schermo e muovere le immagini come se fossero dotate di fisicità. È stata una vera rivoluzione per il mondo degli smartphone e per quello dei tablet.

1.1.3 Storia Modelli

La storia [8] dei modelli del "Melafonino" ha inizio dal modello *iPhone 2G* che era dotato di uno schermo da 3.5 pollici con una risoluzione di 480x320 pixel. Il tutto era sfruttato da un processore con architettura *ARM*[3] sviluppato da *Samsung*, una scheda grafica *PowerVR MBX Lite 3D*, una batteria non rimovibile da 1400 mAh e 128 MB di *RAM*.
Inoltre furono introdotte altri sensori come: l'accelerometro per permettere la rotazione dello schermo in maniera automatica, il sensore di prossimità che consentiva allo schermo di spegnersi quando si avvicina il viso per effettuare una chiamata e un sensore di luce ambientale per adattare la luminosità del display alla luce esterna.
Il primo modello è stato molto criticato per alcune limitazioni come la mancanza del *GPS* (Global Positioning System) e la mancanza del supporto alle linee internet di alta velocità come il *3G UMTS/HSPA*. Non permetteva, inoltre, la rimozione della batteria e a livello software non permetteva il copia/incolla dei testi.
Steve Jobs aveva previsto che la concorrenza avrebbe compiuto diversi anni prima di raggiungere lo stesso livello del suo prodotto.
Il secondo modello rilasciato fu l'*iPhone 3G* il 9 Giugno del 2008. Con questo modello la *Apple Inc.* migliorò i difetti del precedente modello. Ci fu il passaggio dal 2G al 3G come connessione ad internet, disponibilità in 70 paesi a livello internazionale., servizi per le aziende e applicazioni da terze parti. Lo schermo di questo modello rimase invariato come anche il processore e la stessa grafica. La batteria si presentò addirittura meno potente del precedente modello. Invece per la prima volta fece la sua apparizione in uno smartphone il sistema *GPS*. Furono resi possibili due differenti colorazioni, bianco e nero. Precedentemente era disponibile sono in nero.
Quello che fece esplodere la popolarità del terzo modello fu l'introduzione della *App Store* che rese possibile finalmente la possibilità di scaricare applicazioni, come giochi e o utilizzare l'app di Facebook, senza limitarsi ad utilizzare il browser o mandare semplici sms. L'iPhone 3G raggiunse la quota di 20 milioni di unità vendute nel mondo.
Con l'*iPhone 3GS* la *Apple Inc.* presentò un modello identico per quanto riguardo la

[3]Advanced RISC Machine

dimensione e la risoluzione dello schermo ma introducendo un tipo di schermo che limitava le impronte delle dita. Introdusse il doppio della potenza per quanto riguardo la *RAM* e la potenza del processore, anche lo storage arrivo fino a 32 GB e la batteria che con i suoi 1219 mAh consentì di ottenere livelli di batteria accettabili. *iPhone 3GS* si presentava già tre volte più veloce rispetto alla versione precedente.
All'*iPhone 3GS*, l'8 giugno 2009, si deve l'introduzione del controllo vocale.
Il 7 giugno del 2010 fu presentato l'*iPhone 4G* destinato ad essere lo smartphone di maggior successo nella storia della telefonia mobile. Questo modello apportava ben 100 nuove differenze rispetto all'*iPhone 3GS* con un design completamente modificato, una fotocamera innovativa ed uno schermo totalmente rivoluzionato.
Proprio lo schermo si presentava con un pannello retroilluminato a *LED*[4] in cui le immagini avevano una risoluzione davvero rilevante con un angolo di visione migliorato. In più ù stato incrementato il numero di *pixel* nella stessa dimensione di schermo passando da 480x320 a 960x640. Prese il nome di "*Retina display*" e l'occhio umano non riuscì più a distinguere i singoli pixel nello schermo.
L'*iPhone 4S*, presentato il 4 Ottobre 2011, fu presentato insieme ad *iOS 5* e per la prima volta *iCloud*. In questo modello venne presentato per la prima volta l'assistente vocale "*Siri*", che aprì così la strada alla concorrenza per questo tipo di applicativi. Questo modello era identico al precedente come design, così come le dimensioni e la risoluzione del display. La fotocamera posteriore fu aumentata fino a 8 mega *pixel* ed il sensore a cinque lenti. Con il nuovo processore di casa *Apple Inc.* "A5" permise di raddoppiare le prestazioni del melafonino. Divenne una vera e propria console da gioco con il nuovo chip dual-core *PowerVR SGX 543MP2*.
Il 12 Settembre del 2012 fu presentato l'*iPhone 5*. Annunciato come il più veloce, potente e leggero, ma soprattutto per il nuovo display da 4 pollici con una risoluzione maggiore a quella del precedente modello. Per la prima volta l'*iPhone* poteva connettersi alle reti ultra veloci *LTE*[5]. *iOS 6* portò grande stabilità a questo dispositivo.
L'*iPhone 5C e 5S* sono due modelli che per la prima volta *Apple Inc.* presenta nello stesso *keynote*. Il primo è stato pensato come una rivisitazione dell'*iPhone 5* ed il secondo invece, grazie alla sua architettura hardware a 64 bit, si avvicina più che ad un telefonino ad un vero e proprio computer. Per quanto riguarda il modello 5C *Apple Inc.* ha pensato di renderlo disponibile in cinque colorazioni diverse: verde, gialla, blu, rossa e bianca. Inoltre questo modello è praticamente identico, per quanto riguarda l'hardware, al all'*iPhone 5*. L'unica variazione è la potenza incrementata della batteria.
L'*iPhone 5S* esteticamente si presenta pure identico alla precedente generazione, ma le colorazioni sono di tre tipi: oro, grigio siderale e argento. L'hardware di questo modello si presenta totalmente differente a partire dal processore, ormai arrivato all'*A7*. Anche la disponibilità dell'unità flash si presenta fino a 128 GB. La sorpresa maggiore la riserva

[4]Light Emitting Diode
[5]Long Term Evolution 4G

il *Touch ID*. Grazie a questa novità gli utenti Apple possono sbloccare il proprio device attraverso il riconoscimento delle impronti digitali. Anche l'*iOS 7* si presenta con uno stile "minimal" ed essenziale.
Le ultime due versioni presentate sono state l'*iPhone 6* e l'*iPhone 6 Plus* il 19 Settembre 2014. Il primo si presenta totalmente differente da precedente 5S per dimensioni e caratteristiche tecniche, il secondo altrettanto. L'*iPhone 6* ha un display di 4.7 pollici con una risoluzione aumentata fino a 1334x750 pixel, l'iPhone 6 Plus ha invece un display di 5.5 pollici. Entrambi i modelli montano un processore *A8 dual-core*. Queste dimensioni di questi due device hanno totalmente modificato il mercato al quale si affacciava prima *Apple Inc.* per quanto riguarda la telefonia, infatti entra di più in competizione con i concorrenti.

1.1.4 Applicazione Mobile

Un'applicazione mobile è una variante dell'applicazione informatica dedicata ai dispositivi mobile, come gli smartphone ed i tablet. Il termine "App" viene appunto dall'abbreviazione di applicazione.
Quando parliamo di App intendiamo a tutti gli effetti un software che può assomigliare ad una struttura informatica ma che in realtà è molto più semplificata. Progettata in modo da avere più leggerezza, essenzialità e velocità.

1.1.5 App Store

App Store è un servizio creato da *Apple Inc.* disponibile per *iPhone*, *iPod Touch* e *iPad* che permette agli utenti di scaricare ed acquistare le App disponibili. Quest'ultime si dividono in gratuite o a pagamento. A Giugno 2014 sono disponibili nell'*App Store* 1,2 milioni di App sviluppate da terze parti oltre a 75 miliardi di download.

1.1.6 Mercato delle App/SO Smartphone

Nato nel 2008 grazie a *Apple Inc.*, quando lanciò il proprio *App Store*, il mercato della App [10] è cresciuto a dismisura con il passare degli anni.
Solo nel 2013 sono state scaricate oltre 94 miliardi di App. In Italia questo mercato ricopre l'1,6 per cento del PIL con un mercato che vale 24,5 miliardi di euro, con un obiettivo di crescita a 40 miliardi prevista per il 2016.
Si parla di 45 milioni di smartphone e 12 milioni di tablet venduti, numeri che fanno davvero riflettere. Negli store digitali sono presenti più di 2,5 milioni di applicazioni.
Non poco rilevante è la questione occupazionale che crea questo settore in Europa, con 1,8 milioni posti di lavoro creati. In Italia la situazione è leggermente diversa perchè non ha aziende che siano in grado di produrre App tra le prima 50 al mondo al di fuori del

mercato domestico. Gli sviluppatori italiani vivono in una dimensione nazionale.
Nel corso del 2013 il mercato degli smartphone [6] ha superato per la prima volta quota 1 miliardo di pezzi, sulla spinta della continua attenzione da parte del pubblico per i sistemi operativi *Android* e *iOS*. Secondo analisi di mercato, i sistemi operativi di *Google* e *Apple Inc.* costituiscono il 95,7 per cento di tutte le consegne smartphone del quarto trimestre 2013 e il 93,8 per cento di tutte le consegne annuali.
C'è stata chiaramente una forte domanda end-user per i prodotti *Android* e *iOS* nel corso del trimestre e per tutto l'anno. È comunque evidente la differenza tra *Android* e *Apple Inc.* nel rispondere alla domanda. *Android* si affida ad una fetta di mercato che tende ad attirare utenti attraverso prezzi molto bassi ed in generale in qualsiasi segmento di mercato, *Apple Inc.* si presenta con prezzi molto più elevati rivolta al commercio di smartphone catalogati in una fascia alta di qualità. Nonostante queste differenze entrambe le piattaforme hanno incontrato una buona ricezione per le rispettive esperienze d'uso e per la selezione di applicazioni mobile.

1.1.7 Software iOS

Storia
Il sistema operativo è stato presentato il 9 gennaio 2007 al *Macworld Conference & Expo*, e la versione 1.0, ancora priva di nome, è entrata in commercio con il primo iPhone il 29 giugno dello stesso anno. Il 6 marzo 2008 il sistema operativo è stato denominato ufficialmente come "*iPhone OS*".
L'11 luglio 2008 viene pubblicato in concomitanza della vendita di *iPhone 3G* l'aggiornamento a *iPhone OS 2.0* che aggiunge, tra le altre funzioni, il molto atteso App Store e la possibilità di installare applicazioni di terze parti tramite l'app.
iPhone OS 3.0, pubblicato con il modello *iPhone 3GS* il 17 giugno 2009, ha aggiunto molte funzioni che furono richieste dagli utenti, come il copia e incolla del testo. Il primo *iPad*, entrato in commercio nell'aprile 2010, ha avuto inizialmente un "ramo" separato di *iPhone OS 3*, fino all'unificazione con gli altri dispositivi con la versione 4.2.1 del software.
Il quarto rilascio del sistema operativo, pubblicato con *iPhone 4* il 21 giugno 2010, ha aggiunto numerose funzioni quali il *multitasking*[6] per le applicazioni di terze parti. In più sono stati aggiunti *FaceTime*[7] e *iBooks*[8]. In seguito ha preso il nome di "*iOS*" ed ha unificato i vari dispositivi (*iPhone*, *iPod Touch* e *iPad*) con una versione comune.
Il 6 giugno 2011 è stata presentata la quinta versione di *iOS* con numerose nuove funzioni. La sincronizzazione wireless, l'integrazione con il servizio *iCloud*[9] di *Apple Inc.* e un

[6]Multiprocessualità permette di eseguire più programmi contemporaneamente
[7]È un software di videotelefonia integrato da *Apple Inc.* nei dispositivi *iOS* con videocamera frontale
[8]È un'applicazione per la lettura e l'acquisto di libri in formato digitale
[9]iCloud permette la sincronizzazione automatica di dati, contatti, immagini e brani musicali tra i vari dispositivi *Apple Inc.*

rinnovato sistema di notifiche. *iOS 5* è stato reso disponibile per il download agli utenti il 12 ottobre 2011.
L'11 giugno 2012 è stata presentata la sesta versione di *iOS* con un'applicazione mappe completamente rinnovata, nuovissime funzioni e lingue per l'assistente vocale *Siri*, un software basato sul riconoscimento vocale, integrazione con Facebook, nuove funzioni di risposta alle chiamate e novità grafiche. *iOS 6* è disponibile al download dal 19 settembre 2012.
Il 10 giugno 2013 è stata presentata la settima versione di *iOS* con uno stile grafico completamente rinnovato in chiave minimale, che presenta icone molto più semplici e colorate. Altro rinnovamento sostanziale è la totale revisione del *Multitasking*, modificandone il look e le funzionalità, rendendolo più attuale e al pari dei sistemi concorrenti. La data di rilascio agli utenti è stata il 18 settembre 2013.
iOS 8 viene presentato il 2 Giugno 2014. Mantiene la stessa grafica di *iOS 7*, ma presenta nuove funzionalità.

Caratteristiche
Il sistema operativo *iOS* ha quattro livelli di astrazione [11]:

- Cocoa Touch
  In questo livello troviamo la maggior parte delle funzionalità utilizzate all'interno dei dispositivi per *iOS*. Il *framework*[10] che viene utilizzato in ogni progetto è *UIKit*, che permette, ad esempio, la creazione e la gestione dell'interfaccia grafica, la gestione del ciclo di funzionamento di un'App, al gestione dei dati, le notifiche locali ecc.
  Come, ad esempio, il *framework Map Kit* è utilizzato per interagire con le mappe.

- *Media Services*: Questo livello si occupa di grafica, audio, video e animazioni ed è formato da vari framework.

- *Core Services*: In questo livello si parla di tutti quei framework che lavorano ad un livello ancora più basso. Questo consente maggiore flessibilità ma al contempo maggiore complessità.

- *Core OS*: Il livello *Core OS* è formato da *framework* dedicati all'accesso di hardware esterno, alla gestione della memoria, del filesystem e della sicurezza.

[10] Librerie preconfezionate

iOS SDK

Il 17 ottobre 2007, in una lettera aperta scritta nel blog "How News" della *Apple Inc.*, Steve Jobs ha annunciato che un *SDK*[11] sarebbe stato disponibile agli sviluppatori di terze parti nel Febbraio del 2008. L'*SDK* è stato rilasciato il 6 marzo 2008 e permette agli sviluppatori di creare applicazioni per *iPhone* e *iPod touch*, e testarle in un simulatore. Tuttavia il caricamento di una applicazione nei dispositivi è possibile solamente dopo aver pagato una tassa di iscrizione all'*iOS Developer Program*. L'ambiente di sviluppo per *iOS SDK* è *Xcode*.
Gli sviluppatori sono liberi di scegliere qualsiasi prezzo per le loro applicazioni che sono distribuite tramite *App Store*, per le quali riceveranno il 70 percento del ricavo. Essi possono anche optare per rilasciare l'applicazione gratis e non pagheranno nessun costo di rilascio o distribuzione, eccetto la tassa di sottoscrizione al programma *Developer*.

Contenuto SDK

Dato che l'*iPhone* è basato su una variante dello stesso *XNU kernel*. che si trova in *Mac OS X*, gli strumenti usati per lo sviluppo sono basati su *Xcode*.
L'*SDK* è diviso nei seguenti set: All'interno dell'*SDK* è contenuto l'*iPhone Simulator*, un programma usato per emulare il "look and feel" dell'*iPhone* nel desktop dello sviluppatore. Originariamente chiamato *Aspen Simulator*, è stato rinominato con la *Beta 2* dell'*SDK*. Da notare che l'*iPhone Simulator* non è un emulatore ed esegue codice generato per un target x86.
L'*SDK* richiede un *Mac Intel* con *Mac OS X Leopard*. Altri sistemi operativi, inclusi *Microsoft Windows* e vecchie versioni di *Mac OS X*, non sono supportati, ma vi sono applicazioni non ufficiali che permettono ciò.

Diffusione

Il 6 giugno 2011 al *WWDC*[12] viene annunciato che il numero di dispositivi *iOS* venduti ha raggiunto quota 200 milioni, il 18 Febbraio 2012 viene comunicato il superamento di quota 316 milioni. Secondo *StatCounter*, a settembre 2013 *Apple Inc.* è il secondo sistema operativo mobile con il 23 percento del mercato.

Pregi e critiche

Il sistema *Apple Inc.* ha delle peculiarità di funzionamento che ne determinano una diversa fruizione delle applicazioni e dei dispositivi. Uno studio della *Symantec* evidenzia come questo sistema operativo sia più sicuro del maggiore avversario *Android* grazie alle sue peculiarità.
Anche se ha evidenziato che è pi sicuro, soprattutto per via dei minori tentativi di attacco da parte degli hacker, che concentrano gli sforzi su *Android* in quanto maggiormente

[11] Software Development Kit
[12] The Apple Worldwide Developers Conference

diffuso a livello mondiale rispetto a *iOS*.
Con l'aggiornamento del sistema alla versione 4.3, che introduceva un miglioramento del browser e della navigazione si è evidenziato come questi aggiornamenti non fossero abilitati per le applicazioni web (collegamenti internet), questo perchè l'aggiornamento ha effetto solo per il browser e non per il sistema che utilizza un visualizzatore internet integrato, il problema è stato risolto con la versione 5 del sistema operativo.
La particolarità di questo sistema è data anche dal blocco di alcune funzionalità, o in toto o in parte, come ad esempio il *Bluetooth* che può essere usato solo per connettere dispositivi ausiliari, come ad esempio cuffie auricolari, ecc.
Apple Inc. tramite il suo store evita l'installazione di applicazioni non approvate, perché prima di essere approvate vengono vagliate e viene testata la loro sicurezza e un eventuale problematicità nel loro uso, cos come la loro qualità. Questo rende più difficile l'installazione di applicazioni malevoli ma nel contempo limita la libertà dell'utente. I rappresentanti del movimento *Open Source* criticano questo approccio ritenendolo troppo limitante per l'utente e ritengono che dei dispositivi così limitati non possano essere equiparati a dei personal computer. Tuttavia, è possibile evitare questa limitazione di attività attraverso una procedura ormai frequente e dichiarata legale dal Tribunale Federale USA en chiamata *Jailbreak* o in italiano "sblocco", la quale permette l'uso di applicazioni non approvate da *Apple Inc.*, presenti non sull'*App Store* ma su *Cydia*[13] e sblocca anche le "anti-funzionalità" riguardanti il Bluetooth.
Al termine del 2011 sono emersi alcuni bug del sistema, di cui uno riguardante la comunicazione dati tramite reti telefoniche, che venivano permesse nel caso di download di App dallo store indipendentemente dalle impostazioni dell'utente; la seconda, da una vulnerabilità nei test dell'approvazione di App nello store, che permette l'esecuzione di codice non validato.
Nel 2012 è emerso di come le applicazioni potrebbero estrapolare le foto personali dell'utente dal proprio dispositivo, inoltre altre applicazioni come *Path*, memorizzano tutti i contatti, con i rispettivi nomi e cognomi registrati sul cellulare su cui è installata l'applicazione. Problemi del genere vennero poi sistemati, al rilascio di *iOS 6*, con l'introduzione di appositi avvisi e impostazioni che permettono all'utente di consentire o negare alle singole applicazioni l'accesso a contatti, calendari, promemoria e immagini in modo del tutto simile a come già avveniva per la localizzazione.
Nel 2013, all'uscita di *iOS 6.1*, è stata individuata una "falla" di sicurezza che consentiva a chiunque effettuasse una precisa procedura di accedere all'applicazione telefono bypassando il codice di sblocco. In questo modo qualsiasi malintenzionato poteva effettuare telefonate, visualizzare contatti, ecc. Tali problemi si erano già verificati in passato con le versioni 2.0 e 4.1 dell'*iOS* e sono stati corretti con gli aggiornamenti successivi.
Nel Giugno 2013, con la presentazione della settima versione di *iOS*, si sono verificate

[13]Applicazione che consente di utilizzare l'*iPhone* in maniera craccata, ciò consente di personalizzare l'*l'iPhone* raggirando le regole ferree di *Apple Inc.*

innumerevoli critiche avverse alla grande somiglianza delle nuove funzionalità con quelle già presenti da tempo nel rivale *Android*, come le nuove "*Notification center*" e "*Control Center*" e infine la totale somiglianza dei nuovi layout della *lockscreen*, *iTunes Radio*, fotocamera e browser.

1.2 Associazione Italiana Arbitri

L'*Associazione Italiana Arbitri* (AIA) [1] è la settima componente della *FIGC*[14]. Si occupa del reclutamento, della formazione, della gestione tecnica, associativa e disciplinare degli arbitri di calcio italiani. Fondata il 27 agosto 1911, oggi ha sede in via Tevere a Roma, con uffici in Via Campania. Al 31 gennaio 2011 conta 33.113 associati.
L'*AIA* ha il compito di designare arbitri anche per altri due sport oltre al calcio 11: il Calcio a 5 e il Beach Soccer. L'*AIA* è organizzata in 19 Comitati Regionali - *C.R.A.* - e in 211 sezioni arbitrali su tutto il territorio nazionale.
Gli organi tecnici si dividono in:

- *Organi Tecnici Nazionali*
- *Organi Tecnici Periferici*

Gli organi tecnici si occupano delle designazioni di arbitri, assistenti ed osservatori nelle gare di loro competenza.

1.3 Designazione arbitrale AIA

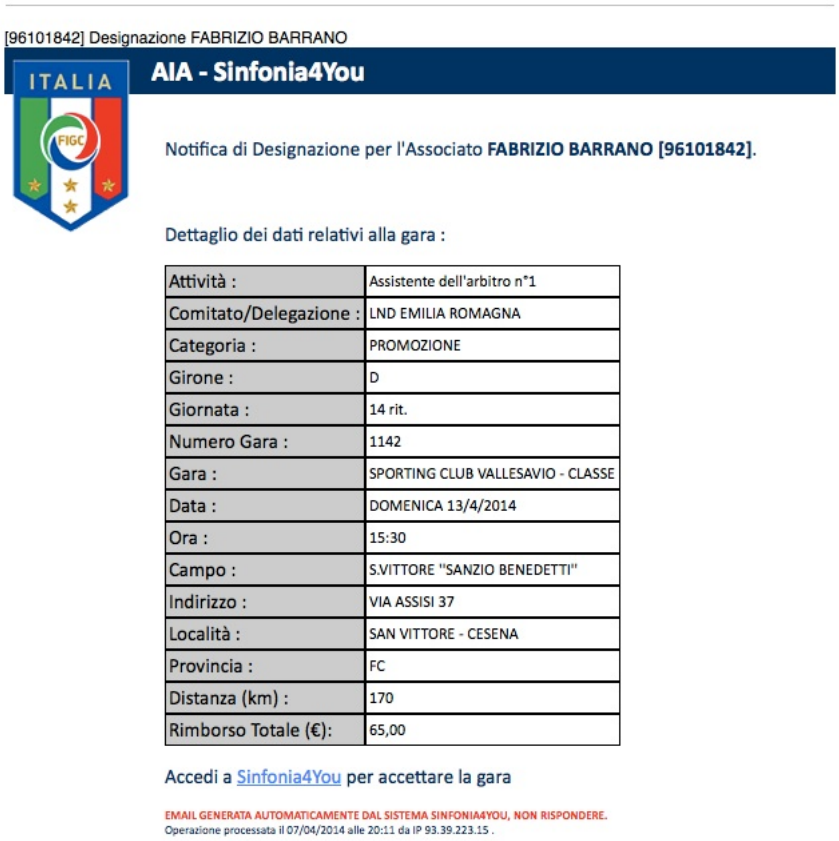
La designazione viene effettuata dagli organi tecnici[15] ad ogni associato e per ogni gara da effettuare. All'associato viene inviata la comunicazione via mail (Figura 1.1) dell'avvenuta designazione, ricevendo un breve riepilogo della gara che è tenuto a svolgere.
Per prendere visione delle specifiche della gara, l'associato dovrà accedere al portale, successivamente potrà prendere visione delle specifiche complete della partita e poi successivamente ha la possibilità di accettare la gara.
Come si può notare dalla figura successiva, nella mail ricevuta dall'associato visualizzerà i seguenti campi:

- Attività
- Comitato/Delegazione
- Categoria

[14]Federazione Italiana Giuoco Calcio

[15]Gli organi tecnici sono quegli associati che sono a capo di un organo tecnico (inteso come commissione) o ne facciano parte come componente.

- Girone
- Giornata
- Numero Gara
- Gara
- Data
- Ora
- Campo
- Indirizzo
- Località
- Provincia
- Distanza (km)
- Rimborso Totale



[96101842] Designazione FABRIZIO BARRANO

AIA - Sinfonia4You

Notifica di Designazione per l'Associato **FABRIZIO BARRANO [96101842]**.

Dettaglio dei dati relativi alla gara :

Attività :	Assistente dell'arbitro n°1
Comitato/Delegazione :	LND EMILIA ROMAGNA
Categoria :	PROMOZIONE
Girone :	D
Giornata :	14 rit.
Numero Gara :	1142
Gara :	SPORTING CLUB VALLESAVIO - CLASSE
Data :	DOMENICA 13/4/2014
Ora :	15:30
Campo :	S.VITTORE "SANZIO BENEDETTI"
Indirizzo :	VIA ASSISI 37
Località :	SAN VITTORE - CESENA
Provincia :	FC
Distanza (km) :	170
Rimborso Totale (€):	65,00

Accedi a Sinfonia4You per accettare la gara

EMAIL GENERATA AUTOMATICAMENTE DAL SISTEMA SINFONIA4YOU, NON RISPONDERE.
Operazione processata il 07/04/2014 alle 20:11 da IP 93.39.223.15 .

Figura 1.1: Esempio mail di ricevuta designazione

1.4 Sinfonia 4 You

Il vecchio software in dotazione a tutte le sezioni si chiamava *Sinfonia*, acronimo di Sistema Informativo Nazionale Integrato AIA, ed era sviluppato in *Clipper*[16].
Ogni sezione ed ogni comitato regionale aveva una sua installazione indipendente in cui l'unica piccola sincronizzazione era fatta a fine anno tra il comitato regionale e le sezioni.
Il procedimento di designazione inizialmente prevedeva la stampa di una lettera di designazione che veniva inviata via posta all'associato; con l'evoluzione dei sistemi, poi, alcuni associati hanno realizzato alcuni strumenti di ammodernamento.
Accanto alla stampa tradizionale, i programmi inviavano sms o email e gestivano in alcuni casi anche le accettazioni sempre con sms o email, andando anche a modificare i

[16]Clipper è un versatile linguaggio di programmazione di alto livello. È stato ed è usato prevalentemente per creare programmi gestionali integrati basati su database relazionali in ambiente DOS.

database, che erano realizzati con file locali in *DB3*[17] (file con estensione DBF).
Sorgevano però alcuni problemi:

- Innanzitutto l'evoluzione dei sistemi operativi creava malfunzionamenti ed incompatibilità con il software realizzato in *Clipper* (che peraltro era puramente testuale e non prevedeva l'uso del mouse)
- I dati erano distribuiti e c'erano spesso discordanze tra i vari comitati, e le gare magari erano replicate in più sezioni quando sono delegate da una struttura ad un'altra.
- La stessa segreteria *AIA* aveva un suo ulteriore database, quindi i dati di uno stesso associato erano replicati in almeno tre database differenti (sezioni, comitato regionale, segreteria centrale).

Negli ultimi anni, si è molto parlato di un ammodernamento dei sistemi, dell'uso dei cellulari e quant'altro, le voci si sono fatte sempre più insistenti ma alla fine non si arrivava ad un prodotto finito per vari motivi. Dalla fine del 2009 si è iniziato a pensare con più forza a questo progetto, per andare incontro alle possibilità di connettività da qualunque postazione per aiutare i vari operatori nel loro lavoro con un database unico, centralizzato, con le informazioni sempre aggiornate per tutti. Da qui nasce il sistema aggiornato *Sinfonia 4 You*.
Attualmente il sistema *Sinfonia 4 You* viene caricato su server *Microsoft* con *Sql Server* 2008.

[17] È un software per la gestione di database, archivi dati, tabelle, report in ambiente *MS-DOS* e *Windows*.

1.5 Portale Web "Sinfonia 4 You"

Il portale *Web Sinfonia 4 You* è attualmente utilizzato a livello nazionale e adottato in tutte se sezioni italiane. Per accedere al portale si fa riferimento al link (https://servizi.aia-figc.it/sinfonia4you/) che da l'accesso al portale, portando l'utente alla prima schermata. La funzione di *Login* (Figura 1.2) , è il primo step per accedere alla pagina personale dell'associato.
I campi da riempire sono:

- *Codice Meccanografico*
- *Password*

Il *Codice Meccanografico* è il numero identificativo assegnato ad ogni associato al momento del tesseramento, mentre la password viene assegnata dagli organi competenti e successivamente inviata all'associato via mail.
Dopo l'inserimento delle credenziali l'associato ha la possibilità di scegliere, tramite il menù a tendina (Figura 1.3), fra due opzioni di memorizzazione (cioè solo per la sessione che sta utilizzando) o scegliere di farle rimanere in memoria per trenta giorni. Infine si clicca su "*ACCEDI AL SISTEMA*" per entrare nella schermata successiva (Figura 1.4). Effettuato il login si accede alla schermata del portale *Web*.

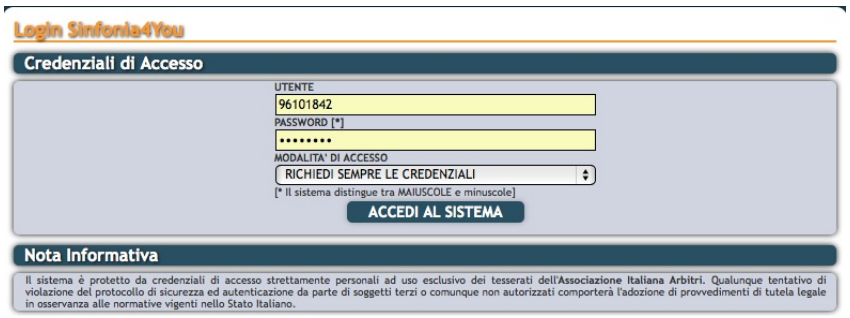


Figura 1.2: Login 1

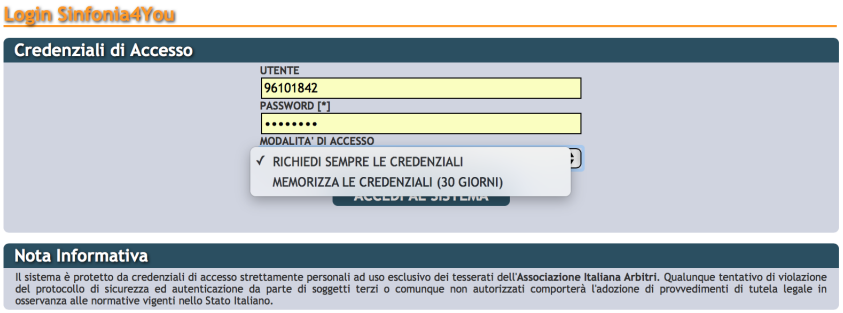


Figura 1.3: Login 2

Si presentano tre macro sezioni:

- *Dati Personali*
  - *Anagrafica Personale*: In quest'area vengono inseriti tutti i dati personali anagrafici dell'associato.
  - *Account Sistema*: in quest'area si possono modificare i dati di sicurezza come la password.
- *Area Tecnica*
  - *Gestione Gare*: È l'area che racchiude l'elenco delle gare, le gare ancora da accettare, scheda tecnica di ogni gara.
    - *Accettazione Gare*
    - *Scheda Tecnica*
  - *Certificato Medico*
  - *Indisponibilità*
  - *Congedi*
  - *Preclusioni*
  - *Gestione Domande*
  - *Gestione Eventi*
  - *Gestione Comunicazioni*
- *Area Assistenza*
  - *Assistenza Tecnica*
  - *Manuale Utente*

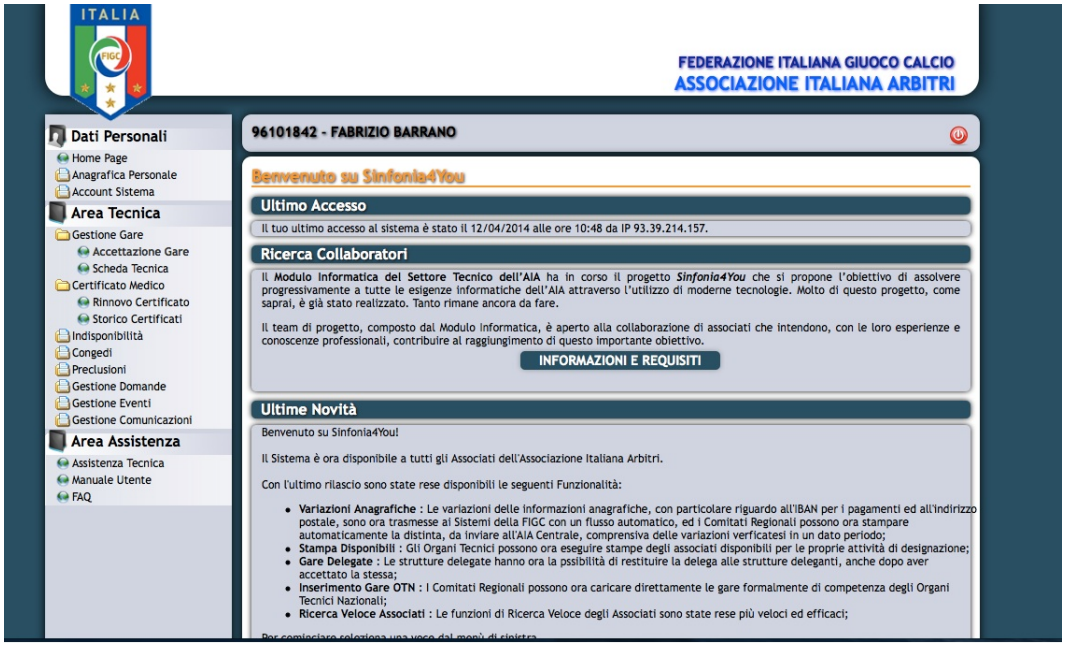


Figura 1.4: Portale web dopo l'avvenuto Login

– *FAQ*

Entrando nella sezione "*Gestione Gare*" e quindi in "*Accettazione Gare*", si accede alla finestra che racchiude l'elenco delle gare effettuate e, nel caso in cui vi siano partite da accettare, è presente l'azione che consente di effettuare dell'accettazione.

La tabella delle gare (Figura 1.5) è così composta:

- Info: dove interagendo con il tasto "i" di informazioni, si aprirà una finestra con i dettagli della partita.
- Cat: che descrive la categoria alla quale si stati designati.
- Data: la data di svolgimento della gara.
- Ora: ora della gara.
- Squadra Ospitante: descrizione della squadra che gioca in casa e che quindi ospita la gara.
- Squadra Ospite: descrizione della squadra che gioca in trasferta.

- *Attività*: descrizione dell'attività che è tenuto a svolgere l'associato.
  Le attività dell'associato possono essere suddivise in 6 tipologie:
  - *Arbitro Effettivo*, di solito abbreviato *A.E.* : la qualifica di arbitro effettivo si acquisisce dopo un esame al termine del corso arbitri, al quale possono fare domanda di ammissione coloro che possiedano determinati requisiti: ad esempio bisogna aver compiuto il 15 anno di età e non superato il 35 e bisogna essere in possesso di un titolo di scuola media inferiore. Con l'acquisizione della qualifica gli *A.E.* vengono inseriti nell'organico dell'*O.T.S.*[18], con la possibilità di essere proposti al *C.R.A.* dal presidente di sezione in base al merito, nei tempi ritenuti opportuni. La qualifica abilita alla direzione di gare di calcio a cinque.
  - *Arbitro Effettivo per il Calcio a Cinque*, di solito abbreviato *A.E. 5* : la qualifica di arbitro effettivo per il calcio a cinque è prerogativa del *C.R.A.* e si ottiene dopo aver svolto tale attività a disposizione dell'*O.T.S.*. per almeno due anni. Nel caso non vi siano gare di calcio a cinque di competenza dell'*O.T.S.*, l'arbitro effettivo dovrà sostenere un corso di qualificazione e avere tre anni di anzianità arbitrale.
  - *Assistente Arbitrale*, di solito abbreviato *A.A.*: la qualifica di assistente arbitrale si ottiene a domanda, dopo tre anni di anzianità arbitrale e previo il superamento di un corso di qualificazione. Con l'acquisizione della qualifica, si cessa l'attività arbitrale e si viene inserito come assistente nell'organico del *C.R.A.*, e si diventa proponibili per il passaggio alle categorie superiori con tale qualifica.
  - *Arbitro Fuori Quadro*, di solito abbreviato *A.F.Q.*; soppressa dal nuovo regolamento associativo 2012/2013.

[18] Organo Tecnico Sezionale

– *Arbitro Benemerito* : La qualifica di arbitro benemerito si può acquisire in vari modi, con il prerequisito di non aver subito sanzioni disciplinari nelle ultime due stagioni sportive e non avere procedimenti in corso:
automaticamente dopo essere stato arbitro o assistente internazionale; automaticamente dopo essere stato a disposizione della *C.A.N. A*[19] o della ex *C.A.N. A-B* dirigendo almeno venti gare in Serie A e superando l'esame di idoneità all'attività di osservatore arbitrale; automaticamente dopo cinquant'anni di tessera.
Inoltre ogni due anni il comitato nazionale proclama arbitri benemeriti secondo una graduatoria e in un numero prefissato coloro che:
abbiano svolto attività tecnica ed abbiano superato le prove di qualificazione alla funzione d'osservatore arbitrale;
abbiano assolto incarichi direttivi associativi, d'elezione o di nomina, anche in ambito sezionale; abbiano maturato 20 anni di anzianità arbitrale;
non siano incorsi in sanzioni disciplinari durante le ultime due stagioni sportive e non abbiano alcun procedimento disciplinare in corso.

– *Dirigenti Benemeriti* : possono essere proposti per la nomina di dirigenti benemeriti *FIGC* associati *AIA* gli ex presidenti nazionali dell'*AIA* non più in carica, nonché gli associati che abbiano svolto una prestigiosa e qualificata attività dirigenziale nell'ambito associativo e/o federale e con almeno trentacinque anni d'anzianità arbitrale.
I dirigenti benemeriti *AIA*, invece sono gli ex presidenti dell'*AIA* e coloro che abbiano svolto una prestigiosa e qualificata attività dirigenziale tecnica e/o associativa in ambito *AIA*, abbiano maturato un'anzianità associativa superiore a trentacinque anni e siano arbitri benemeriti da almeno dodici stagioni sportive.

- Stato: descrizione dello stato nel quale si trova la gara. Nel caso in cui l'associato debba ancora accettare la gara si presenterà vuoto. Nel caso di avvenuta accettazione si avrà un pallino verde che confermerà l'esito positivo di avvenuta accettazione.

- Azioni: descrizione dell'azione che si può svolgere. Si potrà interagire soltanto nel caso in cui la gara debba essere ancora accettata.

Interagendo con il tasto verde della colonna "Azioni" appare una finestra (Figura 1.6) che ha un messaggio di attenzione, dando la possibilità all'associato di confermare la gara alla quale si è stati designati, con il tasto ok.

[19]Commissione Arbitri Nazionale Serie A

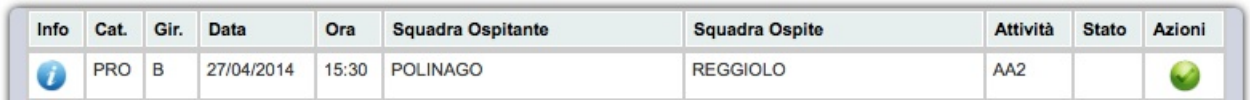


Figura 1.5: Gara appena designata, da accettare

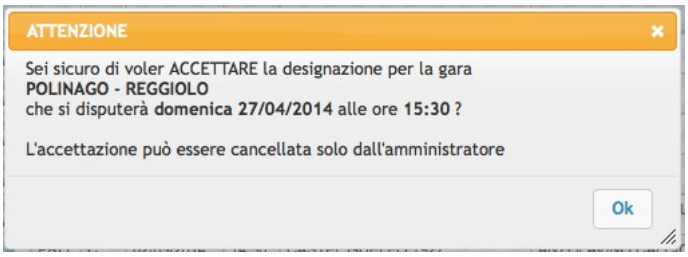


Figura 1.6: Avviso di accettazione

La finestra successiva (Figura 1.7) descrive l'avvenuto successo dell'accettazione della gara.

Figura 1.7: Avviso di avvenuta accettazione

Ritornando nella schermata principale (Figura 1.8) della "Gestione Partite" si ha la situazione aggiornata delle gare disputate.

1.6 Obiettivi della Tesi

La tesi ha l'obiettivo di semplificare la visione delle gare designate, creando un servizio sviluppato in un'App che consenta la visualizzazione di una lista partite, il dettagli delle rispettive, la visualizzazione dei collaboratori e il calcolo del percorso all'impianto sportivo per il quale si è stati designati.

1.6.1 Problema

L'idea è nata per semplificare l'attività degli arbitri riguardo la gestione e la consultazione della gara alla quale si è stati designati. L'assenza di un'App che consentisse la

Info	Cat.	Gir.	Data	Ora	Squadra Ospitante	Squadra Ospite	Attività	Stato	Azioni
	PRO	B	27/04/2014	15:30	POLINAGO	REGGIOLO	AA2		
	PRO	D	13/04/2014	15:30	SPORTING CLUB VALLESAVIO	CLASSE	AA1		
	PRO	A	06/04/2014	15:30	CASTELNOVESE A.D.	TRAVERSETOLO	AA2		
	PRO	A	16/03/2014	14:30	TERME MONTICELLI	BASILICA 2000	AA1		
	PRO	D	09/03/2014	14:30	TORCONCA	SPORTING CLUB VALLESAVIO	AA1		
	PRO	C	02/03/2014	14:30	CASTEL GUELFO 1927	ANZOLAVINO CALCIO	AA1		

Figura 1.8: Situazione gare aggiornate dopo l'ultima accettazione

visualizzazione delle gare da disputare o già effettuate, ha acquisito, con passare del tempo, sempre maggiore importanza e ha spinto lo studente alla realizzazione della suddetta App.

1.6.2 Soluzione

Per rendere disponibile questo servizio si deve far riferimento al portale attualmente in uso dell'associazione.
Dal portale web "*Sinfonia 4 You*" si è voluto rendere disponibile all'associato[20] la possibilità di interagire con la sezione "*Gestione Gare*". Una volta ricevuta la mail di avvenuta designazione, si è preso in considerazione la possibilità di consultare la gara con tutte le specifiche, tramite device, creando un'App che permettesse all'utente di evitare l'accesso al portale, ogni qualvolta se ne avesse di bisogno, nel browser , eseguire la funzione di login e successivamente prendere visione della gara.
La soluzione è, quindi, quella di evitare tutti questi passaggi rendendo, di conseguenza, più semplice ed immediato l'accesso al servizio.

[20] Associato: È l'arbitro, assistente o l'osservatore

Capitolo 2

Analisi degli aspetti tecnologici

2.1 Xcode

Xcode [7] è un ambiente di sviluppo integrato **IDE***(Integrated development environment)* sviluppato da *Apple Inc.* per agevolare lo sviluppo di applicazioni per *Mac OS X* e *iOS*. Supporta la compilazione incrementale ed è in grado di compilare il codice mentre viene scritto, in modo da ridurre il tempo di compilazione.

2.1.1 Struttura Xcode

L'interfaccia dell'ambiente di sviluppo di Xcode ed è così formata:

- **Barra degli strumenti**
  La barra degli strumenti, nella parte superiore della finestra, include i controlli di base per la costruzione di un progetto e per l'elaborazione delle diverse sezioni dell'area di lavoro. A sinistra sono riportati i pulsanti *Run* e *Stop*, che avviano e interrompono la simulazione dell'App.
  A fianco a questi si trova il selettore *Set the Active Sheme*, per decidere con quale tipologia di device deve avviarsi il simulatore. Infine, il pulsante *Organaizer* apre la finestra omonima, che riguarda elementi relativi a più progetti, per esempio la gestione dei device, la documentazione e i repository di controllo dei sorgenti. Si può nascondere la barra degli strumenti utilizzando il menu, ma si suggerisce di averla sempre a disposizione perché è particolarmente utile.

- **Area Navigator**:
  Il pannello/colonna di sinistra, che può essere visualizzato o nascosto dal pulsante *Show Navigator*, permette di esaminare il contenuto del progetto. In questa colonna si può selezionare, tramite una barra orizzontale, la sezione alla quale si vuole accedere. Le sezioni sono:

  - *Project*, che mostra i file sorgente e i file risorse del progetto ed è pertanto quello più importante e maggiormente impiegato.
  - *Symbol Navigator*, dove vengono elencati tutte le tipologie di metodi utilizzate nel progetto.
  - *Find Navigator*, permette di fare una ricerca di file in tutto il progetto.
  - *Issue Navigator*, sezione molto importante per la fase di testing. Qui vengono riportati tutti gli errori in fase di compilazione o simulazione.
  - *Test Navigator*, per avviare il test dell'App.
  - *Debug Navigator*, in questa sezione di testing viene mostrato il peso che ha l'App nel simulatore. Vengono prese in considerazione: l'utilizzo della *CPU*, la *RAM* utilizzata, il peso dell'App sull'Hardisk e sul Web.

- **Area Editor**:
  La parte principale dell'area di lavoro è costituita dall'Editor, l'unica vista che non può essere nascosta. Il suo contenuto dipende dal file selezionato nell'area *Navigator*, mentre la forma assunta dalla finestra dipende dal tipo di file che si deve modificare. Si supponga per esempio di selezionare un file sorgente: in questo caso si visualizza un tipico editor di codice sorgente. La selezione di un file *GUI* comporta invece che l'area diventi un editor *GUI* visuale.
  Il pulsante *Editor* della barra degli strumenti consente di selezionare tre modalità di visualizzazione del pannello di editing:

  - *Standard*, che corrisponde all'editor predefinito per il tipo di file selezionato;
  - *Assistant*, che mostra i file collegati fianco a fianco;
  - *Version*, che utilizza il controllo sorgente per mostrare le versioni corrente e storiche del file, una a fianco dell'altra. Questa terza modalità consente di individuare il "responsabile" di ogni riga di codice, oppure riportare una serie di commenti inseriti nel codice.

L'area *Editor* contiene inoltre una barra di accesso rapito in stile *Breadcrumb*[1] che mostra la gerarchia dell'oggetto che si sta elaborando; per un file sorgente, la barra mostra per esempio il percorso "progetto, gruppo, file, metodo". Ogni elemento della barra corrisponde a un menu a comparsa che permette di navigare altri punti di interesse collegati o elaborati di recente.

- **Area Utility**:
  La selezione a destra dell'area di lavoro del progetto contiene utility che forniscono viste dettagliate e modificano determinate soluzione dell'area *Editor*. La barra degli strumenti nella parte superiore dipende dal file da modificare e riporta strumenti differenti nel pannello File *Inspector*. L'area mostra sempre informazioni di base relative al file selezionato, cui si aggiunge un aiuto rapido legato alla selezione corrente. Per i file *GUI*[2] sono disponibili inspector che elaborano singole entità d i classi degli oggetti *UI*, i rispettivi attribuiti da configurare, la loro dimensione, il loro layout e i collegamenti al codice sorgente. Nella parte inferiore dell'area *Utility* si trova il pannello *Library*, che consente di accedere con facilità(clic e trascina) a *Snippet*[3] di codice, a oggetti dell'interfaccia utente e altro ancora.

- **Area Debug**:
  In fondo alla finestra, sotto l'area *Editor* e tra le aree *Navigator* e *Utility*, è presente un pannello che visualizza le informazioni del *Debug* in fase di esecuzione di un'App. La sua piccola barra degli strumenti include un menu a tendina per passare da una vistar di debug che ispezione la memoria quando si incontra un *Breakpoint*, a una vista che mostra l'output prodotto dall'applicazione mobile oppure a una vista che presenta entrambe le opzioni.

Infine il grosso pulsante *Run* consente l'avvio dell'applicazione. Verificare l'opzione *Sheme* sia *iPhone 6.0 Simulator* e poi fare clic su *Run*. L'area di stato mostra una barra di progresso che si riempie durante il build dei file e il bundle dell'applicazione; al termine dell'operazione si avvia Simulatore *iOS*. Il simulatore è un'altra applicazione *OS*, che ha l'aspetto e si comporta come un *iPhone* o come un *iPad*.
L'esecuzione dell'applicazione in Simulatore *iOS* implica la scomparsa della finestra principale, che viene sostituita da un box grigio riempito della schermata della simulazione.

[1] Le *Breadcrumb* sono una tecnica di navigazione usata nelle interfacce utente. Il loro scopo è quello di fornire agli utenti un modo di tener traccia della loro posizione in documenti o programmi.

[2] GUI è un tipo di interfaccia utente che consente all'utente di interagire con la macchina controllando oggetti grafici convenzionali.

[3] Gli Snippet sono frammenti ed esempi di codice sorgente

2.2 Objective-C

Caratteristiche del linguaggio
Objective-C [2] è un linguaggio di programmazione che ha le stesse caratteristiche di altri linguaggi come *Java* o *C#*. Cioè un completo supporto agli oggetti e all'ereditarietà, la gestione automatica o semiautomatica della memoria, le eccezioni, i blocchi, multithreading e altre funzionalità tipiche dei linguaggi orientati agli oggetti.
Objective-C introduce la chiocciola "@" prendendo il significato di *direttiva*, che contraddistingue in modo esplicito gli elementi di sintassi specifici di questo linguaggio.
Partendo dalla base del linguaggio *C*, *Objective-C* non fa altro che aggiungere la possibilità di programmare ad oggetti. Il linguaggio supporta l'ereditarietà singola, allo stesso modo di *C#* e *Java*. Come con questi linguaggi, è possibile ottenere alcuni dei vantaggi dell'ereditarieà multipla con sistemi alternativi.
Objective-C è un linguaggio di tipo dinamico, infatti in fase di esecuzione si può tranquillamente prendere la decisione sul metodo da invocare e sul tipo di un oggetto e le sue caratteristiche.

Sviluppo con Objective-C
Objective-C è utilizzabile in qualsiasi piattaforma dato che questo linguaggio è supportato da *GCC* (GNU C Compiler), compilatore che è presente nelle piattaforme più note (*Windows*, *Linux* e *Mac OS X*).
Xcode, un insieme di strumenti che include un *IDE*, strumenti di analisi delle prestazioni, un simulatore *iPhone/iPad* e librerie di sistema sotto forma di *SDK* per *Mac* e *iPhone*, sono necessarie per sviluppare con *Objective-C*. È necessario esser forniti di un *Mac* di nuova generazione con processore *Intel*.

2.3 Struttura App

Nella memoria dell'*iPhone* [**?**], quando viene avviata un'App vengono caricate in ordine:

- Caricamento App
- Caricamento Storyboard
- Caricamento UIViewController
- Caricamento UIView

2.3.1 AppDelegate

L'idea di base dell'applicazione è che un oggetto può avere un singolo delegato che richiamerà quando si verificano determinati eventi.
AppDelegate consiste nel:

- A: Assegnare il delegato all'interfaccia della classe.
- B: Assegnare il delegato all'oggetto che ne fa uso.
- C: Scrivere il metodo del delegato interessato.

I processi dell'*AppDelegate* sono:

- L'App sta per essere disattivata.
- L'App sta per andare in background.
- L'App sta per tornare in primo piano.
- L'App sta per tornare attiva.
- L'App sta per essere terminata.

2.3.2 Cocoa Touch

Framework per lo sviluppo di applicazioni destinate a touchscreen.
I *framework* principali per la programmazione *Objective-c* sono:

- **Foundation**; contiene classi dati essenziali, utility di base e stabilisce alcune convenzioni di programmazione core che non possono essere espresse dal linguaggio *Objective-C* da solo, come le tecniche per la gestione della memoria. Praticamente tutte le classi sono eredi di una classe root, *NSObject*, definita in *Foundation*.

La parte "touch" di *Cocoa Touch* è largamente rappresentata dal *Framework* **UIKit**, anch'esso importato per default in qualsiasi applicazione *iPhone*. Questo *Framework* offre modello di disegno, gestione di eventi, ciclo di vita delle applicazioni e altri elementi essenziali per un'applicazione destinata a un dispositivo con un touchscreen.

2.3.3 Gestione della memoria dell'applicazione

Objective-C utilizza un sistema di conteggio di riferimenti per la gestire la memoria. Ogni oggetto è associato a un *reference count*. L'allocazione imposta il conteggio a 1. Ogni altro oggetto che utilizza l'oggetto e vuole che non venga rilasciato dalla memoria può chiama *retain* per aumentare il conteggio. Si può inoltre copiare (*Copy*) un oggetto per ottenerne uno nuovo, copiato dall'originale, che abbia un conteggio pari a 1. Terminate le operazioni su un oggetto si pu'o chiamare *Release* per diminuire il conteggio di 1; quando si arriva a 0, l'oggetto viene liberato dalla memoria.

2.3.4 Automatic Reference Count: ARC

Nella fase di costruzione dell'App è stata introdotto, negli ultimi aggiornamenti di *Xcode*, una nuova caratteristica del compilatore: l'*ARC*. Esso non fa altro che aggiungere in maniera automatica agli oggetti le istruzioni di ritenzione e rilascio, togliendo così allo sviluppatore questa responsabilità. La regola è che finché esiste una variabile che punta a un oggetto, quest'ultimo resterà "in vita", quando il puntatore all'oggetto assume un nuovo valore o cessa di esistere, l'oggetto assuma un nuovo valore o cessa di esistere, l'oggetto associato sarà rilasciato e libererà la memoria occupata. Non sempre questo automatismo funziona al cento per cento. Ci sono casi in cui bisogna liberare la memoria manualmente o far puntare le proprie variabili all'oggetto *Nil* per poterla liberare. Se *ARC* è abilitato non ci saranno errori neanche in fase di esecuzione. Se invece l'oggetto su cui stiamo lavorando non è valido diventa responsabilità del programmatore sapere in ogni momento quale oggetto è valido e chi punta ad esso.

2.3.5 Storyboard

Una nuova caratteristica introdotta in *Xcode* è la *Storyboard* [11]. Essa rappresenta graficamente la sequenza delle schermate della nostra applicazione. Se prima si doveva "saltare" da un *".xib"*[4] all'altro per seguire delle modifiche o creare nuove *View*[5], adesso viene tutto semplificato con la *Storyboard* e anche parte del codice che si occupa del caricamento e delle visualizzazione delle varie view che compongono l'applicazione mobile. Questa caratteristica è risultata molto comoda per tutti gli sviluppatori perchè da la possibilità, nel caso in cui un progetto presenti tante *View*, di gestirle tutte in una sola videata. Le *Storyboard* di *iPhone* ed *iPad* naturalmente differiscono tra di loro in quanto le dimensioni sono diverse.

[4]Il file *.xib* è stato introdotto da *Apple Inc.* per dare la possibilità all'utente di creare delle viste in *Xcode*

[5]Le *View* sono le viste rappresentate nelle App.

Capitolo 3

Progettazione App

3.1 Sinfonia 4 You vs. Sinfonia 4 You Mobile

In questo paragrafo si evidenzia, in maniera più specifica, cosa viene preso sviluppato dal portale *Web Sinfonia 4 You*. L'argomento principale preso in considerazione riguarda la sezione *Gestione Gare*. In questa sezione appare ad ogni associato la gara per la quale si è stati designati. La sezione in questione è formata da due sotto sezioni che si chiamano *Accettazione Gare* e *Scheda Tecnica*. Nel progetto viene sviluppata un'App che racchiude entrambe le sezioni. Verranno prese in considerazione le partite inviate come designazione ed inserite in una *View* che racchiuderà lo storico delle partite.
Nella colonna di sinistra del portale Web si presenta con le seguenti sezioni:

- Gestione Gare
- Certificato Medico
- Indisponibilità
- Congedi
- Preclusioni
- Gestione Domande
- Gestione Eventi
- Gestione Comunicazioni

3.2 Mockup

Pensando ad un design per lo sviluppo dell'App si è preso in considerazione di realizzare quattro *View*. Una è la funzione di login, la seconda rappresenta la lista delle partite, cliccando nella gara scelta nella lista si accede alla terza *View* nella quale viene visualizzato il dettaglio della partita ed anche un bottone che rappresenta l'accettazione della gara, la quarta *View* ritrae il dettaglio dei collaboratori designati per la gara.

Di seguito si dimostra in modo concettuale e intuitivo come si accede all'App:

- Nella Figura 3.1 viene rappresentata la prima *View* dove viene effettuato il login per accedere al sistema. Inserendo le credenziali, cioè il codice meccanografico e la password, si accede alla lista delle partite.

- La Figura 3.2 rappresenta la visualizzazione della lista delle partite, questa raccoglie l'elenco di tutte le partite arbitrate.
  Alla fine della lista delle partite si trova un bottone di logout che permette di tornare alla *View* della funzione di login. Cliccando sulla gara desiderata si accede alla *View* del dettaglio delle partita partita corrispondente.

- Nella Figura 3.3 viene rappresentato il dettaglio della gara. In particolare le squadra che sono state designate all'associato, il numero identificativo della gara, il ruolo per il quale l'associato è stato designato a svolgere, la categoria del campionato (ad esempio Promozione o Eccellenza) appartenente, il nome dell'impianto dove andare a svolgere la gara e l'indirizzo, l'ora, la città, i km da percorrere ed il rimborso spese. In fine si trova un bottone che consente l'accesso alla *View* successiva dei collaboratori.

- Nella Figura 3.4 viene rappresentata la *View* del calcolo del percorso tramite l'applicazione nativa dell'iPhone.

- Nella Figura 3.5 vengono rappresentati i dettagli degli eventuali collaboratori come nome, cognome, ruolo, mail e numero di telefono.

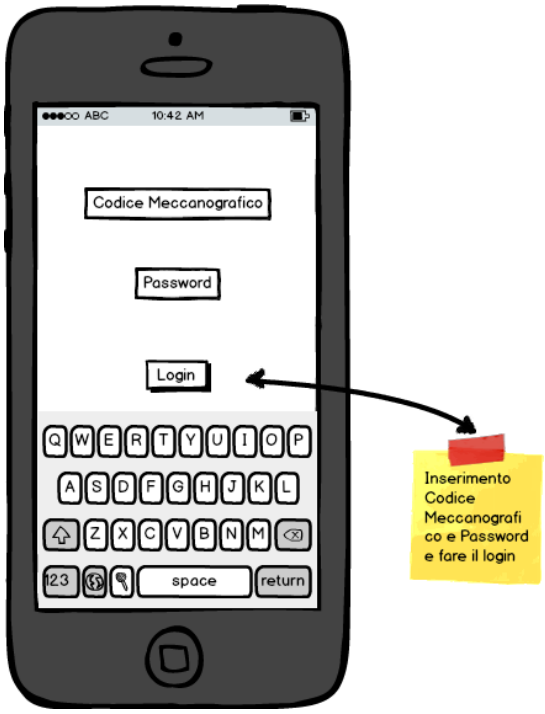


Figura 3.1: Login

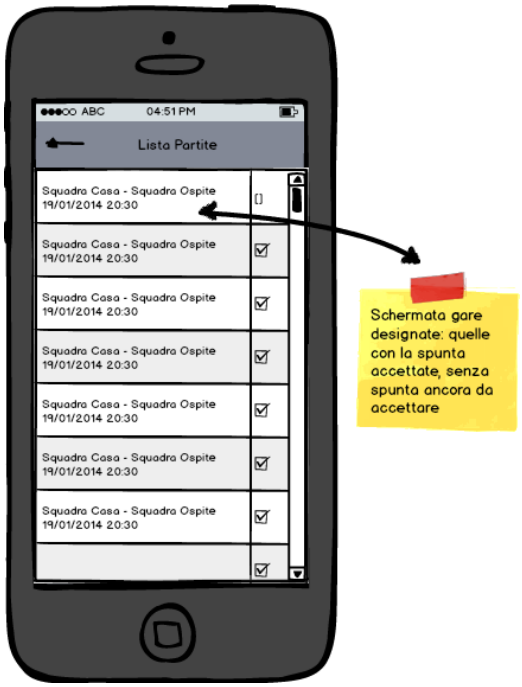


Figura 3.2: Lista Partite

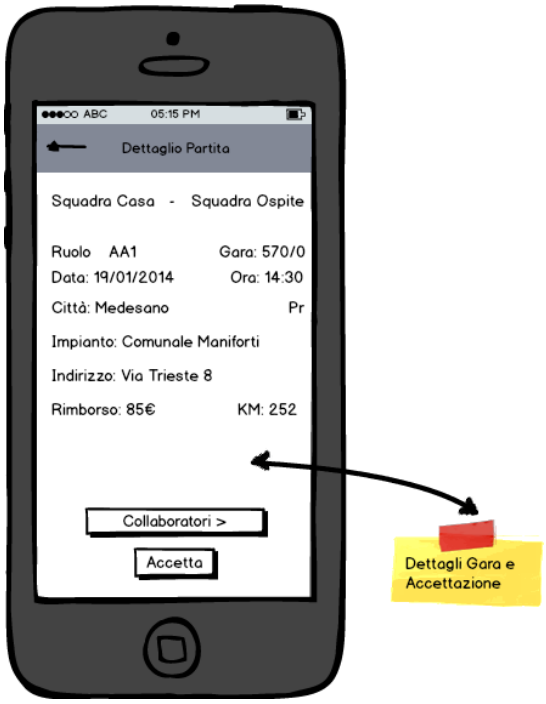


Figura 3.3: Dettaglio Partita

Figura 3.4: Calcolo Percorso

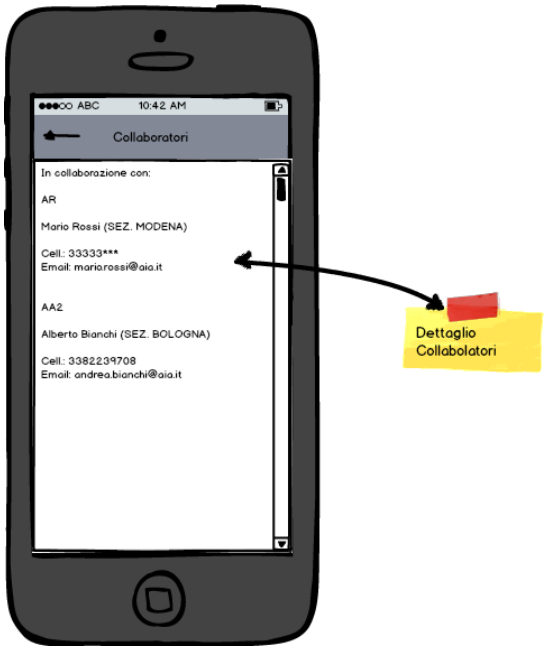


Figura 3.5: Dettaglio Collaboratori

3.3 Balsamiq Mockups

Balsami Mockups[1] è un software per sviluppatori, designer, progettisti ed altri creativi, progettato per creare *Wireframe*[2], ovvero design di interfacce utenti. Usandolo si possono facilmente creare bozze di schermate per applicazioni di tutti i tipi: siti *Web*, applicazioni *Web*, Deskop e Mobile.
Prendendo come esempio questa tesi di progetto è stato sfruttato *Balsamiq* per creare una rappresentazione di come si poteva immaginare come l'App venisse sviluppata e cosa si potesse vedere attraverso le *View*.

[1] `http://balsamiq.com`
[2] Indica un tipo di rappresentazione grafica da computer di oggetti tridimensionali.

3.4 Architettura App

3.4.1 Model View Controller

Il *Model-View-Controller*, è un *Pattern* architetturale[3] molto diffuso nello sviluppo di sistemi software, in particolare nell'ambito della programmazione orientata agli oggetti, in grado di separare la logica di presentazione dei dati dalla logica di business.
Questo pattern si posiziona nel livello di presentazione in una Architettura *Multi-tier*[4].
In questo progetto il *MVC* è così formato:
I Manager eseguono una richiesta al server di tipo *REST/HTTP*. Quest'ultimo da come risposta un *JSON*. Il mapping del *JSON* viene effettuato sempre dal Manager il quale mappa il *JSON* nel Model. La rappresentazione del *JSON* in *Objective-C* implementata nei Model, cioè nel progetto *Sinfonia 4 You*.
Infine il View Controller valorizza le *Label* con i Model restituiti da Manager.

3.4.2 Struttura dei Model

I Model implementati sono quattro. Nel *".h"* e nel *".m"* saranno dichiarati rispettivamente le *Property* ed i *Synthesize*[5] per ogni campo presente nel *JSON*. I quattro Model vengono di seguito elencati:

- Lista Partita Model
- Dettaglio Partita Model
- Collaboratori Model
- Contatto Model

[3]Il *MVC* esprime schemi di base per impostare l'organizzazione strutturale di un sistema software. In questi schemi si descrivono sottosistemi predefiniti insieme con i ruoli che essi assumono e le relazioni reciproche.

[4]Indica un'architettura software in cui le varie funzionalità del software sono logicamente separate ovvero suddivise su più strati o livelli software differenti in comunicazione tra loro (nel caso di applicazioni web questi strati sono la logica di presentazione, l'elaborazione dei processi e la gestione della persistenza dei dati)

[5]La scrittura dei metodi set e get viene gestita in automatico dai costrutti *@property* e *@synthesize*.

3.4.3 Struttura delle View

Le *View* sviluppate in quest'App sono quattro, così descritte ed elencate:

- Login View Controller
  La struttura della *LoginViewController* è composta da tre *Image View*[6], due *Text Field*[7] ed un *Button*. La prima *Image View* costituisce il titolo dell'App denominata *"Sinfonia 4 You"*, la seconda invece, rappresenta il logo dell'*Associazione Italiana Arbitri* ed infine un'altra *Image View* che fa da background alla *View* intera. Le due *Text Field* invece prendono il nome di *Codice Meccanografico* e di *Password*.

- Lista Partite View Controller
  La *View* della lista delle partite è caratterizzata da una struttura a celle. Ogni cella costituisce una partita che è composta da quattro Label. La *Label "squadraCasa"* specifica la squadra che viene inserita in quella *Label* ed è, appunto, la squadra che disputa la gara in casa, così come per la Label *"squadraOspite"* che rappresenta la squadra che gioca in trasferta. Altre due *Label*, *"dataLabel"* e *"oraLabel"*, caratterizzano rispettivamente la data e l'ora di svolgimento della gara.
  Altro elemento importante è l'unico *Button* della *View* che è rappresentato dal *"Logout"*, e permette all'utente di ritornare alla schermata iniziale di login.
  Infine la cella è caratterizzata da una *Image View* che riprende il colore dell'erba dei campi da calcio, e fa da background all'intera cella *View*.

[6] L'*Image View* serve a visualizzare una singola immagine o un'animazione descritta da una serie di immagini

[7] La *Text Field*Visualizza un rettangolo che può contenere del testo modificabile. Quando un utente tocca un campo di testo, viene visualizzata una tastiera; quando un utente tocca di nuovo nella tastiera, la tastiera scompare e il campo di testo può gestire l'input in modo specifico dell'applicazione. Il metodo *UITextField* supporta viste sovrapposte per visualizzare informazioni aggiuntive, come ad esempio l'icona di segnalibri. *UITextField* fornisce anche un controllo di testo in chiaro quando un utente tocca per cancellare il contenuto del campo di testo.

- Dettaglio Partita View Controller
  Il *DettaglioPartitaViewController* è strutturato da varie *Label* suddivise in due categorie: *Label Statiche e Dinamiche*[8].
  Le *Label* statiche sono:

  - Categoria
  - Ruolo
  - VS.
  - Gara
  - Map View Controller
  - Data
  - Ora
  - Città
  - Provincia
  - Rimborso
  - Km
  - Impianto
  - Indirizzo

  Mentre le *Label* dinamiche sono:

  - Squadra Casa
  - Squadra Ospite
  - Categoria
  - Attività
  - Attività Estesa
  - Numero Gara
  - Data Gara
  - Ora Gara
  - Provincia
  - Città
  - Impianto

[8]Le *Label* Statiche si definiscono tali perché sono delle etichette descrittive e fisse al contrario di quelle dinamiche che sono valorizzate con le informazioni frutto dell'iterazione dell'utente con l'App.

  - Indirizzo
  - Rimborso
  - Km Totali

  Oltre alle *Label* sono presenti tre *Button*. Il primo *"Collaboratori"* permette l'accesso alla *View* dei collaboratori, il secondo *"Accetta"* permette all'utente di accettare la gara ed infine il terzo *Button* (situato nella *Navigation Bar*) permette, una volta azionato, di avviare l'App nativa dell'iPhone *Mappe* e poter calcolare così il percorso stradale, attraverso l'indirizzo, dalla posizione di partenza nella quale si trova il device, fino all'impianto sportivo al quale si è stati designati.

- Collaboratori View Controller
  La struttura della *View CollaboratoriViewController* contiene anch'essa delle *Label* statiche e delle altre dinamiche, questa *View* vede la presenza di due Collaboratori. Le *Label* statiche sono così composte:

  - Sezione
  - Ruolo
  - Telefono
  - Mail

  distribuiti per due collaboratori, quindi le *Label* vengono ripetute due volte.
  Per quanto riguarda le *Label* dinamiche la composizione è la seguente:

  - Nome
  - Cognome
  - Sezione
  - Ruolo
  - Telefono
  - Mail

  anche queste sono ripetute due.

3.4.4 Struttura dei Manager

In questo progetto vengono implementati due Manager i quali saranno incaricati di fare le richieste al server.
I Manager presenti sono di seguito elencati:

- *ListaPartitaManager*
- *DettaglioPartitaManager*

3.5 Integrazione con il backend (server)

Lo scambio di informazioni tra l'App e il server avviene tramite scambio di *JSON*. Questi sono stati forniti dalla federazione all'inizio dello svolgimento della tesi. I *JSON* rappresentano la struttura dati che servono alla corretta implementazione delle interazioni delle gare.
Ai fini dell'implementazione del test dell'App stessa e della sua testabilità, si è simulato, tramite dei file *PHP* presenti su un sito ospite, il comando atteso.

Di seguito verranno elencati i file in questione:

```
http://< host>:<port>/login.php
http://<host>:<port>/listaPartite.php
http://<host>:<port>/dettaglioPartita.php
```

3.5.1 REST/HTTP

Representational State Transfer (REST) [14] è un tipo di architettura software per i sistemi di ipertesto distribuiti come il World Wide Web. I termini "representational state transfer" e "RES" furono introdotti nel 2000.
REST si riferisce ad un insieme di principi di architetture di rete, i quali delineano come le risorse sono definite e indirizzate. Il termine è spesso usato nel senso di descrivere ogni semplice interfaccia che trasmette dati su HTTP senza un livello opzionale come SOAP o la gestione della sessione tramite i cookie. Questi due concetti possono andare in conflitto così come in sovrapposizione.

3.5.2 JSON

JSON [13] , acronimo di *JavaScript Object Notation*, è un formato adatto per lo scambio dei dati in applicazioni client-server basato sul concetto di "Chiave"/"Valore". Nel progetto sono presenti tre rappresentazioni *JSON*. Il primo riguarda la funzione di *Login* che allo stato attuale è implementata semplicemente con un controllo all'inserimento del

Codice Meccanografico e della *Password*. Il secondo *JSON* prende il nome di *"Scheda tecnica"* e rappresenta la lista delle partite. Il terzo ed ultimo *JSON* contiene il dettaglio della gara, compreso il dettaglio dei collaboratori. Di seguito viene fatta vedere in dettaglio la struttura dei*JSON* appena elencati.

- Login Json

Il primo *JSON* ritorna dal server la risposta dell'avvenuto o non avvenuto successo della login, cioè se Codice Meccanografico e Password sono stati inseriti correttamente o meno. Di seguito viene elencato il *JSON* di risposta alla richiesta al server.


```
{"success":0,"error_message":"Codice meccanografico e/o password errati."}
```

- Lista Partite Json

Il secondo *JSON* rappresenta la lista delle partite e contiene l'elenco delle partite alla quale l'associato è stato designato:


```
{
 "result" : 0,
 "message" : "OK",
 "resultObject" : [{
   "attivitaDes" : "AR",
   "idGara" : 1001,
   "squadraCasa" : "LENTIGIONE CALCIO",
   "squadraOspite" : "SAMPOLESE",
   "dataGara" : "28/09/2014",
   "oraGara" : "15:30",
   "stato" : null,
   "categoria" : "ECC",
   "retiCasa" : -1,
   "retiFuori" : -1,
   "esito_des" : 2,
   "ingiustificato" : false,
   "idOmologazione" : -1,
   "idNonEff" : -1,
   "statodes" : "A",
   "idDesignazione" : 1083637
   }, {
   "attivitaDes" : "AR",
   "idGara" : 1002,
```

```
"squadraCasa" : "PIETRACUTA",
"squadraOspite" : "SPORTING CLUB VALLESAVIO",
"dataGara" : "05/10/2014",
"oraGara" : "15:30",
"stato" : null,
"categoria" : "PRO",
"retiCasa" : -1,
"retiFuori" : -1,
"esito_des" : 2,
"ingiustificato" : false,
"idOmologazione" : -1,
"idNonEff" : -1,
"statodes" : "A",
"idDesignazione" : 1083637
}, {
"attivitaDes" : "AA1",
"idGara" : 1003,
"squadraCasa" : "FONTANA AUDAX",
"squadraOspite" : "POLISPORTIVA BRESCELLO",
"dataGara" : "12/10/2014",
"oraGara" : "15:30",
"stato" : null,
"categoria" : "PRO",
"retiCasa" : 0,
"retiFuori" : 0,
"esito_des" : 2,
"ingiustificato" : false,
"idOmologazione" : 1,
"idNonEff" : 1,
"statodes" : "A",
"idDesignazione" : 1096138
}, {
"attivitaDes" : "AA1",
"idGara" : 1004,
"squadraCasa" : "SCANDIANESE CALCIO A.S.D",
"squadraOspite" : "SOLIERESE POLISPORTIVA",
"dataGara" : "19/10/2014",
"oraGara" : "15:30",
"stato" : null,
"categoria" : "PRO",
"retiCasa" : 0,
```

```
    "retiFuori" : 0,
    "esito_des" : 2,
    "ingiustificato" : false,
    "idOmologazione" : 1,
    "idNonEff" : 1,
    "statodes" : "A",
    "idDesignazione" :
    }
  ]
}
```

- Dettaglio Partita Json

Il terzo ed ultimo *JSON* rappresenta il dettaglio della partita incluso il nominativo dei collaboratori:


```
{
  "result" : 0,
  "message" : "ok",
  "resultObject" : {
    "idGara" : 1001,
    "attivita" : "AA1",
    "attivitaEstesa" : "Assistente dell\u0027arbitro n 2",
    "dataGara" : "28/09/2014",
    "categoriaRidotta" : "ECC",
    "categoria" : "ECCELLENZA",
    "girone" : "A",
    "giornata" : 5,
    "ar" : "A",
    "squadraCasa" : "LENTIGIONE CALCIO",
    "squadraOspite" : "SAMPOLESE",
    "stato" : null,
    "associate" : [{
        "idArbitro" : 5866,
        "attivitaSigla" : "AR",
        "attivitaEstesa" : "Arbitro",
        "cognome" : "ANDREA",
        "nome" : "NERI",
        "sezione" : "PIACENZA",
        "contatti" : [{
            "Tipo" : "C",
```

```
            "Valore" : "3391111111",
            "Note" : ""
          }, {
            "Tipo" : "M",
            "Valore" : "s4y.test.associati@gmail.com",
            "Note" : ""
          }
        ] ,
          "idArbitro" : 2651,
        "attivitaSigla" : "AA1",
        "attivitaEstesa" : "Assistente dell\u0027arbitro n 1",
        "cognome" : "CORMANO",
        "nome" : "RICCARDO",
        "sezione" : "IMOLA",
        "contatti" : [{
            "Tipo" : "C",
            "Valore" : "+3331111112",
            "Note" : ""
          }, {
            "Tipo" : "M",
            "Valore" : "s4y.test.associati@gmail.com",
            "Note" : ""
          },
        ]
      }
    ],
    "impianto" : "COMUNALE VOLANTE LEVANTINI",
    "provincia" : "RE",
    "citta" : "LENTIGIONE DI BRESCELLO",
    "cap" : 42041,
    "indirizzoGara" : "VIA BACCHI, 22",
    "nroGara" : 264,
    "oraGara" : "15:30",
    "risultato" : null,
    "kmTotali" : 198,
    "rimborso" : "69,00",
    "retiCasa" : -1,
    "retiFuori" : -1
  }
}
```

3.6 CocoaPods

CocoaPods[9] gestisce le dipendenze delle librerie dei progetti in *Xcode*. Le dipendenze dei progetti sono racchiuse in un unico file di testo chiamato *Podfile*. *CocoaPods* risolve le dipendenze tra le librerie presenti nei progetti, recupera il codice sorgente, quindi quindi collega insieme in un progetto nuovo di *Xcode* per iniziare a lavorare.
Lo scopo di *Cocoapods* è quello di migliorare ed avere a disposizione librerie di terze parti *open-source*.[9]
In questo progetto è stato utilizzato *CocoaPods* per gestire le dipendenze del framework *AFNetworking*.

[9]Open-source in informatica, indica un software i cui autori (più precisamente i detentori dei diritti) ne permettono e favoriscono il libero studio e l'apporto di modifiche da parte di altri programmatori indipendenti.

Capitolo 4

Implementazione

4.1 Storyboard

Lo *Storyboard* è composto da quattro *View*, nella Figura 4.1 si visualizza la composizione delle *View* nella storyboard ed il loro flusso.

Lo *Storyboard* è formato da 4 *View Controller* ed un *Navigation Controller*[1]:

- LoginViewController (Figura 4.3)
- ListaPartiteViewController (Figura 4.4)
- DettaglioPartitaViewController (Figura 4.5)
- CollaboratoriViewController (Figura 4.6)

[1]Il *Navigation Controller* gestisce una pila di *View*, ciascuna dei quali gestisce le successive *View*. Ad ogni *View* viene assegnato un titolo e un'azione di ritorno alla *View* precedente.

Figura 4.1: Storyboard

4.2 Gesture

Le *Gesture* [3] sono utilizzate per interagire con le App. Attraverso le gesture gli utenti possono interagire con gli oggetti presenti nelle view ed in generarle nelle App. Di seguito elencati i tipi di gesture:

- Tap: premere o selezionare un comando.
- Grag: trascinare un oggetto.
- Flick:
- Swipe: con un dito, trascinandolo da sinistra verso destra, si può tornare alla view precedente.
- Double Tap: toccare due volte lo schermo per effettuare lo zoom.
- Pinch: permette, tramite due dita, di fare lo zoom-in o lo zoom-out allargando le due dita

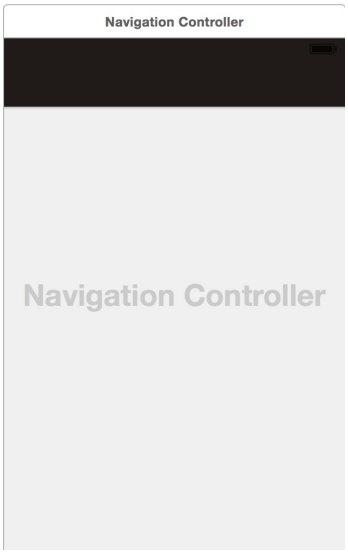


Figura 4.2: Navigation Controller

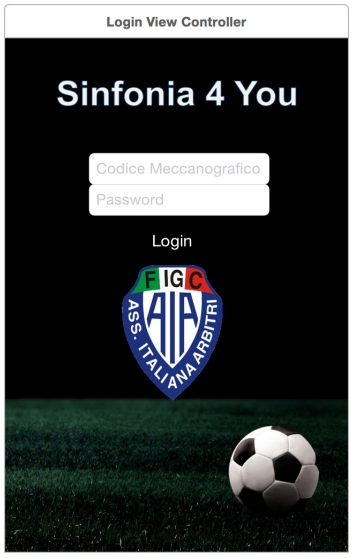


Figura 4.3: Login View Controller

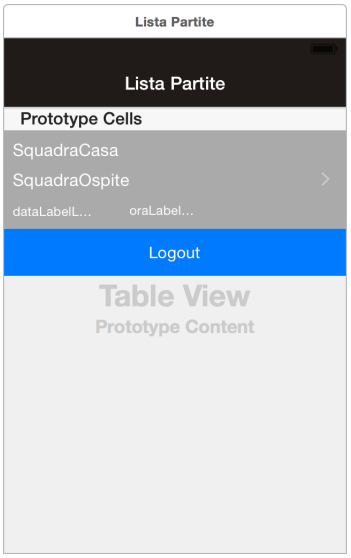


Figura 4.4: Lista Partite View Controller

Figura 4.5: Dettaglio Partita View Controller

Figura 4.6: Collaboratori View Controller

- Touch and Hold
- Shake: scuotendo l'iPhone si può avviare o annullare un'azione.

4.3 NotificationCenter

Un oggetto di tipo *NSNotificationCenter* [4] fornisce un meccanismo di "ascolto" di informazioni dentro un programma. Una lista di oggetti con un centro notifiche, riceve notifiche usando l'omonimo metodo. Ogni invocazione di questo metodo, specifica un set di notifiche. Concludendo, gli oggetti di tipo NSNotificationCenter possono registrarsi come osservatori di diversi set di notifica chiamando questi metodi più volte.

4.4 Singleton

Il *Singleton* [5] è un pattern tra i più importanti e rappresenta un tipo particolare di classe che garantisce che soltanto un'unica istanza della classe stessa può essere creata all'interno di un programma. Per ottenere questo comportamento è necessario avvalersi dello specificatore di accesso «private» anche per il costruttore della classe (cosa che generalmente non viene mai praticata in una classe "standard") ed utilizzare un metodo statico che consenta di accedere all'unica istanza della classe.
In generale, la scelta del singleton viene effettuata in tutti quei casi in cui è necessario che venga utilizzata una sola istanza di una classe.

Ciò consente di:

- Avere un accesso controllato all'unica istanza della classe
- Avere uno spazio di nomi ridotto
- Evitare la dichiarazione di variabili globali
- Assicurarsi di avere un basso numero di oggetti utilizzati in condivisione grazie al fatto che viene impedita la creazione di nuove istanze ogni volta che si voglia utilizzare la stessa classe.

4.5 Model

In questo paragrafo vengono listate le classi che rappresentano i modelli degli oggetti *Objective-C* e rappresentati dai *JSON* forniti dalla federazione.

4.5.1 ListaPartitaModel.h

Di seguito vengono elencate le *Property* ed il costruttore che serve per creare la partita nella cella della *View*.

```
#import <Foundation/Foundation.h>

@interface ListaPartitaModel : NSObject

@property (copy, nonatomic) NSString * attivitaDes;
@property (copy, nonatomic) NSString * idGara;
@property (copy, nonatomic) NSString * squadraCasa;
@property (copy, nonatomic) NSString * squadraOspite;
@property (copy, nonatomic) NSData * dataGara;
@property (copy, nonatomic) NSString * oraGara;
@property (copy, nonatomic) NSString * stato;
@property (copy, nonatomic) NSString * categoria;
@property (copy, nonatomic) NSString * retiCasa;
@property (copy, nonatomic) NSString * retiFuori;
@property (copy, nonatomic) NSString * esito_des;
@property (copy, nonatomic) NSString * ingiustificato;
@property (copy, nonatomic) NSString * idOmologazione;
@property (copy, nonatomic) NSString * idNonEff;
@property (copy, nonatomic) NSString * statodes;
```

```
@property (copy, nonatomic) NSString * idDesignazione;

+ (id)squadraCasa:(NSString*)squadraCasa
         squadraOspite:(NSString*)squadraOspite
            dataGara:(NSString*)dataGara oraGara:(NSString*)oraGara
                idGara:(NSString*)idGara;

@end
```

4.5.2 ListaPartitaModel.m

Di seguito vengono elencati i *Synthesize* ed implementato il costruttore.

```
#import "ListaPartitaModel.h"

@implementation ListaPartitaModel

@synthesize attivitaDes = _attivitaDes;
@synthesize idGara = _idGara;
@synthesize squadraCasa = _squadraCasa;
@synthesize squadraOspite = _squadraOspite;
@synthesize stato = _stato;
@synthesize dataGara = _dataGara;
@synthesize oraGara = _oraGara;
@synthesize categoria = _categoria;
@synthesize retiCasa = _retiCasa;
@synthesize retiFuori = _retiFuori;
@synthesize esito_des = _esito_des;
@synthesize ingiustificato = _ingiustificato;
@synthesize idOmologazione = _idOmologazione;
@synthesize idNonEff = _idNonEff;
@synthesize statodes = _statodes;
@synthesize idDesignazione = _idDesignazione;

+ (id)squadraCasa:(NSString*)squadraCasa
        squadraOspite:(NSString*)squadraOspite
          dataGara:(NSData *)dataGara
             oraGara:(NSString *)oraGara idGara:(NSString *)idGara {
```

```
    ListaPartitaModel * lpm = [[ListaPartitaModel alloc] init];

    lpm.squadraCasa = squadraCasa;
    lpm.squadraOspite = squadraOspite;
    lpm.dataGara = dataGara;
    lpm.oraGara = oraGara;
    lpm.idGara = idGara;

    return lpm;
}

@end
```

4.5.3 DettaglioPartitaModel.h

Il dettaglio delle partite ha la dichiarazione delle *property* per ogni campo dichiarato nella *View*, di seguito si trovano elencati:

```
#import <Foundation/Foundation.h>
#import "CollaboratoreModel.h"

@interface DettaglioPartitaModel : NSObject

@property (nonatomic, strong) NSString *idGara;
@property (nonatomic, strong) NSString *attivita;
@property (nonatomic, strong) NSString *attivitaEstesa;
@property (nonatomic, strong) NSString *data;
@property (nonatomic, strong) NSString *categoriaRidotta;
@property (nonatomic, strong) NSString *categoria;
@property (nonatomic, strong) NSString *girone;
@property (nonatomic, strong) NSString *giornata;
@property (nonatomic, strong) NSString *ar;
@property (nonatomic, strong) NSString *squadraCasa;
@property (nonatomic, strong) NSString *squadraOspite;
@property (nonatomic, strong) NSString *stato;
@property (nonatomic, strong) NSString *impianto;
@property (nonatomic, strong) NSString *provincia;
@property (nonatomic, strong) NSString *citta;
@property (nonatomic, assign) NSInteger *cap;
@property (nonatomic, strong) NSString *indirizzoGara;
```

```
@property (nonatomic, strong) NSString *nroGara;
@property (nonatomic, strong) NSString *ora;
@property (nonatomic, strong) NSString *risultato;
@property (nonatomic, strong) NSString *kmTotali;
@property (nonatomic, strong) NSString *rimborso;
@property (nonatomic, strong) NSString *retiCasa;
@property (nonatomic, strong) NSString *retiFuori;

@property (nonatomic, strong) NSArray *listaCollaboratori;

@end
```

4.5.4 DettaglioPartitaModel.m

In questa classe vengono dichiarate tutti i *Synthesize* di seguito elencate:

```
#import "DettaglioPartitaModel.h"
#import "DettaglioPartitaManager.h"
#import "CollaboratoreModel.h"

@implementation DettaglioPartitaModel

@synthesize idGara = _idGara;
@synthesize attivita = _attivita;
@synthesize attivitaEstesa =_attivitaEstesa;
@synthesize data = _data;
@synthesize categoriaRidotta = _categoriaRidotta;
@synthesize categoria = _categoria;
@synthesize girone = _girone;
@synthesize giornata = _giornata;
@synthesize ar = _ar;
@synthesize squadraCasa = _squadraCasa;
@synthesize squadraOspite = _squadraOspite;
@synthesize stato = _stato;
@synthesize impianto = _impianto;
@synthesize provincia = _provincia;
@synthesize citta = _citta;
@synthesize cap = _cap;
```

```
@synthesize indirizzoGara = _indirizzoGara;
@synthesize nroGara = _nroGara;
@synthesize ora = _ora;
@synthesize risultato = _risultato;
@synthesize kmTotali = _kmTotali;
@synthesize rimborso = _rimborso;
@synthesize retiCasa = _retiCasa;
@synthesize retiFuori = _retiFuori;

@end
```

4.5.5 CollaboratoreModel.h

La classe si presenta con tutte le *Property* dichiarate:

```
#import <Foundation/Foundation.h>
#import "ContattoModel.h"

@interface CollaboratoreModel : NSObject

@property (nonatomic, strong) NSString *idArbitro;
@property (nonatomic, strong) NSString *attivitaSigla;
@property (nonatomic, strong) NSString *attivitaEstesa;
@property (nonatomic, strong) NSString *nome;
@property (nonatomic, strong) NSString *cognome;
@property (nonatomic, strong) NSString *sezione;

@property (nonatomic, strong) NSArray *listaContatti;

@end
```

4.5.6 CollaboratoreModel.m

Di seguito elencate i *Synthesize*:

```
@implementation CollaboratoreModel

@synthesize idArbitro = _idArbitro;
```

```
@synthesize nome = _nome;
@synthesize cognome = _cognome;
@synthesize attivitaSigla = _attivitaSigla;
@synthesize attivitaEstesa = _attivitaEstesa;
@synthesize sezione = _sezione;

@end
```

4.5.7 ContattoModel.h

```
#import <Foundation/Foundation.h>

@interface ContattoModel : NSObject

@property (nonatomic, strong) NSString * tipo;
@property (nonatomic, strong) NSString * valore;
@property (nonatomic, strong) NSString * note;

@end
```

4.5.8 ContattoModel.m

```
#import "ContattoModel.h"

@implementation ContattoModel

@synthesize tipo = _tipo;
@synthesize valore = _valore;
@synthesize note = _note;

@end
```

4.6 Manager

In questo paragrafo vengono elencati i Manager utilizzati all'interno dell'App. I Manager hanno la funzionalità dei Controller del *MVC*.

4.6.1 ListaPartitaManager.h

Di seguito vengono riportati i metodi dichiarati nell'interfaccia e successivamente implementati nel .m dell'omonima classe.

```
#import <Foundation/Foundation.h>

#import <AFNetworking.h>

@class ListaPartitaModel;

@interface ListaPartitaManager : NSObject

+ (ListaPartitaManager *) sharedClass;

- (void)setup;

- (int)numeroDiPartite;

- (NSString*)getSquadraCasaPerRiga:(int)riga;

- (NSString*)getSquadraOspitePerRiga:(int)riga;

- (NSString*)getDataPerRiga:(int)riga;

- (NSString*)getOraPerRiga:(int)riga;

- (NSString*)getIdGara:(int)riga;

- (ListaPartitaManager*)dammiLaPartitaPerLaRiga:(int)riga;

@end
```

4.6.2 ListaPartitaManager.m

In questa classe vengono implementati i metodi precedentemente dichiarati nel *".h"*.

```
#import "ListaPartitaManager.h"
#import "ListaPartitaModel.h"

@interface ListaPartitaManager ()

@property (strong, nonatomic) NSArray * dati;

@end

@implementation ListaPartitaManager

static ListaPartitaManager *sharedClassInstance = nil;

+ (ListaPartitaManager *) sharedClass {}

- (void)setup {}

- (void) mappingJson {}

- (int)numeroDiPartite {}

- (NSString*)getSquadraCasaPerRiga:(int)riga {}

- (NSString*)getSquadraOspitePerRiga:(int)riga {}

- (NSString*)getDataPerRiga:(int)riga {}

- (NSString*)getOraPerRiga:(int)riga {}

- (NSString*)getIdGara:(int)riga {}

- (ListaPartitaManager*)dammiLaPartitaPerLaRiga:(int)riga {}

@end
```

Nel manager della lista partita viene creato un Singleton (Vedi **??**).

4.6.3 DettaglioPartitaManager.h

Nell'interfaccia vengono dichiarati i seguenti metodi:

```
#import <Foundation/Foundation.h>
#import "DettaglioPartitaModel.h"

@interface DettaglioPartitaManager : NSObject

@property (strong, nonatomic) DettaglioPartitaModel * dettaglioPartita;

+ (DettaglioPartitaManager * ) sharedClass;

- (void) MappingJSON :(NSString*)idGara;

- (DettaglioPartitaModel *) getDettaglioPartita;

@end
```

4.6.4 DettaglioPartitaManager.m

Di seguito l'implementazione dei metodi precedentemente dichiarati.

```
#import "DettaglioPartitaManager.h"
#import "DettaglioPartitaModel.h"
#import "CollaboratoreModel.h"
#import "ContattoModel.h"
#import <AFNetworking.h>

@implementation DettaglioPartitaManager

@synthesize dettaglioPartita = _dettaglioPartita;

+ (DettaglioPartitaManager *) sharedClass{}

- (DettaglioPartitaModel *) getDettaglioPartita{}
```

```
- (void) MappingJSON :(NSString*)idGara {}
```

Il metodo *"DettaglioPartitaManager"* implementa il pattern *Singleton*.
Il metodo *"MappingJSON"* mappa le interazioni contenute nella rappresentazione *JSON* in oggetti di modello *Objective-C*, incaricate alla vera e propria visualizzazione e interazione dell'App.

4.7 Controller

4.7.1 LoginViewController.h

La LoginViewController.h è una classe di tipo UIViewController.
Di seguito la dichiarazione dei metodi dell'interfaccia e delle sue proprietà:

```
#import <UIKit/UIKit.h>

#import <AFNetworking/AFNetworking.h>

@interface LoginViewController : UIViewController <UITextFieldDelegate>

@property (weak, nonatomic) IBOutlet UITextField *txtUsername;

@property (weak, nonatomic) IBOutlet UITextField *txtPassword;

- (IBAction)signInClicked:(id)sender;

- (IBAction)backgroundTap:(id)sender;

@end
```

4.7.2 LoginViewController.m

Di seguito descritti i metodi della classe implementativa.

```
- (void)viewDidLoad {}

- (void)didReceiveMemoryWarning {}
```

```
- (IBAction)signInClicked:(id)sender {}

- (void) alertStatus:(NSString *)msg :(NSString *)title :(int) tag {}

- (IBAction)backgroundTap:(id)sender {}

- (BOOL) textFieldShouldReturn:(UITextField *)textField {}
```

- Nel metodo **"signInClicked"** di tipo IBAction, viene implementata la richiesta di tipo "AFHTTPRequestOperationManager". Al tap sul button si verifica:
  1. L'avvenuto inserimento dei parametri per completare la login.
  2. Nel dictionary vengono dichiarati i due parametri da andare a confrontare nel Json, il Codice Meccanografico e la Password.
  3. Tramite due if si possono verificare i relativi successi. Se il successo è uguale a uno allora sono stati inseriti i parametri giusti, se invece il successo è uguale a zero vuol dire che i parametri inseriti sono errati e quindi appare il messaggio "Sign in Failed! Codice Meccanografico e/o Password errati".
- Il metodo **"alertStatus"** permette di far apparire una UIAlertView. Quest'ultima appare a video se avviene l'errato inserimento delle credenziali. Nell'UIAlertView appare la Figura 4.7 con visualizzato l'errore.

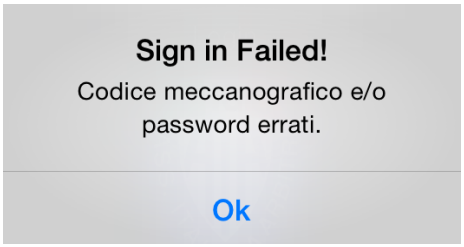


Figura 4.7: signInFailed

4.7.3 ListaPartiteViewController.h

La ListaPartitaController.h è una classe di tipo UITableViewController.
Di seguito la dichiarazione dei metodi dell'interfaccia.

```
#import <UIKit/UIKit.h>

@interface ListaPartiteViewController : UITableViewController

@property (nonatomic, strong) NSMutableArray *listaPartite;

@end
```

4.7.4 ListaPartitaViewController.m

Di seguito descritti i metodi della classe implementativa.

```
- (id)initWithStyle:(UITableViewStyle)style {}

- (void)viewDidLoad {}

- (void)didReceiveMemoryWarning {}

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {}

- (NSInteger)tableView:(UITableView *)tableView
      numberOfRowsInSection:(NSInteger)section {}

- (UITableViewCell *)tableView:(UITableView *)tableView
     cellForRowAtIndexPath:(NSIndexPath  {}*)indexPath

-(void) tableView:(UITableView *)tableView
     willDisplayCell:(UITableViewCell *)cell
        forRowAtIndexPath:(NSIndexPath *)indexPath {

- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {}

- (IBAction)logout:(id)sender {}
```

In questa classe viene utilizzato il Singleton che è un pattern tra i più importanti.

- Il metodo "viewDidLoad" viene implementato un NSNotificationCenter (Vedi 4.3) che permette l'avvenuto caricamento dei dati del Json. Di seguito viene invocato il Singleton che è stato creato nella ListaPartitaManager.
- Il metodo **"tableView:numberOfRowsInSection:"** di tipo NSInteger ritorna il Singleton con il numero di celle da creare nella ListaPartiteViewController.
- Il metodo **"tableView:cellForRowAtIndexPath:"** va a formare l'interno delle celle, quindi le rispettive label.
- Il **"metodo tableView:willDisplayCell:forRowAtIndexPath:"** si ha la possibilità di modificare il background della cella inserendo un'immagine a scelta.
- Il metodo **"prepareForSegue:sender:"** è molto importante perché grazie ad esso si ha la possibilità di passare alla view successiva implementando dei parametri. In questo caso ad ogni tap della cella si passa alla seguente view cioè il dettaglio della partita corrispondente, ed il tutto avviene grazie all'identificativo (Id) della gara che è stato assegnato nel Json.
- L'ultimo metodo **"logout"** di tipo IBAction permette all'utente di effettuare il logout e di tornare alla view precedente, view della login.

4.7.5 DettaglioPartitaViewController.h

La DettaglioPartitaViewController.h è una classe di tipo UIViewController.
Di seguito la dichiarazione dei metodi dell'interfaccia.

```
#import <UIKit/UIKit.h>

@interface DettaglioPartitaViewController : UIViewController

- (IBAction)accettaIsClicked:(id)sender;

@end
```

4.7.6 DettaglioPartitaViewController.m

Di seguito descritti i metodi della classe implementativa.

```
- (id)initWithNibName:(NSString *)nibNameOrNil
     bundle:(NSBundle *)nibBundleOrNil {}

- (void)viewDidLoad {}

- (void)didReceiveMemoryWarning {}

- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {}

- (IBAction)accettaIsClicked:(id)sender {}

- (void) alertStatus:(NSString *)msg :(NSString *)title :(int) tag {}
```

4.8 Collaboratori

4.8.1 CollaboratoriViewController.h

La CollaboratoriViewController.h è una classe di tipo UIViewController.
Di seguito la dichiarazione dei metodi dell'interfaccia.

```
#import <UIKit/UIKit.h>

@interface CollaboratoriViewController : UIViewController

@property (nonatomic, strong) NSMutableArray *listaCollaboratori;
@property (nonatomic, strong) NSMutableArray *listaContatti;

@end
```

4.8.2 CollaboratoriViewController.m

Di seguito descritti i metodi della classe implementativa.

```
- (id)initWithNibName:(NSString *)nibNameOrNil
      bundle:(NSBundle *)nibBundleOrNil {}

- (void)viewDidLoad {}

- (void)didReceiveMemoryWarning {}

- (void)valorizzaContatti:(CollaboratoreModel*)collaboratore
      cellulareLabel:(UILabel*)cellulareLabel
         email:(UILabel*)emailLabel {}
```

4.9 Accettazione Gara

L'accettazione della partita viene gestita tramite un *Pop-up*. I *Pop-up* sono degli elementi dell'interfaccia grafica, come ad esempio riguadri, che compaiono automaticamente durante l'uso di un'App ed in determinate situazioni, per attirare l'attenzione dell'utente. Attraverso il metodo *"alertView:didDismissWithButtonIndex:"* si implementa l'azione del *Button "ACCETTA"*. Subito dopo appare un altro *Pop-up* che da la possibilità all'utente di accettare o meno. Se si accetta il *Button* non ha più iterazione con l'utente, se invece si sceglie di fare *Tap* su "Cancel" allora si ritorna alla *View* del dettaglio partita e con il *Button* si potrà ancora interagire.

Capitolo 5

Testing

In questo capitolo viene descritto il collaudo (detto anche Testing) dell'App. Il testing dell'App di solito è di due tipologie. La prima è quella più utilizzata ed è l'*iOS Simulator* di *Xcode*. In questo progetto di tesi è stato utilizzato l'*iOS Simulator* con modello *4S* e sistema operativo mobile *iOS* di versione *"7.1"*.
Per avviare *iOS Simulator* si clicca su *Run* dell'ambiente di sviluppo *Xcode*.
La seconda tipologia viene effettuata al termine del progetto, e consiste nel caricamento dell'App in un dispositivo fisico. Questo viene effettuato soltanto quando tutte le funzioni sono operative e il design delle immagini è definitivo. In questo modo si può effettuare un vero e proprio testing dell'App vedendo il comportamento delle transizioni delle *View* e il sorgere di eventuali errori. *iOS Simulator* è di fondamentale importanza per il testing iniziale e finale ma soltanto interagendo con l'App in un dispositivo fisico si possono notare tutte le eventuali imperfezioni sia grafiche che di implementazione. Una volta terminata la fase di testing, se richiesto, l'App può essere distribuita ad un ristretto gruppo di persone per vedere la funzionalità in altri dispositivi e per avere un feedback di eventuali errori.

5.1 Avvio

Alla fase di avvio del simulatore viene visualizzata l'icona dell'App (Figura 5.1).
L'icona rappresenta un fischietto arbitrale, disegnata in maniera stilizzata, di colore bianco e nero. Successivamente si avvia l'App.

5.2 Schermata Login

Questo paragrafo rappresenta il vero e proprio testing per quanto riguarda l'interazione tra l'utente e il simulatore. La prima *View* che l'utente visualizza è quella che permette di effettuare la funzione di *Login* (Figura 5.2).

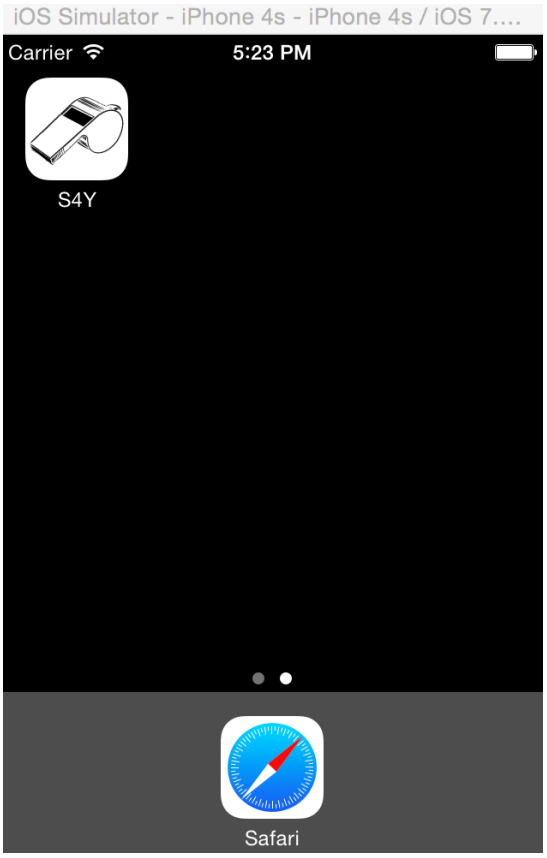


Figura 5.1: Home

La prima interazione che può effettuare l'utente è un *Tap* sul *Codice Meccanografico* così da inserire il proprio codice identificativo fornitogli dalla federazione al momento del tesseramento. Il passo successivo è quello di effettuare un altro *Tap* per inserire, nella *Text Field*, la *Password* che viene generata dal portale *Sinfonia 4 You* al momento della registrazione. L'ultima interazione che può effettuare l'utente è quella di azionare il *Button "Login"* con un *Tap*.
Se l'utente ha inserito entrambe le credenziali in maniera corretta ha la possibilità di accedere alla *View* successiva, la lista delle partite relative all'utente loggato.
Se invece l'utente inserisce le credenziali in maniera errata allora apparirà il *Pop-up* di errore (Figura 5.3), facendo *Tap* su "Ok" si ha la possibilità di reinserire in maniera corretta le credenziali.

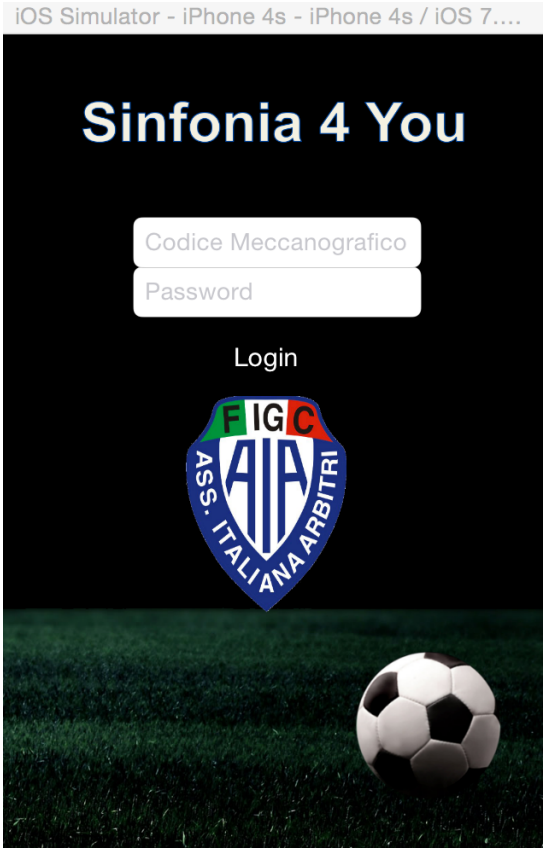


Figura 5.2: Login

5.3 Schermata Lista Partite

La seconda *View* (Figura 5.4) raffigura la lista delle partite dell'utente loggato. L'utente può vedere le celle composte dalle partite suddivise l'una dall'altra da una linea bianca. Le celle saranno composte dalle due squadre che si affronteranno nella gara, dalla data in cui viene svolta la gara e dall'ora stabilita. Lo sfondo della cella si presenta di colore verde cercando di riprendere il colore dei campi da calcio. L'interazione che ha l'utente è quella di poter fare *Tap* sulla cella così da far scattare l'azione che permette di andare alla *View* successiva. L'unica altra interazione che ha l'utente,al di là delle celle, è il *Button* che si trova alla fine della *View* di nome *Logout* e consente di tornare alla *View* della login.

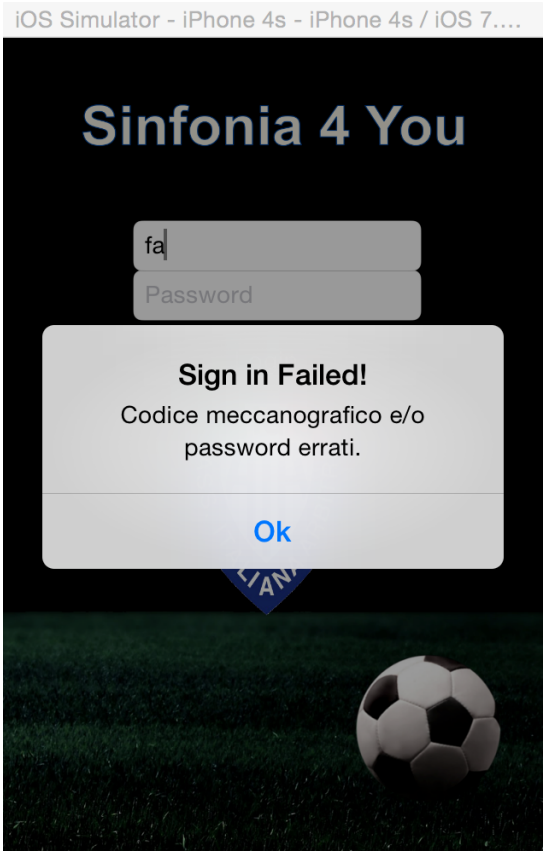


Figura 5.3: Login Error

5.4 Schermata Dettaglio Partite

La *View* (Figura 5.5) del dettaglio delle partite è la più articolata del progetto. Essa presenta nelle prime due *Label* le squadre che si fronteggeranno e a seguire saranno presenti tutte le specifiche riguardanti la gara. Su uno sfondo che raffigura un campo da calcio illuminato in notturna, vengono visualizzate all'utente tutti i dettagli della partita. Si visualizza la categoria per la quale l'utente è stato designato, attività che va a svolgere quindi "Arbitro" o "Assistente", il numero identificativo della gara, la data in cui si svolge la gara, l'ora, la città dove si svolge l'incontro e la provincia rispettiva, il rimborso spese al quale ha diritto l'*Associato*(calcolato in base al chilometraggio), i chilometri da percorrere per arrivare nella città in questione, il nome dell'impianto quindi dello stadio dove si svolge la gara ed infine l'indirizzo corrispondente.
Nella *View* sono presenti due *Button*:

- *Button Collaboratori*: con un *Tap* si accede alla *View* dei collaboratori.

Figura 5.4: Lista Partite

- *Button ACCETTA*: l'utente può interagire, con un *Tap*, e così facendo ha la possibilità di visualizzare attraverso un *Pop-up* la scelta tra "ACCETTA" e "Cancel". Facendo *Tap* su "ACCETTA" l'utente, successivamente, vede che il *Button* della *View* non può essere più attivo. Se invece l'utente sceglie di fare *Tap* su "Cancel" allora ritorna sempre alla *View* del dettaglio della partita e, al contrario di prima, ha ancora la possibilità di interagire con il *Button Accetta*.

Sulla *Navigation Bar* è presente un *Button* con il simbolo del "Segna Posto" che si trova di solito nei principali software, o siti *Web*, per il calcolo del percorso.
Azionando questo *Button* con un *Tap* si avvia l'App nativa dell'iPhone "Mappe".

Figura 5.5: Dettaglio Partita

5.5 Schermata Mappe

Questa *View* (Figura 5.6) viene visualizzata se l'utente vuole, al *Tap*, avviare l'applicazione nativa Mappe dell'iPhone. È stato implementato facendo in modo che l'applicazione delle Mappe si avvia in modo da avere nella posizione di partenza la posizione attuale del device che ha aperto l'App *Sinfonia 4 You* e nella posizione di arrivo l'indirizzo corrispondente della gara. A questo punto l'utente è libero di calcolare il percorso come meglio crede secondo le proprie necessità e sfruttando al meglio le potenzialità dell'applicazione nativa Mappe di *iOS*.

Figura 5.6: Calcolo del percorso

5.6 Schermata Collaboratori

La *View* (Figura 5.7) dei collaboratori è stata modellata al fine di avere separatamente la vista dei collaboratori. Si presenta all'utente con uno sfondo di un campo da calcio (sempre in notturna ma diverso dalla *View* del dettaglio partita). Vengo raffigurati le informazioni dei due collaboratori. Separatamente i due collaboratori hanno le *Label* che rappresentano il nome, il cognome, la sezione di appartenenza, il ruolo che il collaboratore ricopre nella gara, il numero di telefono per essere contattato dagli altri associati e la mail.

Figura 5.7: Collaboratori

Capitolo 6

Conclusioni e Sviluppi Futuri

Superata la fase di testing, ho potuto constatare e verificare che l'applicazione realizzata risulta funzionale. Durante il lavoro svolto ho avuto modo di imparare gli strumenti utilizzati, quali *Xcode* e *Balsamiq Mockups*, che mi hanno permesso di sviluppare in maniera professionale l'obiettivo che mi ero prefissato. Partendo dall'idea iniziale, si è potuto sviluppare un'applicazione mobile che mettesse a disposizione un servizio fino ad ora non esistente. Attualmente, ogni associato è costretto a loggarsi sul *browser* per poter consultare la propria designazione e i rispettivi dettagli. Per questo motivo, l'implementazione dell'applicazione mobile risulta utile alla consultazione del dettaglio partita, dei collaboratori e del calcolo del percorso. L'applicazione mobile per *iOS*, con la quale l'utente si trova ad interagire, è senza dubbio di basilare utilizzo.
Il servizio che è stato più utile sviluppare riguarda il calcolo del percorso. Grazie a questa funzionalità, gli associati in possesso di un dispositivo *iPhone* hanno la possibiltà di calcolare il percorso dalla posizione in cui si trovano fino alla posizione di arrivo - che poi sarà l'indirizzo dell'impianto sportivo al quale sono stati designati.

Per quanto riguarda gli sviluppi futuri, ho in progetto di sviluppare, insieme alla collaborazione dei responsabili informatici nazionali, i seguenti punti:

1. Testare l'applicazione mobile con il server dell'associazione, così da poter implementare anche l'accettazione della partita in maniera reale.

2. Implementare la sicurezza dell'applicazione mobile utilizzando dei *Token* di risposta.

3. Inserire l'anagrafica personale dell'assocciato.

4. Dare la possibilità all'associato di inserire eventuali congedi o indisponibilità.

5. Inserire una sezione che dia la possibilità ad ogni associato di visualizzare eventi come raduni o test atletici.

Concludendo, l'esperienza di progetto che ho sviluppato si è rivelata molto positiva, in quanto mi ha permesso di acquisire competenze come l'utilizzo del linguaggio di programmazione *Objective-C* e della piattaforma di sviluppo *Xcode*. Quest'ultimo è una prerogativa essenziale per lo sviluppo di applicazioni mobile nel mondo *Apple Inc.*. Molto utile è stato, infine, acquisire le fasi di progettazione, implementazione e testing, essenziali per lo sviluppo di un ottimo prodotto.

Bibliografia

[1] AIA. Associazione italiana arbitri. AIA.

[2] E. Amedeo. *Objective-C*. Apogeo, 2011.

[3] Apple. Gesture.

[4] Apple. Notificationcenter.

[5] Apple. Singleton.

[6] Andrea Bai. Smartphone: Android e ios dominano il mercato, ma il sistema operativo di google batte la mela. *http://www.businessmagazine.it/*, 13/02/2014.

[7] B. Dudney C. Adamson. *Sviluppare applicazioni con iOS SDK*. Apogeo, 2013.

[8] Roberto Cinotti. La storia dell'iphone - speciale. *http://www.everyeye.it*, 08/09/2014.

[9] CocoaPods.org. Cocoapods. http://guides.cocoapods.org/using/getting-started.html.

[10] Luigi Gia. App, business miliardario. *Repubblica*, 17/04/2014.

[11] A. Iacubino. *Creare Applicazioni di Successo per iPhone e iPad*. Hoepli, 2012.

[12] W. Isaacson. *Steve Jobs*. Mondadori, 2011.

[13] The JSON Data Interchange Standard. Json. http://json.org/json-it.html.

[14] Luca Vetti Tagliati. Architetture rest: un modello di maturità. *Mokabyte*, Dicembre 2011.