UNIVERSITÀ DI PADOVA FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

TESI DI LAUREA

SPREAKER:
SPREAKER RADIO PER IOS

Relatore: **Chiar.mo Prof. Sergio Congiu**

Laureando: **Alessandro Calzavara**

Anno Accademico 2011-2012

ai miei genitori,
per il loro sostegno
e per la loro fiducia,

ai miei amici e a Giulia,
per il loro affetto

ai miei colleghi di Spreaker
per la continua spinta a migliorare

e alla Romagna
per avermi accolto

Grazie

Indice

Capitolo 1

Introduzione

Nelle prossime pagine, dopo una breve introduzione al settore della creazione e diffusione di contenuti multimediali, con particolare riferimento al mercato degli smartphone, si analizzeranno i servizi offerti da Spreaker, lo stato dell'arte del mercato mobile e gli obiettivi che Spreaker vuole raggiungere.

1.1 Trasmettere audio dallo smartphone

Con la diffusione massiccia di Internet come mezzo di comunicazione per notizie, alternativo ai canali classici come la radio o la tv, ci si è imbattuti sempre di più in persone che, nel loro piccolo e con i pochi mezzi a disposizione, volevano raccontare una storia o un evento. Molto spesso, l'immediatezza data dall'avere in tasca uno strumento compatto ma altamente ricco di funzionalità come uno smartphone, ha portato ad una crescita importante del cosiddetto fenomeno del *reporting dal basso*.

Sono gli stessi utenti finali, che da passivi ascoltatori, si propongono come giornalisti per diffondere una registrazione di un evento che stanno vivendo. Il mercato quindi chiede servizi sempre più semplici ma ubiqui e alla portata di tutti, per registrare e condividere su Internet qualunque genere di media, sia esso immagine statica, video o audio.

Se ad oggi la creazione e diffusione di foto e video prodotte da smartphone sembrano una cosa irrinunciabile, con pochi servizi a veicolare la quasi tota-

lità dei contenuti, la creazione di contenuti audio non ha ancora trovato il suo servizio di riferimento.

Con *Spreaker Radio*, app[1] per iOS di *Spreaker*, si vuole ambiziosamente provare a lasciare un segno profondo in questo specifico settore.

Figura 1.1: Icona dell'applicazione Spreaker Radio

1.2 Spreaker

Spreaker è una piattaforma audio che consente a chiunque di creare, trasmettere e condividere un podcast[2] live attraverso internet in modo semplice ed accessibile. Spreaker offre un insieme di strumenti, tra i quali una console da DJ funzionante nel browser, una libreria musicale con brani ed effetti sonori, e la possibilitá di trasmettere tutti i contenuti audio prodotti (anche in diretta) su altri siti web, social network e blog.

L'azienda voleva entrare nel mercato mobile sviluppando un'applicazione nativa per la piattaforma iOS, in grado di fornire ai suoi utenti un approccio semplice e immediato, quindi nel suo stile, alla creazione e distribuzione di contenuti audio in mobilitá.

Inizialmente l'unico modo per utilizzare i servizi di Spreaker era quello di utilizzare un browser da un computer, nel quale, con l'utilizzo di tecnologie *Adobe Flash*®, si metteva a disposizione la ricca console di registrazione (do-

[1]app è il termine con il quale si intende un'applicazione software per smartphone con iOS.

[2]Podcast: si intende "una registrazione digitale di una trasmissione radiofonica o simili, disponibile su Internet con lo scopo di permettere il download su riproduttori audio personali", dizionario New Oxford, 2005

ve poter unire voce, musica ed effetti) e un riproduttore audio per ascoltare gli episodi creati.

La modalità e la tecnologia utilizzata limitava fortemente però l'utilizzo del servizio, escludendo completamente i dispositivi mobili in genere e, in particolare, i dispositivi iPhone.

1.3 Stato dell'arte

Sul mercato mobile, le soluzioni per la creazione e distribuzione di contenuti audio che riescono a dare un'esperienza da DJ, ossia consentono di mixare più tracce musicali contemporaneamente, sono molte. Il miglior esempio è sicuramente **djay**[3] della *algoriddim*.

Figura 1.2: Schermata dell'app djay della algoriddim

Tali soluzioni però non rispondono alle esigenze di chi vuole fare *radio* e non semplicemente mixare solo musica. Un dj radiofonico cerca un'esperienza meno basata sulla personalizzazione della musica e più sull'unione della propria voce con una o più basi musicali, dove può usare effetti sonori (come jingle ad esempio) di tanto in tanto, con lo scopo di raccontare ed intrattenere il proprio pubblico, possibilmente in diretta. App di questo genere non esistono e quelle che più si avvicinano, mancano sempre di qualche caratteristica importante. Molte registrano o solo voce o solo musica o non consentono di trasmettere in diretta.

[3]djay, algoriddim: `http://www.algoriddim.com/djay-iphone`

Tipicamente troviamo app che si comportano da semplici registratori di suoni e che, nei casi piu evoluti, integrano una funzione di distribuzione della registrazione al suo termine su servizi esterni come Soundcloud[4].

Manca quindi un'applicazione unica che fornisca tutte le caratteristiche sopra descritte, per far vivere agli utenti un'esperienza completa da dj radiofonico dal proprio dispositivo mobile.

1.4 Obiettivi

L'obiettivo primario di questo progetto è dunque lo sviluppo di un applicazione su piattaforma iOS in grado di riprodurre contenuti audio prodotti dagli utenti di *Spreaker* e di crearne di nuovi, dando un'esperienza utente simile (con le dovute limitazioni) ad un dj radiofonico, che trasmette la propria voce, musiche ed effetti e da ai suoi ascoltatori un canale per interagire con lui, in diretta, dal proprio smartphone.

[4]SoundCloud: piattaforma social per la creazione e condivisione di suoni, `http://soundcloud.com`

Capitolo 2

Il Progetto

Il progetto prevedeva la realizzazione di una app per dispositivi iPhone (piattaforma iOS) in grado di

- ascoltare contenuti audio creati dagli utenti, siano essi trasmessi in diretta e non
- trasmettere in diretta contenuti audio ottenuti tramite voce, musica ed effetti
- gestire il profilo e i contenuti prodotti dell'utente.

Questi macro obiettivi sono stati analizzati attraverso una serie di casi d'uso, per determinare più in dettaglio i requisiti dell'applicazione. A lato dello sviluppo dell'app iOS, sono stati anche realizzati e messi in produzione i servizi lato server senza i quali non si sarebbero potuti raggiungere gli obiettivi richiesti.

Oltre a questi macro obiettivi, si sono susseguiti una lunga serie di obiettivi secondari, come l'implementazione del login e della registrazione di un utente, l'integrazione con vari social network (Facebook e Twitter) e con servizi esterni (SoundCloud), o ancora la risoluzione di problemi segnalati dagli utenti. Nonostante questi non siano ne superflui ne più semplici dei macro obiettivi qui analizzati, si vuole dar maggior risalto a questi in quanto rappresentano il cuore del progetto.

2.1 Ascolto di contenuti audio

Caso d'uso

L'utente scopre i contenuti audio disponibili (episodi) tramite una serie di elenchi visualizzati dall'app. Tali elenchi sono ordinati per popolarità, per vicinanza alla sua posizione o per preferenza personale. Da questi elenchi, l'utente sceglie uno degli episodi, facendo così iniziare la riproduzione dello stesso mentre l'app mostra una schermata dedicata ai controlli del volume e della riproduzione.

Analisi

Dal caso d'uso si evidenziano due schermate differenti:

1. un elenco di episodi disponibili per l'ascolto
2. una schermata che permette di vedere riepilogato cosa l'utente sta ascoltando e di controllarne la riproduzione.

Gli elenchi e le descrizioni degli episodi sono già messi a disposizione da *Spreaker* tramite delle **API REST**[1]. Sarà solo necessario inserire nelle singole richieste dell'app la posizione geografica e un identificatore (un ID) dell'utente che fa la richiesta.

Per i contenuti audio veri e propri, vi sono due strade possibili:

1. rendere scaricabile il file audio col contenuto che l'utente vuole ascoltare
2. creare uno stream audio per il contenuto richiesto.

Se la prima è la soluzione più semplice da implementare lato server, non consente la diffusione e l'ascolto di contenuti creati in diretta, per i quali uno stream è l'unica soluzione possibile. Inoltre quest'ultima risulta essere più flessibile perchè consente anche l'invio di metadati all'interno del flusso audio, nonché inserimento di segmenti audio diversi (come spot pubblicitari) targhettizzati per l'ascoltatore.

[1] API REST: vedi paragrafo 3.3.1

Tra i possibili protocolli disponibili per lo streaming si è scelto Icecast (vedi il paragrafo 3.3.2 per maggiori dettagli) in quanto è largamente usato e semplice da implementare. Un altro protocollo per lo scopo poteva essere RTMP[2] di Adobe, protocollo usato per lo streaming verso riproduttori audio Flash ma la sua implementazione su iOS sarebbe stata molto più complessa e non avrebbe dato benefici maggiori.

2.2 Trasmissione in diretta

Caso d'uso

L'utente, una volta aperta la console, avvierà la registrazione e in pochi secondi potrà iniziare a parlare ed inserire musica all'interno del suo nuovo episodio. Fatto partire il primo brano, preparerà il secondo pezzo musicale da mettere in onda e risponderà ai saluti dei suoi ascoltatori direttamente con la propria voce o in chat. Userà degli effetti sonori durante le parti di parlato per rendere più piacevole l'ascolto e dopo i saluti finali, premerà il pulsante di fine registrazione, terminando così l'episodio.

Analisi

Dato che un singolo smartphone non può farsi carico neanche di una decina di connessioni verso cui inviare contemporaneamente dati, l'unico modo per consentire un numero arbitrariamente grande di ascoltatori è quello di trasmettere l'audio generato dal telefono ad un server, il quale registrerà l'audio (e ne convertirà il formato) e potrà gestire tutte le connessioni in arrivo dagli ascoltatori. Questi quindi si collegheranno al server per ricevere, in streaming, l'audio prodotto dal dj.

Risulta comunque evidente l'importanza dell'interfaccia utente in questo caso d'uso. Tutta la registrazione (e trasmissione) che l'utente fa viene eseguita da un unica schermata, dalla quale deve avere il totale controllo sull'audio che sta registrando. È per tanto necessario dedicare particolare cura alla scelta degli elementi da mettere a disposizione dell'utente, per mantenere l'interfaccia chiara ed utilizzabile (ricordo che stiamo lavorando con uno

[2]Specifiche protocollo RTMP di Adobe `http://www.adobe.com/devnet/rtmp.html`

smartphone, con limitazioni sia nella potenza di calcolo che nelle possibilità di inserire controlli per l'utente), nonchè cercare di ottimizzare il più possibile il codice per evitare blocchi o scatti di audio durante la registrazione.

Oltre a dei controlli per gestire le tracce musicali, vi saranno anche dei pulsanti per far partire e fermare alcuni effetti sonori, un regolatore di volumi e, data la natura mobile del progetto, un indicatore della qualità della connessione di rete con il server.

Nel caso d'uso si fa anche riferimento ad una chat, ossia un servizio tramite il quale sia gli ascoltatori che il dj possono scambiarsi messaggi. Anche questa funzionalità dovrà trovare spazio in quella schermata di registrazione.

2.3 Gestione del profilo e dei contenuti dell'utente

Caso d'uso

L'utente accede al suo profilo per controllare e modificare informazioni come il nome, la posizione geografica e il tipo di abbonamento premium[3] che possiede. Può visionare gli utenti che apprezzano le sue creazioni (chiamati *follower*) per ricambiare l'apprezzamento, ma soprattutto vedrà i propri show (aggregati di episodi) per modificarne alcuni dettagli, come la descrizione o gli episodi in esso contenuti. Potrà pure modificare i singoli episodi, cambiandone il nome o la descrizione o, nel caso lo ritenga opportuno, potrà eliminarli in modo definitivo.

Analisi

Dal caso d'uso si può notare come, innanzitutto, siano prese in considerazione tre risorse differenti: utenti, show ed episodi. Utenti ed episodi sono entità semplici mentre lo show è definito come aggregato di vari episodi di uno stesso utente (ne è l'autore).

Possiamo così distinguere, oltre alla schermata con il profilo dell'utente, tre schermate di modifica dati:

[3]Spreaker ha vari piani di abbonamento per gli utenti, di cui uno gratuito. Ogni piano ha delle limitazioni, come la durata massima dei singoli episodi e la quantità di episodi che si possono tenere online.

1. schermata per la modifica del profilo utente
2. schermata per la modifica di uno show
3. schermata per la modifica di un episodio.

Per quanto riguarda la schermata di sola visualizzazione del profilo dell'utente, oltre al mostrare effettivamente le informazioni, vi sarà un pulsante con il quale l'utente potrà iniziare a modificarle. Tale pulsante troverà spazio anche nelle schermate di presentazione di uno show (verrà presentata nel paragrafo 4.3.4) nel caso l'utente ne sia l'autore.

Le schermate di modifica mostreranno all'utente tutte le informazioni modificabili della risorsa, sia quelle già inserite che quelle ancora vuote. Sarà poi lui a scegliere quale voce modificare, selezionandone una. Così facendo, verrà aperta una schermata ad-hoc nella quale cambiare il valore di tale informazione. Una volta terminata la modifica, l'utente ritornerà alla schermata principale di modifica dove potrà rivedere le modifiche fatte e salvarle inviandole al server tutte in una volta.

Le informazioni modificate verranno inviate al server tramite una API. Questa eseguirà la validazione di ogni informazione e, in caso di esito positivo, aggiornerà la risorsa modificata.

Capitolo 3

Tecnologie

La scelta della piattaforma iOS comporta varie limitazioni, dall'ambiente di sviluppo al linguaggio da usare. Per tale piattaforma, Apple mette a disposizione un SDK[1] sufficientemente ricco per sfruttare in modo pieno tutte le caratteristiche hardware e software dei dispositivi iOS e, data la forte adozione che la piattaforma ha avuto in questi anni, ha anche una vasta community di sviluppatori che rilasciano molte librerie opensource in rete.

3.1 Dispositivi iPhone, iPod ed iPad

Apple ha prodotto nel 2007 uno smartphone assolutamente rivoluzionario, l'iPhone, che ha cambiato il modo definitivo il settore della telefonia mobile. Il sistema operativo iOS unito alle caratteristiche hardware di assoluto pregio, hanno reso questo prodotto molto interessante sia per l'utilizzatore finale che per le aziende e gli sviluppatori indipendenti, che hanno iniziato a creare moltissime applicazioni per questa piattaforma. Un applicazione veniva intesa come un aggiunta di *funzionalità* al telefono e prendeva il nome di **app**.

Dopo il successo del primo iPhone, è stata prodotta una particolare versione dell'iPod, l'iPod Touch, il quale non era altro che un iPhone senza componente telefonica, ma basato sullo stesso sistema operativo.

[1] SDK o Standard Development Kit, è un insieme di strumenti, librerie e documentazione che consentono lo sviluppo di un applicativo

Con l'introduzione nel 2010 del primo tablet della storia, l'**iPad**, iOS è stata la scelta più ovvia come sistema operativo per questo nuovo genere di dispositivo.

Tre tipologie di dispositivi con lo stesso sistema operativo hanno portato ad una crescita esponenziale del numero di applicazioni disponibili per gli utenti, rendendo l'intera piattaforma iOS una delle più floride degli ultimi decenni.

Dati di vendita del terzo quadrimeste 2012 [2] riportano che Apple ha venduto, solo in quel quadrimestre, 26 milioni di iPhone, 17 milioni di iPad e 6.8 milioni di iPod complessivamente nel mondo. Abbiamo quindi un prodotto molto desiderato dal mercato. Complessivamente sono stati venduti 250 milioni di iPhone dal 2007 ad oggi [3] e si stima che, solo negli Stati Uniti, siano correntemente in uso circa 35 milioni di iPhone [4].

3.2 L'SDK di iOS

L'SDK di iOS mette a disposizione una serie di software per lo sviluppo del codice e un buon numero di librerie per utilizzare a pieno le funzionalità dei dispositivi con iOS.

3.2.1 Xcode

Xcode è l'IDE[5] presente nell'SDK di iOS (è presente anche in quello per Mac OS X, il sistema operativo per desktop e portatili di Apple) ed è fornito gratuitamente da Apple stessa tramite il suo **Mac App Store**[6]. Tale IDE è svilup-

[2] Fonte Apple, Risultati Q3 2012, `http://www.apple.com/pr/library/2012/07/24Apple-Reports-Third-Quarter-Results.html`

[3] Fonte BusinessInsider, `http://www.businessinsider.com/apple-has-shipped-250-million-iphones-in-the-past-five-years-2012-6`

[4] Fonte ComScore, `http://blog.comscore.com/2012/06/5_years_later_a_look_back_at_the_rise_of_the_iphone.html`

[5] IDE o Integrated Development Environment è un software che integra tutta una serie di strumenti sotto un unica interfaccia, tramite il quale lo sviluppatore viene aiutato nello sviluppo di altre applicazioni.

[6] Mac App Store è il nome del negozio virtuale dove gli utenti possono acquistare e scaricare applicazioni per i computer Apple con installato Mac OS X 10.6.6 o successivi.

pato solo per sistemi operativi Mac OS X ed è quindi necessario un computer Apple per poterlo installare ed usare.

Figura 3.1: Xcode durante l'uso di Interface Builder (fonte Apple)

Oltre alle funzioni di editor di codice, Xcode integra un compilatore *LLVM*, il relativo debugger *LLDB* e un software per costruire interfacce grafiche chiamato *Interface Builder*. Assieme ad Xcode vengono messi a disposizione un simulatore di dispositivi iOS e uno strumento per l'analisi approfondita del software, chiamato *Instruments*, in grado di eseguire analisi sul comportamento del codice, come ad esempio l'andamento dell'uso della memoria o il consumo di energia durante l'esecuzione vera e propria del software.

Xcode inoltre è ben integrato con il portale per sviluppatori Apple (chiamato *Dev Center*) tanto da essere in grado di scaricare e rinnovare i certificati necessari per lo sviluppo e la pubblicazione di software su *App Store*[7], aggiungere automaticamente nuovi dispositivi alla lista dei dispositivi per il testing,

[7]App Store è invece il negozio virtuale per dispositivi iOS dove gli utenti possono acquistare e scaricare applicazioni per i propri dispositivi portatili.

e ovviamente pubblicare le app sul sito *iTunes Connect*[8] per il rilascio delle stesse nello store.

Compilatore LLVM

LLVM (Low Level Virtual Machine) è un infrastruttura di compilazione progettata per ottimizzare software durante le fasi di compilazione, linking, esecuzione e non di codice applicativo.

Il progetto LLVM[9] è stato sviluppato inizialmente presso l'Università dell'Illinois nel 2000 con lo scopo di riunire una serie di strumenti e tecnologie all'avanguardia per l'analisi statica e dinamica del codice. Tale progetto è stato poi rilasciato in licenza Open Source nell'ottobre del 2003.

Il compilatore **Clang**, nato da questo progetto, supporta i linguaggi C, C++, Objective-C[10] ed Objective-C++ (variante dell'Objective-C) e risulta essere tre volte più veloce rispetto a GCC[11] a compilare codice Objective-C.

Xcode 3.2.1 ha incluso il compilatore LLVM Clang come compilatore sperimentale, assieme al più consolidato GCC. Dalla versione 4.2 invece, Clang è diventato il compilatore di default.

Interface Builder

Interface Builder (IB) è un'applicazione per sviluppo di software per sistemi Mac OS X e iOS integrato in Xcode. IB consente agli sviluppatori di creare interfacce utente e collegarle al codice in maniera grafica, semplice e veloce.

IB produce file *.xib*, ossia dei file XML, che descrivono l'aspetto e la posizione dei vari elementi grafici di una vista. Definendo poi anche un oggetto responsabile per la vista (tipicamente un oggetto di tipo *View Controller*) si potranno collegare gli elementi grafici a variabili (o proprietà) dell'istanza dell'oggetto responsabile.

[8] iTunes Connect è il portale tramite il quale gli sviluppatori si possono pubblicare le loro app, `http://itunesconnect.apple.com/`

[9] The LLVM Compiler Infrastructure, `llvm.org`

[10] Il linguaggio Objective-C viene descritto nell'appendice A.

[11] GCC, GNU C Compiler è il compilatore creato da Richard Stallman nel 1987 nell'ambito del progetto GNU.

Non è obbligatorio usare questo applicativo per creare le viste ma torna molto utile usarlo per definire e posizionare velocemente tutti gli elementi che ci saranno. Sarà comunque possibile rifinire successivamente la vista complessiva via codice, creando nuovi elementi o agendo sugli elementi già inseriti.

Simulatore iOS

Assieme ad Xcode viene anche fornito un simulatore di dispositivi iOS.

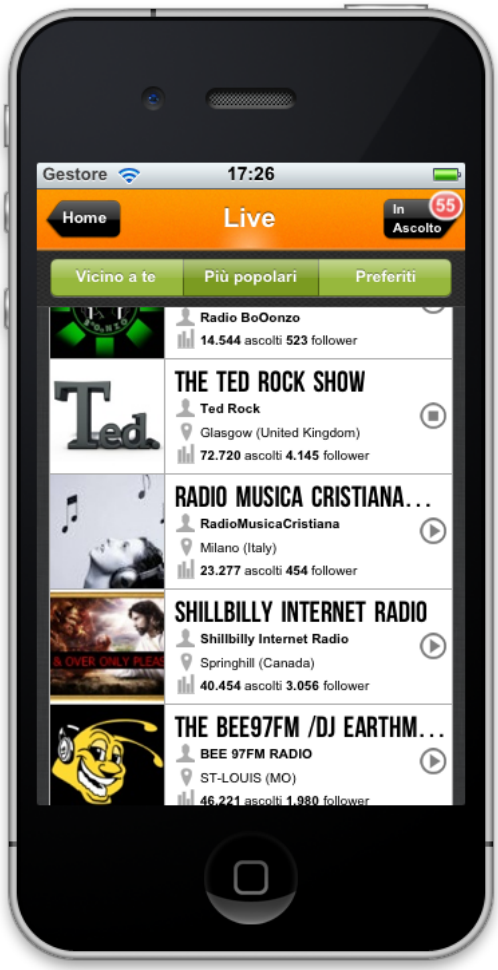

Figura 3.2: Spreaker Radio in esecuzione nel simulatore

Questo software emula quasi completamente iOS e, nonostante usi l'architettura hardware del computer su cui è in esecuzione (l'applicazione deve essere compilata per l'architettura x86/x64 e non per ARM per essere eseguita nel simulatore), riesce ad essere molto utile per verificare il funzionamento dell'applicativo che si sta sviluppando.

Il simulatore consente anche di scegliere il dispositivo che si vuole simulare. Si può scegliere se simulare un iPhone (il che comprende anche l'iPod) o un iPad, quale tipo di schermo e quale risoluzione esso deve avere (per iPhone sono supportate le risoluzioni in pixel 480x320, 960x640 e 1136x640, ossia lo schermo dell'ultimo iPhone 5; per gli iPad 1024x768 e 2048x1536) e quale versione di iOS eseguire (dalla 4.3 alla 6.0[12]).

Rispetto alla compilazione ed esecuzione dell'applicativo su dispositivo fisico, il simulatore consente una compilazione ed installazione più rapida, riducendo così i tempi morti tra scrittura del codice e la sua esecuzione. Alcune feature non sono supportate dal simulatore, come l'accesso alla libreria musicale, l'uso reale del GPS (il simulatore restituisce sempre una posizione predeterminata), l'invio di sms o email (viene simulato un invio con successo senza effettivamente inviare la mail) ed altre ancora. Di conseguenza, il simulatore risulta un ottimo strumento per sviluppare la struttura principale dell'app ed eseguire test automatici, ma non sostituisce i test su dispositivi reali.

Instruments

Instruments è un software in grado di registrare diverse informazioni sul comportamento di un processo in esecuzione per analizzarle a posteriori.

Rende quindi possibile analizzare le performance (ad esempio in termini di memoria e uso di CPU), eseguire test automatici o stress test della propria applicazione, dando una profonda conoscenza di come un software si comporti.

È possibile usare Instruments sia con il simulatore iOS che con dispositivi fisici, collegati tramite USB al computer.

[12]Le versioni di iOS supportate dal simulatore dipendono dalla versione di Xcode e non è possibile installare tutte le versioni che si desidera.

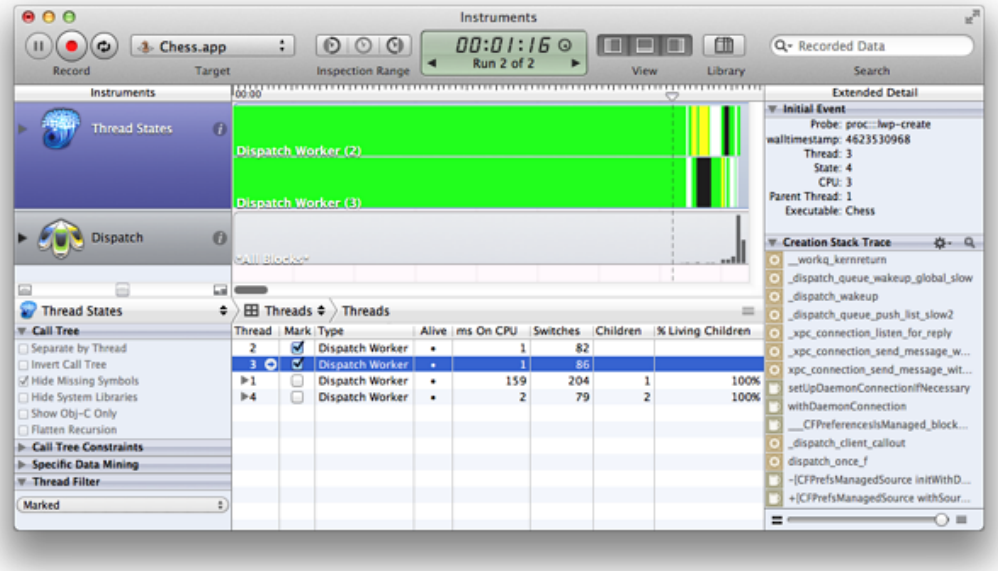

Figura 3.3: Instruments durante una sessione di registrazione (fonte Apple)

Per ogni sessione di registrazione, si possono scegliere una serie di strumentazioni da usare. Ogni strumentazione raccoglie informazioni specifiche come, ad esempio, le allocazioni in memoria (numero e taglia), il numero di connessioni di rete o il carico della CPU. Una volta selezionati gli strumenti di interesse, si esegue l'applicativo e si lascia Instruments registrare tutte le informazioni richieste.

Durante e al termine della registrazione è possibile vedere i dati raccolti ed analizzarli. Effettuata l'eventuale modifica al codice, è possibile eseguire una nuova raccolta dati e confrontarli con la registrazione precedente.

3.2.2 Linguaggi

I linguaggi supportati da Xcode, tramite il compilatore Clang, sono: C, C++, Objective-C ed Objective-C++. In questo progetto i linguaggi effettivamente utilizzati sono stati C ed Objective-C, dato che tutte le librerie di iOS sono sviluppate e sono disponibili esclusivamente in questi due linguaggi.

C è un linguaggio procedurale che non ha certo bisogno di presentazioni.

Objective-C (o ObjC) è invece un linguaggio molto poco conosciuto in quanto viene usato quasi esclusivamente nello sviluppo di software in ambiente Mac ed iOS.

ObjC è un linguaggio ad alto livello orientato agli oggetti, basato sul passaggio di messaggi tra oggetti (come Smalltalk[13]) e senza una forte tipizzazione. Un approfondimento su questo linguaggio viene fatto nell'appendice A.

3.2.3 Librerie di iOS

Tra tutte le librerie (chiamate *Framework*) presenti nell'SDK di iOS, verranno qui descritte solo quelle più interessanti per questo progetto.

UIKit

UIKit è il framework che racchiude tutte le classi necessarie a costruire le interfacce utente delle applicazioni per iOS. Fornisce la gestione degli eventi dallo schermo, le finestre (`UIWindow`) e le viste (la generica vista è `UIView` mentre alcune sotto classi sono `UIImageView`, `UIScrollView` e `UINavigationBar`), oltre ai controlli progettati per interfacce touch screen (ad esempio pulsanti generici `UIButton` e quelli per le barre `UIBarButton`, gli interruttori `UISwitch` e selettori multipli `UISegmentedControl`).

Core Audio

Core Audio mette a disposizione un insieme di servizi tramite i quali si implementano le caratteristiche audio (o musicali) delle applicazioni iOS. Consente ad esempio la registrazione, la modifica e la riproduzione di dati audio in formato sia compresso che non, elaborazioni di segnali e sintesi musicale.

Tale framework lavora molto a basso livello ed è quindi scritto completamente in C per consentire la massima efficienza in termini di latenza (nell'elaborazione di segnali audio, una bassa latenza è importantissima). Integra il supporto per la lettura e scrittura di molti formati (*codec*) audio in molti file diversi (*container*) e la creazione di catene audio per l'elaborazione di suoni.

[13]Smalltalk: `http://en.wikipedia.org/wiki/Smalltalk`.

Core Location

Core Location è il framework che consente la determinazione della posizione corrente del dispositivo (tramite GPS) e della sua orientazione (tramite bussola). Tramite le sue API è possibile configurare il modo in cui gli eventi di posizione vengono inviati al nostro codice e a quali eventi siamo interessati (ad esempio se usciamo da una certa regione).

Tecnologie di iOS

Ci sono tecnologie a disposizione dell'utente estremamente utili per superare difficoltà personali come quelle per l'accessibilità. Su iOS abbiamo a disposizione una tecnologia in particolare che consente ad ipovedenti o non vedenti di interagire con il proprio dispositivo ed usare qualunque (in via teorica) applicazione installata su di esso.

VoiceOver

La radio da sempre rende possibile la comunicazione a quelle persone che hanno perso la vista. Nella realizzazione di quest'app, si vuole poter dare anche a queste persone la possibilità di fare radio.

VoiceOver è il nome del servizio che, su dispositivi sia Mac che iOS, legge le informazioni presenti sullo schermo e, tramite dei semplici gesti dell'utente, consente la navigazione all'interno del software (cercare un pulsante, premerlo, tornare alla schermata precedente, e così via).

Esso permette all'utente di sapere cosa c'è in ogni punto dello schermo semplicemente toccandolo o scorrendo in sequenza tutti gli elementi presenti. L'utente quindi può interagire direttamente con lo schermo mentre VoiceOver descriverà quello che trova in quel punto. Se l'utente toccherà due volte lo schermo, VoiceOver eseguirà l'azione legata all'ultimo elemento selezionato (se fosse un bottone, verrebbe eseguita l'apertura di una nuova vista, ad esempio).

Inserendo negli elementi grafici visibili piccole informazioni specifiche per VoiceOver, si rende l'app facilmente utilizzabile con questo servizio e di conseguenza accessibile anche a chi non può vedere cosa accade sullo schermo con i propri occhi.

3.3 Comunicazione app-server

Le comunicazioni tra app e server sono di due tipi: testuali e multimediali. Il primo tipo viene fatto per scambiare informazioni sulle risorse messe a disposizione da Spreaker (gli episodi, gli utenti, gli show e così via) mentre il secondo viene fatto per trasmettere ed ascoltare episodi.

Per le trasmissioni testuali, si è scelto di utilizzare delle API REST mentre per quelle multimediali, si è scelto il protocollo Icecast.

3.3.1 API REST

Per API si intende un interfaccia di programmazione che facilita il re-uso del codice, astraendo logica applicativa dietro una semplice invocazione remota.

REST (REpresentational State Transfer) è un architettura software che si basa sul concetto di risorsa. Ogni risorsa è identificata univocamente da un *URI*[14] ed è possibile cambiarne lo stato invocando su di essa delle operazioni. Ogni richiesta è senza stato e deve contenere tutte le informazioni necessarie per la sua esecuzione. Solo il client deve tener traccia delle eventuali richieste precedenti mentre il server può processare ogni singola richiesta per conto proprio (per quanto possibile).

In questo caso specifico (seppur molto comune), Spreaker ha sviluppato delle API REST su protocollo HTTP[15]. Questo consente, con una semplice richiesta HTTP con metodo `GET` su un URL[16] preciso, di ottenere tutte le risorse disponibili in un dato momento, o con una richiesta `DELETE` di cancellarne una. I metodi del protocollo HTTP vengono quindi usati per eseguire operazioni sulle risorse identificate da un URI.

3.3.2 Icecast

Il protocollo di streaming Icecast è un protocollo per lo streaming di contenuti multimediali su Internet, del tutto simile a SHOUTcast, sviluppato dalla *Nullsoft*. SHOUTcast è però un protocollo proprietario, mentre Icecast è opensource.

[14]URI: Uniform Resource Identifier
[15]HTTP: HyperText Transfer Protocol
[16]URL: Uniform Resource Locator, un indirizzo di una risorsa in rete

Icecast prevede la trasmissione client-server di dati audio/video intervallati (opzionalmente) da metadati e usa HTTP come protocollo di trasporto. Consente quindi l'invio di un flusso continuo di audio nonchè di informazioni testuali contestuali all'audio (ad esempio, quale titolo ed autore abbia la canzone che si sta trasmettendo in questo preciso momento).

Data la semplicità di tale protocollo, è spesso usato nell'ambito dello streaming di contenuti sul web, nonostante HTTP non sia il protocollo di trasporto più indicato per la trasmissione di contenuti in diretta.

Capitolo 4

Lavoro

Durante il tirocinio si è strutturato il lavoro di settimana in settimana, con opportune riunioni per valutare i risultati conseguibili in quel lasso di tempo. Nonostante l'implementazione sia stata lasciata completamente al sottoscritto, il confronto sugli schemi di sviluppo più indicati da utilizzare o sulle strategie per affrontare un determinato problema emerso durante lo sviluppo, è stato continuo, grazie al supporto del tutore aziendale e degli altri colleghi.

4.1 Organizzazione

Data l'importanza che viene data alla voce degli utenti, si è da subito cercato di lavorare settimanalmente, progettando ed implementando soluzioni che dovevano in breve tempo essere rilasciate al pubblico. Per questo motivo si è scelto di investire del tempo, tutte le settimane, per organizzare il lavoro di tutti e discuterlo insieme.

Appena si aveva una modifica tangibile per l'utente, si rilasciava una versione Beta dell'app (di fatto non era altro che la stessa versione disponibile a tutti nell'AppStore ma con solo qualche caratteristica in più) ad un buon numero di utenti, che provava in anteprima la nuova caratteristica e ci riportava il proprio pensiero a riguardo. Questo feedback è stato importantissimo anche per scovare problemi che durante lo sviluppo non erano emersi.

Dopo un certo tempo, variabile a seconda delle modifiche ed aggiunte effettuate, si pubblicava l'applicazione Beta come versione normale nell'AppStore

tramite il servizio *iTunes Connect* di Apple.

4.1.1 Riunioni settimanali

Il lunedì mattina veniva svolta una riunione per decidere quali attività svolgere in quella settimana, in modo aperto. Ognuno di noi, a turno, programmava la propria settimana con i lavori da svolgere, valutandoli con il tempo che avremmo impiegato per svolgerli.

Questa attività aveva un duplice scopo:

- far conoscere a tutti quali lavori venivano affrontati dai colleghi in quella settimana, per fare squadra ed organizzare gli eventuali lavori che richiedono più compentenze (e dunque persone diverse)
- imparare a conoscere il costo (in tempo) dei lavori che stiamo per svolgere e imparare a rispondere alla domanda *"quanto tempo ci vuole per realizzare questa funzionalità ?"*

Al termine della settimana, il venerdì pomeriggio, si svolgeva un altra breve riunione in cui, a turno, si raccontava l'andamento della propria settimana lavorativa, sia dal punto di vista della schedulazione fatta il lunedì, cercando di spiegare l'eventuale errore nelle stime fatte, sia dal punto di vista personale sul lavoro svolto o sulle difficoltà o imprevisti accaduti.

4.1.2 Feedback degli utenti

Dati i lunghi tempi di pubblicazione che l'App Store comporta, è importante fare in modo che l'applicazione sia il più stabile possibile perchè ogni versione rimane comunque a disposizione degli utenti per un tempo piuttosto lungo. Abbiamo quindi scelto di raccogliere un buon numero di utenti desiderosi di provare in anteprima la versione futura dell'app per ottenere da loro informazioni sul comportamento dell'app stessa e scoprire problemi prima che tale versione arrivi nell'App Store per tutti gli utenti del mondo.

L'interazione con questa cerchia di utenti, i beta tester, è molto utile in quanto permette di mettere alla prova le modifiche introdotte su un parco di dispositivi molto più ampio di quello che avevo personalmente a disposizione, sia per tipi di dispositivi (varie versioni di iPhone, iPod ed iPad) nonchè del

sistema operativo installato su di essi (da iOS 4.0 all'ultimo iOS 6.0) e dal carico ed utilizzo che questi utenti ne fanno (quantità di musica installata, spazio libero, jailbreak[1], ecc).

I betatester possono contattarci liberamente ed in via diretta per segnalarci situazioni di criticità o di successo nell'uso dell'applicazione, ma anche lo sviluppatore (il sottoscritto) può controllare le varie sessioni di utilizzo che ogni applicazione registra e rende disponibile tramite un sito web allo sviluppatore.

Il servizio usato per interagire con i betatester è *TestFlight*[2], e consente il reclutamento di nuovi utenti, la distribuzione delle nuove versioni dell'app e un sistema per evidenziare le sessioni fatte e i crash che avvengono.

Il rapporto diretto con questo gruppo di utenti volenterosi, ha permesso in molte occasioni di capire e sistemare problemi che non erano emersi sui dispositivi disponibili in ufficio. Importanti sono stati i feedback di quegli utenti che utilizzavano l'app con tecnologie per l'accessibilità come *VoiceOver*[3], i quali usano il loro dispositivo con "occhi" molto diversi dai nostri ma possono ugualmente avere benefici dall'uso di tanti software.

4.2 Strutturazione del codice

Il codice dell'app è stato stratificato su cinque livelli. Dal basso troviamo:

- Modello dei dati
- Servizi
- Manager
- Controller
- Viste.

[1]Per jailbreak si intende una procedura informatica fatta con lo scopo di rimuovere determinati limiti di sicurezza imposti da iOS e consentire quindi l'installazione di software non verificato per cambiare il comportamento del dispositivo.

[2]TestFlight, servizio per il testing dedicato agli sviluppatori iOS, `www.testflightapp.com`

[3]VoiceOver: vedi paragrafo 3.2.3.

Tale strutturazione è un estensione del design pattern architetturale MVC[4] e migliora la pulizia del codice nei controller.

Figura 4.1: Stratificazione del codice di Spreaker Radio

Nel modello dei dati abbiamo classi che rappresentano entità di interesse, come l'utente e l'episodio.

Nei servizi abbiamo classi che si occupano di interagire con i dati, con lo scopo di generarli e manipolarli.

I manager invece fanno da tramite tra i Controller e i Servizi, ascoltando e generando notifiche nell'app.

I controller si occupano di gestire l'interazione con l'utente, ascoltando e generando eventi da/per le viste e da/per le notifiche dei manager. In questo modo c'è una separazione totale tra la logica applicativa e l'interfaccia utente.

Le viste, infine, si occupano solo di mostrare le informazioni all'utente, qualunque sia la loro origine.

[4]Design pattern MVC o Model-View-Controller, `http://en.wikipedia.org/wiki/Model-View-Controller`

Modello dei dati

Questo livello rappresenta il modello dei dati a partire dalle informazioni che le API di Spreaker rendono disponibili.

Abbiamo varie classi che rappresentano entità ovvie come l'utente, lo show e l'episodio. Ci sono anche classi che rappresentano lo stream a cui l'app deve collegarsi per eseguire l'ascolto di un episodio o il server di registrazione al quale collegarsi per trasmettere un episodio in diretta.

Ogni possibile risorsa è modellata in una classe corrispondente.

Servizi e Manager

I servizi invece sono le entità che creano e agiscono sui dati.

Abbiamo il servizio che esegue le richieste alle API di Spreaker e si occupa quindi di scaricare le informazioni e convertirle dalla loro rappresentazione testuale (JSON[5] ad essere precisi) al modello corrispondente. Altri esempi sono il servizio che si occupa di eseguire il mix delle varie fonti audio, come il microfono e i file audio delle canzoni e degli effetti, quello che si occupa di eseguire l'encoding (la conversione di formato) dell'uscita del mixer per generare i pacchetti audio da inviare al server di registrazione, quello che si occupa di ottenere ed inviare i messaggi della chat e così via.

Ogni servizio ha un interfaccia molto semplice e, nel caso esso esegua delle operazioni asincrone, genera degli eventi a cui altri oggetti devono registrarsi per riceverli.

I manager invece incapsulano i servizi per usarli quando ricevono richieste da parte dei controller. L'unica vera funzione dei manager è quella di separare i servizi dai controller utilizzando il sistema delle notifiche per comunicare con i controller, in modo asincrono.

Controller e viste

Le viste sono quelle classi che disegnano elementi grafici sullo schermo del telefono, siano esse semplici immagini statiche o viste generate dinamicamente

[5]JSON, acronimo di JavaScript Object Notation, è un formato per lo scambio dei dati in applicazioni client-server. `http://it.wikipedia.org/wiki/JSON`

(ad esempio, nella console estesa abbiamo per ogni traccia caricata la forma d'onda che viene generata mediante la lettura del file audio).

In generale, quando si parla di viste, si intende una schermata, ossia una collezione di elementi visivi distinti ma gestiti da uno stesso controller. Ogni controller infatti, gestisce una singola schermata, ossia tutti (se necessario) gli elementi visivi presenti in essa.

Il controller quindi ha la responsabilità di fare da tramite tra le viste e i manager, recuperando informazioni o chiedendo l'aggiornamento di essi dopo che l'utente ha espresso tale richiesta tramite un evento sullo schermo, come la pressione di un pulsante. Allo stesso modo, sarà il controller ad aggiornare la vista qualora ci fossero dati aggiornati da mostrare all'utente, come il cambiamento nello stato del pulsante una volta che un operazione particolare è andata in porto.

4.3 Risultati

I risultati di questo progetto si riassumono nelle esperienze che l'utente può fare con quest'app. Presenterò quindi le principali schermate realizzate e poi le caratteristiche non visibili ma che rendono questa applicazione assolutamente apprezzabile per l'utente.

4.3.1 Schermata principale

La schermata principale è quella visibile all'avvio dell'app. Consente l'accesso alle varie sezioni dell'applicazione tramite dei pulsanti.

Nella parte alta della schermata si trova anche una barra di ricerca tramite la quale è possibile cercare uno show o un utente per nome.

La schermata principale è presentata in figura 4.2.

4.3.2 Sezione profilo utente

Il profilo dell'utente può essere sia mostrato all'utente, elencando le informazioni conosciute, sia modificato.

Nella schermata di visualizzazione si trovano riepilogate informazioni come il nome completo, la descrizione, il luogo in cui vive, la data di nascita e il

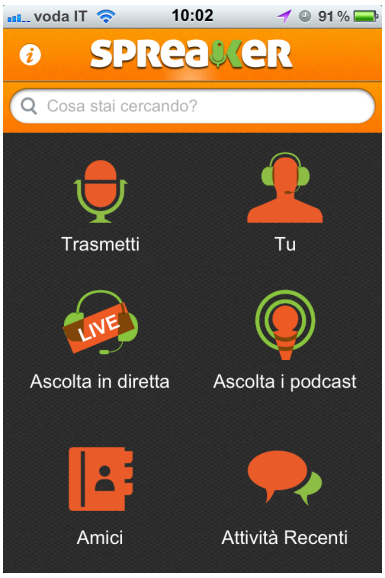

Figura 4.2: Schermata principale dell'app

sesso. È anche presente il tipo di abbonamento (piano premium) che l'utente ha attivato al momento. Selezionando tale voce, è possibile vedere in maggior dettaglio il nome del piano, le limitazioni di questo e il livello di utilizzo che l'utente ha rispetto ai limiti del piano.

La schermata con il profilo dell'utente è presentata in figura 4.3.

Tramite il pulsante *Modifica*, si accede alla schermata per la modifica dei dati del profilo.

Nella schermata di modifica, vi sono elencati tutti i campi possibili per descrivere un utente, con il loro valore attuale. Selezionando uno di questi campi, viene proposta una schermata in cui l'utente può cambiare a proprio piacimento il contenuto, a seconda del tipo di informazione che si sta per modificare. Ad esempio, nel caso della data di nascita, verrà proposta una schermata con un selettore circolare, con cui selezionare giorno, mese ed anno di nascita; nel caso del luogo verrà proposta una schermata nella quale l'utente potrà cercare la propria città usando il GPS oppure un testo; nel caso più semplice del nome o della descrizione, verrà proposta una schermata con solo la tastiera e uno spazio nel quale inserire del testo. È persino possibile cambiare la propria immagine, eseguendo uno scatto sul momento o utilizzando

Figura 4.3: Schermata di riepilogo del profilo dell'utente

le foto presenti nella libreria del dispositivo.

La schermata per la modifica del profilo dell'utente è presentata in figura 4.4.

4.3.3 Ascolto di contenuti in diretta

L'ascolto di contenuti audio in diretta può avvenire da molte schermate diverse, ovunque vi sia un pulsante di play che fa riferimento ad un episodio *LIVE*. Generalmente però l'utente può accedere a questi contenuti tramite la schermata "Ascolta in diretta".

Tali episodi sono ordinati in base a tre modalità diverse:

- Vicino a te
- Più popolari
- Preferiti.

L'utente dovrà solo selezionare la modalità che preferisce toccando una delle voci nella barra segmentata verde in alto. Gli episodi corrispondenti verranno subito richiesti al server e verranno visualizzati sempre come elenco.

Figura 4.4: Schermata per modificare il profilo dell'utente

La schermata descritta è presentata in figura 4.5.

Per iniziare l'ascolto di uno di questi episodi, l'utente dovrà solo toccare l'elemento che vuole ascoltare. L'app mostrerà una nuova schermata dalla quale l'utente vedrà riepilogato cosa sta ascoltando, potrà (se sta ascoltando un episodio in diretta) accedere alla chat per interagire con il dj e con gli altri ascoltatori, e controllare il volume e il playback dell'ascolto.

Tale schermata prende il nome di "In ascolto" è presentata in figura 4.6.

Tramite la pressione del pulsante *Follow* (o *Unfollow*) si aggiunge (o rimuove) l'autore dell'episodio tra quelli preferiti. Premendo invece il pulsante *Chat* ovviamente si accede alla schermata di chat ma solo nel caso si sta ascoltando un episodio in diretta.

La schermata di chat ha, oltre ai messaggi lasciati dagli utenti, ha un campo di testo tramite la quale è possibile inviare il proprio messaggio. Tale schermata è presentata in figura 4.7.

4.3.4 Ascolto di contenuti in podcast

Una volta che i contenuti in diretta vengono terminati, essi diventano disponibili sotto forma di podcast anche dall'app. Similmente alla presentazione

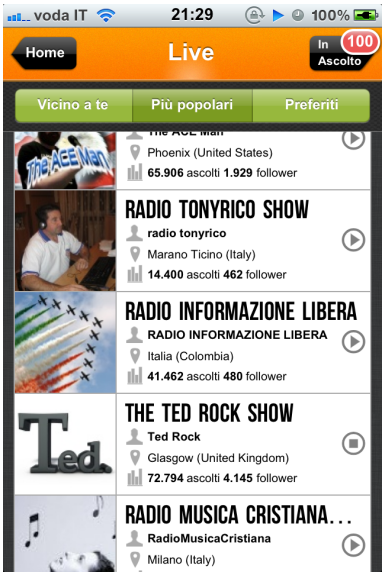

Figura 4.5: Schermata con gli episodi in diretta attualmente

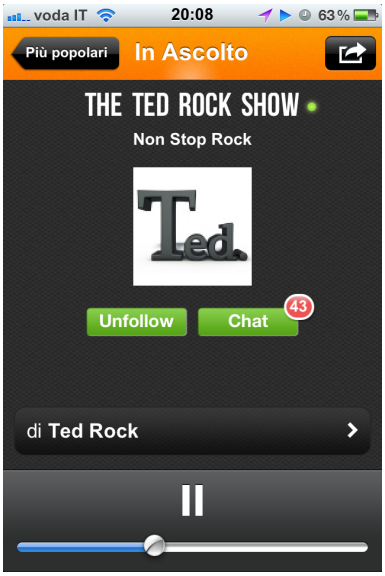

Figura 4.6: Schermata con l'episodio in diretta che l'utente sta ascoltando

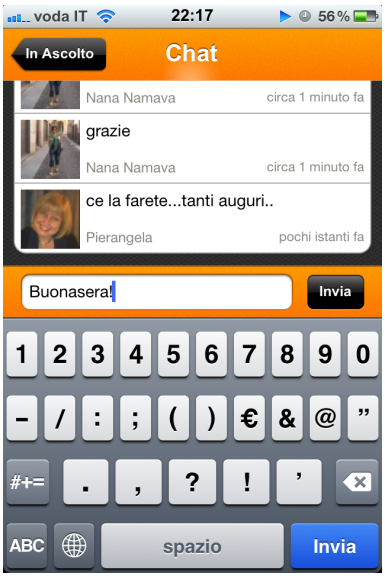

Figura 4.7: Schermata di chat

degli episodi live (episodi trasmessi ed ascoltabili in diretta), vi sono degli elenchi ordinati secondo le stesse tre modalità dai quali l'utente può cercare lo show di suo interesse.

Tale schermata è presentata nella figura 4.8.

Selezionato lo show, può scorrere gli episodi contenuti nello stesso e, toccando il pulsante play a destra dell'episodio, l'app inizierà a riprodurre il contenuto scelto portando l'utente nella schermata "In ascolto". Nella parte superiore vi sono riportate alcune informazioni a riguardo dello show e dell'autore, come gli ascolti conteggiati sugli episodi dello show e sul numero di utenti che seguono l'autore. Anche qui abbiamo il pulsante *Follow / Unfollow* per aggiungere/rimuovere l'autore dalla cerchia dei propri utenti da seguire.

La schermata dello show, con gli episodi contenuti è presentata in figura 4.9.

4.3.5 Trasmissione in diretta

La trasmissione in diretta è resa possibile da tue tipi di console diverse. La prima, più semplice e limitata, è stata realizzata per consentire una trasmissione di reportage, ove non vi è necessità di mettere musiche o effetti ma

Figura 4.8: Schermata con i podcast disponibili

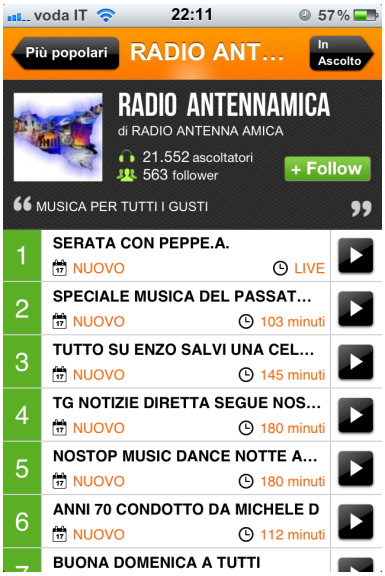

Figura 4.9: Schermata con i gli episodi di uno show

si vuole solo registrare dal microfono senza altri fronzoli. La prima console infatti ha solamente un pulsante: registra.

Alla pressione del pulsante di registrazione, viene chiesto all'utente se vuole trasmettere in diretta o se preferisce registrare localmente sul proprio telefono per pubblicare la registrazione in un secondo momento. In entrambi i casi, l'utente ritorna alla console, dove può registrare tenendo d'occhio il volume della sua registrazione, il tempo che ha a disposizione e lo stato della connessione tra l'app e il server di registrazione.

Tale indicatore risulta molto utile per indicare all'utente se, durante la trasmissione in diretta, si ritrova con una velocità di banda insufficiente per trasmettere il proprio episodio. Di fatto una cattiva connessione comporta frequenti momenti di silenzio per gli ascoltatori e dunque una pessima esperienza complessiva.

In più, nel caso decida di trasmettere in diretta, avrà a disposizione anche la chat, con le stesse modalità previste per l'ascolto di un episodio in diretta. Nella console vedrà solamente l'ultimo messaggio ricevuto ma toccandolo, accederà alla schermata della chat, dove potrà leggere tutti i messaggi o scriverne uno.

La schermata della console base è presentata in figura 4.10.

La console però di maggiore impatto è la console evoluta, ossia quella in cui è possibile registrare la propria voce assieme a musica ed effetti sonori. L'interazione con l'utente è forzatamente complessa dato il numero di funzioni messe a disposizione.

Nella parte alta della schermata, oltre al pulsante per l'avvio e l'interruzione della registrazione, abbiamo i controlli per le due tracce audio. Essi sono un pulsante per la selezione del brano da caricare nella traccia ed il pulsante per far partire o fermare la traccia stessa. Troviamo poi una barra di avanzamento, nel quale sullo sfondo viene riprodotta abbastanza fedelmente la forma d'onda del brano scelto mentre il cursore è liberamente trascinabile per consentire di far partire la canzone dal punto che si preferisce.

Nella parte in basso a sinistra abbiamo invece una tastiera a tre pulsanti, per lanciare e fermare tre effetti, selezionabili tramite l'apposito pulsante a sinistra. Gli effetti sono tecnicamente dei piccoli file audio caricati assieme all'app e quindi non possono provenire dalla libreria musicale del dispositivo,

Figura 4.10: Schermata della console base durante la registrazione di un podcast

come invece è per le tracce. Di fatto questa è una scelta fatta per consentire di introdurre altri effetti nel futuro come caratteristica a pagamento.

Al centro dello schermo troviamo il selettore per spegnere o accendere il microfono e un selettore per bilanciare il volume tra le due tracce audio.

A destra troviamo una serie di informazioni di controllo, ossia il tempo di registrazione trascorso e quello ancora a disposizione, i livelli dell'audio del microfono e dell'uscita del registratore e, come nella console base, un indicazione della qualità della rete.

Nella parte inferiore della console, durante le trasmissioni in diretta, troviamo un anteprima della chat con le stesse modalità previste per la console base.

La schermata della console evoluta è presentata in figura 4.11.

Entrambe le console hanno la possibilità sia di trasmettere in diretta che di registrare in locale nel telefono. Nel caso in cui il dispositivo non abbia una connessione ad internet disponibile, la trasmissione in diretta non è possibile, mentre le registrazioni in locale si. La pubblicazione dell'episodio necessita della connessione ma questa può avvenire in qualsiasi momento successivo.

Figura 4.11: Schermata della console completa durante una trasmissione in diretta

4.3.6 Caratteristiche comuni

Sia durante l'ascolto di un episodio che durante la trasmissione, è possibile condividere l'episodio corrente. Nella schermata di ascolto vi è un pulsante con una freccia in alto a destra, nella barra superiore. Nella console vi è un esplicito pulsante *Condividi* sempre in alto a destra. Alla pressione di tali pulsanti, viene proposto un piccolo menu con le possibili destinazioni verso le quali condividere l'episodio. Tali destinazioni, al momento sono Twitter, Facebook ed email.

Per rendere semplice il controllo della riproduzione di ciò che si sta ascoltando, quasi in ogni schermata viene mostrato il pulsante *In ascolto* in alto a destra dello schermo, con il quale l'utente può immediatamente aprire la schermata *In Ascolto* per regolare il volume, fermare la riproduzione o vedere i nuovi messaggi arrivati nella chat.

4.3.7 Caratteristiche non grafiche

Uno dei problemi maggiori che si incontrano nello sviluppare per dispositivi mobili è l'instabilità della connessione ad Internet. Essa non si può mai dare per scontata, neanche quando sembra esserci. Risulta quindi necessario partire dal presupposto che ogni richiesta eseguita attraverso la rete fallirà ed organizzare l'architettura del codice di conseguenza.

Durante l'ascolto e la trasmissione, questo problema è veramente critico perchè condiziona fortemente l'esperienza d'uso per l'utente e quindi ha richiesto uno sforzo maggiore per ridurre al minimo i problemi. È stata implementata la riconnessione automatica verso il server sia durante una trasmissione di un episodio sia durante l'ascolto. Nel momento in cui la connessione risulta in stallo per diverso tempo o quando si riceve un errore di rete, essa si attiva per ripristinare il collegamento.

Questa tecnica ha un'alta efficacia nell'ascolto, per via del buffering che l'app fa tra quello che riceve dal server e quello che effettivamente riproduce. Durante la trasmissione essa è meno efficace ma comunque apprezzabile, almeno per la mancanza di intervento da parte dell'utente che può quindi rimanere concentrato nella creazione di episodi.

Capitolo 5

Conclusioni

Spreaker Radio, nella versione attualmente rilasciata nell'AppStore ha raggiunto tutti gli obbiettivi previsti all'inizio di questo progetto, sia quelli principali che quelli secondari e risulta essere apprezzata dagli utenti che la usano.

5.1 Obiettivi raggiunti

L'app è in grado di far ascoltare tutti i contenuti presenti in Spreaker, siano essi live o podcast. In più, durante l'ascolto di un episodio live, viene messo a disposizione il servizio di chat.

L'utente può cercare un utente o uno show particolare tramite la funzionalità di ricerca integrata nella pagina principale. Può anche scegliere di ascoltare gli episodi e gli show più popolari, quelli vicini a lui o i suoi preferiti, tramite le schermate *Ascolta in diretta* ed *Ascolta i podcast*.

L'app è in grado di far trasmettere in diretta e registrare localmente da ovunque l'utente sia. Esso può scegliere se usare la console semplice o quella completa, usare la propria voce e la musica che ha nel proprio dispositivo, per condividere con chiunque nel mondo la sua ispirazione del momento. Nel caso di episodi registrati solamente nel telefono, può pubblicarli in uno dei suoi show in qualsiasi momento, sia su Spreaker che successivamente su SoundCloud.

L'app è in grado di visualizzare il profilo dell'utente e modificarne sia i dati sia le impostazioni, come gli account collegati dei socialnetwork dell'utente. Può visualizzare gli show dell'utente e anche i singoli episodi, nonchè modificarne le informazioni o eliminarli del tutto.

Obiettivi secondari

L'app è molto integrata con Facebook, tanto che l'utente può eseguire il login nell'app tramite il proprio account Facebook e può condividere su di esso i propri episodi e show. Può anche invitare i propri amici a provare Spreaker (sia l'app che il servizio), tramite Facebook e via SMS. In più, usando l'app di Facebook, se si seleziona una risorsa di Spreaker pubblicata da qualcuno, Facebook richiederà alla nostra app di aprirsi e visualizzare tale risorsa, sia essa un episodio, uno show o un utente.

Infine, grazie all'aiuto di due colleghe, l'app è stata tradotta in tre lingue: italiano, inglese (americano) e spagnolo.

Un pò di numeri

In questi sei mesi di sviluppo, con rilasci abbastanza regolari in AppStore, l'app è stata scaricata oltre 50 mila volte in tutto il mondo ed è nella classifica TOP 50 della categoria musica in italia[1].

È stata utilizzata oltre 1 milione di volte, nelle quali sono stati ascoltati quasi 350 mila episodi e ne sono stati prodotti oltre 70 mila.

Il paese che più utilizza quest'app sono gli Stati Uniti (col 35%) mentre il secondo paese è l'italia (col 20%).

5.2 Prospettive per il futuro

Con le caratteristiche chiave implementate, gli ulteriori sviluppi possono essere di due tipi: rifinire l'app con piccole migliorie oppure aggiungere nuove caratteristiche chiave.

[1] Dato del 3 ottobre 2012, fonte APPlyzer `http://applyzer.com`.

Tra i piccoli sviluppi troviamo, ad esempio, il miglioramento della ricerca dei brani dalla libreria musicale per velocizzarne l'accesso e la gestione da parte dell'utente o l'aggiunta di altri effetti sonori utilizzabili nella console.

Tra i macro sviluppi, sentendo anche il parere degli utenti, la cosa più richiesta è la realizzazione di un app per iPad, cosa che comporterebbe *solamente* una revisione della parte grafica.

In ogni caso, ogni sviluppo successivo deve portare sia benefici per l'utente sia stabilità all'app, per non intaccare l'utenza che la usa anche come strumento di lavoro e non solo per piacere.

Infine un commento personale.

L'esperienza maturata in questi mesi di lavoro, mi ha lasciato un segno profondo. Lavorare in una startup, in un periodo economico particolarmente critico come quello attuale, mi hanno reso una persona più determinata e consapevole delle mie debolezze e dei miei punti di forza. Mi ha convinto ancor di più che il mercato mobile abbia ancora ampi margini di crescita e che, puntando su di esso, le chance di lavoro non mancheranno.

Elenco delle figure

Appendice A

Objective-C

Objective-C è un linguaggio orientato agli oggetti costruito come estensione del C (ne mantiene la compatibilità e consente ad un compilatore Objective-C di saper compilare anche codice C) ed ispirato da Smalltalk. È dinamico, non fortemente tipizzato e si basa sullo scambio di messaggi tra oggetti. Altre peculiarità del linguaggio sono i protocolli e le categorie.

Passaggio di messaggi

Comunemente, nei linguaggi ad oggetti, due oggetti interagiscono tra di loro come un oggetto che invoca un metodo su un altro oggetto, con un forte controllo sul metodo e sui tipi dei parametri.

In C++ ad esempio, l'esecuzione di un metodo un altro oggetto avviene con il seguente codice:

```
obj->doSomething(argument);
```

In Objective-C invece, un oggetto invia un messaggio ad un altro e solo a runtime viene risolto il tipo dell'oggetto destinatario e il messaggio inviato, senza garanzia che esso venga compreso dal ricevente.

In Objective-C invece, il codice sarebbe il seguente:

```
[obj doSomething:argument];
```

Tipizzazione dinamica

La tipizzazione dinamica (o debole) comporta la mancanza di controllo formale qualora si cerchi di inviare un messaggio ad un oggetto che non dichiara di saper rispondere a tale messaggio.

Questo consente quello che viene definito come inoltro, ossia l'oggetto può catturare il messaggio a cui non sa rispondere e inoltrarlo ad un altro. Se invece tale messaggio non viene inoltrato a nessuno, viene sollevato un errore di run-time.

Protocolli

I protocolli vengono usati per introdurre nel linguaggio il concetto di ereditarietà multipla. Ogni protocollo può forzare la classe ad implementare la gestione di un particolare messaggio oppure lasciarlo opzionale.

Categorie

Le categorie sono semplicemente un modo per organizzare il codice di un oggetto, suddividendone i metodi per scopo, o appunto categoria.

Bibliografia

[1] **Spreaker**: *Spreaker for Developers*, `http://developers.spreaker.com`.

[2] **Leonard Richardson, Sam Ruby**: *RESTful Web Services*, O'Reilly, 2007.

[3] **Apple**: *iOS Developer Library*, `http://developer.apple.com/library/ios`.

[4] **Facebook**: *Facebook SDK for iOS*, `https://developers.facebook.com/ios`.