Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

Realizzazione di un'applicazione mobile multi piattaforma per il dispacciamento di contenuti multimediali

Relatore:
Chiar.mo Prof.
Vittorio Ghini

Corelatore:
Dott. Emanuele Preda

Presentata da:
Riccardo Serafini

Sessione I
Anno Accademico
2013/ 2014

“Non hai veramente capito qualcosa

fino a quando non sei in grado di spiegarlo a tua nonna.”

- Albert Einstein

Indice

Capitolo 1: Introduzione

Il mondo della comunicazione è stato rivoluzionato nell'ultimo decennio, dall'avvento di *smartphones* e *tablet*, dispositivi di ridotte dimensioni in grado di essere connessi costantemente alla rete Internet.

Oltre ad innovare la società dal punto di vista dei paradigmi di comunicazione, introducendo nuovi modi per scambiare informazioni, dal punto di vista economico, spalancando le porte di un nuovo mercato di prodotti e applicazioni, la diffusione di questi dispositivi ha introdotto nuove sfide e problemi nel campo informatico, alcuni dei quali ancora non risolti.

Su questi presupposti si basa il mio lavoro di tesi, il cui scopo era la realizzazione di un'applicazione mobile sviluppata all'interno dell'azienda SMS ITALIA, per la distribuzione di contenuti pubblicitari e multimediali innovativi ed interattivi e rivolta ad un ampio numero di utenti.

Il resto del documento è strutturato in questa maniera: dopo una prima parte compilativa in cui introdurremo la storia e la situazione attuale del mondo dei dispositivi mobili, verrà descritta l'analisi dei requisiti da cui è nata l'esigenza di realizzare l'applicazione, verranno analizzati i problemi riscontrati durante la sua realizzazione, le soluzioni adottate e verrano motivate le scelte implementative.

Capitolo 2: La breve storia degli smartphones: dispositivi di comunicazione mobile “intelligenti”

Con il termine “*smartphone”* intendiamo comunemente un dispositivo di calcolo portatile di ridotte dimensioni, la cui funzione principale è permettere la comunicazione telefonica, ma che mette a disposizione dell’utente uno spettro di caratteristiche “intelligenti”, tradizionalmente fruibili attraverso personal computer.

Il termine comparve per la prima volta con il lancio dell GS 88 "Penelope" da parte dell’azienda *Ericsson.*

Nei primi anni del 2000 il mercato degli smartphone era occupato quasi interamente da BlackBerry, con una linea di dispositivi rivolti principalmente a dirigenti aziendali, dotati di una connessione ad Internet, utilizzata quasi esclusivamente per scaricare le email lavorative.

Oltre alla capacità di collegarsi alla rete per leggere la posta e poter visualizzare pagine web in modo primitivo, questi dispositivi erano molto meno “*smart”* di quelli che siamo abituati ad utilizzare oggi.

Una piccola svolta nel mercato è avvenuta dal 2005 da parte dell’azienda finlandese *Nokia*, con il lancio della serie N, una gamma che ha riscosso molto successo, grazie alla solidità del proprio hardware e all’efficacia e stabilità del sistema operativo Symbian.
Pur nelle sue limitazioni, Nokia riusci a venderne molte copie, rivolgendosi al segmento *consumer*, non più solo a quello *business*.

Nel 2007 Apple ha rivoluzionato il mondo dei dispositivi mobili con l'introduzione dell'iPhone.
Le novità furono presentate da Steve Jobs durante il tradizionale Keynote[1] annuale dell'azienda.
Con questo prodotto, l'azienda è riuscita a ridefinire il concetto di *smartphone* , con un dispositivo che presentava caratteristiche rivoluzionare rispetto ai concorrenti.

Le novità tecniche introdotte da Apple furono notevoli. Non tutte queste sono state ideate da zero dall'azienda, ma iPhone ha il pregio di averle racchiuse tutte insieme in modo equilibrato ed intuitivo, introducendo uno standard de facto che non è stato superato dopo 7 anni e che presenta queste caratteristiche:

- sistema operativo "moderno". La novità forse più importante, e sicuramente quella più rilevante dal punto di vista informatico, è quella dell'utilizzo di un sistema operativo moderno, con caratteristiche del tutto comparabili a quelli utilizzati nei tradizionali computer. La possibilità di realizzare ed installare applicazioni di terze parti che possono accedere a tutte le funzionalità hardware e del sistema operativo tramite un SDK, la possibilità di eseguire applicazioni in multitasking permettono a questo tipo di dispositivi di realizzare funzionalità avanzate, dal punto di vista della comunicazione e non solo.
- un ecosistema di applicazioni di terze parti scaricabili ed installabili attraverso un'unica applicazione integrata (di solito chiamata Market o Store).
  Questo sistema mette a disposizione dell'utente una miriade di funzionalità aggiuntive, attraverso programmi realizzati da terze parti, che generano anche un enorme mercato tramite le applicazioni a pagamento.

[1] www.youtube.com/watch?v=t4OEsI0Sc_s

I realizzatori dell'applicazione sono costretti a seguire delle rigide specifiche imposte dal sistema operativo, attraverso un kit di sviluppo appositamente realizzato (SDK), e una politica di accettazione delle app nello store che permette di filtrare quelle indesiderate, instabili o non di sufficiente qualità, pericolose o malevole.

- formato "*slate*", cioè un dispositivo rettangolare e piatto, con un ridotto spessore e peso limitato, con un profilo uniforme e pochi tasti fisici.
- superficie anteriore completamente occupata da un touch-screen capacitivo in grado di rilevare anche tocchi multipli. Questo ha permesso di definire un nuovo metodo di input, quello delle "*gesture*": movimenti delle dita sullo schermo che imitano movimenti naturali, sono pertanto estremamente semplici da imparare e da ricordare, e consentono azioni prima inimmaginabili in un device di dimensioni così ridotte.
- rimozione della tastiera hardware in favore di quella software. Nei tradizionali dispositivi di tipo *candybar*, metà della superficie anteriore era occupata dalla tastiera QWERTY con bottoni fisici. Rimuovere la tastiera ha permesso di guadagnare molto spazio per lo schermo e per la presentazione dei contenuti, ma ha anche importato un nuovo concetto proveniente dalle interfacce grafiche desktop tradizionali: l'interfaccia di input varia a seconda dell'applicazione in esecuzione.

  Non tutte le applicazioni, infatti, hanno bisogno di una tastiera sempre presente, come altre non hanno bisogno dei bottoni della tastiera QWERTY ma di altri tipi di pulsanti. Con il dispositivo di input integrato nello schermo, il sistema operativo presenta all'utente l'interfaccia grafica più adatta ad ogni situazione, ottimizzando i tempi e la praticità d'uso del dispositivo.
- connettività avanzata. L'introduzione di antenne per la connettività GSM/EDGE/UMTS per le reti ad alta velocità, supporto per WiFi e Bluetooth

ha permesso di essere collegati ovunque, di scambiare dati e di interfacciarsi con un ampio numero di dispositivi.

- sensori d'ambiente. L'integrazione di un ampio numero di sensori d'ambiente ha permesso la realizzazione di applicazioni che sfruttano l'ambiente esterno per ricevere input ed elaborarli. Accedendo ai dati di accelerometro, giroscopio, sensore di temperatura, bussola magnetica, sensore di luminosità, sensore di prossimità, GPS e GLONASS, le applicazioni sono in grado di fornire un'esperienza di utlizzo "consapevole" del mondo esterno.
- fotocamera. L'introduzione di una fotocamera ad alta definizione ha permesso la realizzazione di foto e video di alta qualità, finendo per rimpiazzare l'utilità delle macchine fotografiche compatte e permettendo la realizzazione di applicazioni che sfruttano la fotocamera: ad es. per effettuare videochiamate, ritocchi o elaborazioni, condivisione di immagini e video, ecc.

Sulla scia del successo di iPhone, è nata un'altra piattaforma che attualmente contende il mercato con iOS: Android.

Il progetto Android nasce nel 2003 da una start-up denominata Android inc. che si pone l'obiettivo di realizzare "*dispositivi cellulari più consapevoli della posizione e delle preferenze del loro proprietario"*.

Nel 2005 la società è stata acquistata da Google, un investimento in vista del lancio nel mercato degli smartphone del colosso di Montain View.

Adeguandosi allo standard definito dalla Apple, il progetto Android ha adottato strategie differenti, puntando sulla distribuzione e la concessione del sistema operativo ad aziende terze, piuttosto che alla realizzazione di hardware proprio.

Questo, nel tempo ha portato alla definizione di un “ecosistema” di smartphone Android, composto da differenti produttori, differenti versioni e personalizzazioni del sistema operativo.

Se da una parte questa scelta ha portato alla vendita di un numero incredibile di dispositivi, questa frammentazione ha impedito una forte ottimizzazione del sistema operativo sull’hardware, conducendo a performance in alcuni casi scadenti, soprattutto su dispositivi di fascia bassa.

Negli ultimi anni una svolta significativa è stata apportata da Microsoft e Nokia, che nel 2011, dopo la stipulazione di un accordo commerciale, hanno lanciato la linea di smartphone prodotti da Nokia con sistema operativo Windows Phone, denominata Nokia Lumia.

Grazie alla rinomata qualità dell’hardware Nokia e alla diffusione della piattaforma Windows, soprattutto in ambito aziendale, Lumia è riuscita a posizionarsi in una posizione minoritaria ma stabile nel mercato, guadagnando quote soprattutto in ambito business, sottraendo quote a BlackBerry.

Capitolo 3: Sfide e problemi dei sistemi operativi mobili

La nascita di smartphone e tablet ha portato una ventata di novità nel mondo dell'informatica.
I sistemi di elaborazione mobili sono caratterizzati da particolari esigenze, limiti e caratteristiche, alcuni già affrontati in passato, altri inediti al mondo dello sviluppo software.

3.1 Batteria

I dispositivi mobili, a differenza dei computer fissi, sono alimentati da una batteria. Benchè la tecnologia delle batterie negli ultimi anni sia migliorata notevolmente per quanto riguarda la densità di carica e l'autonomia, questa rappresenta la maggiore criticità dei sistemi mobili attuali.

Questo problema è stato già affrontato, in informatica, con l'avvento dei *laptop*, ma si è riproposto in modo ancora più radicale e problematico.

Con una batteria in grado di fornire una quantità molto limitata di alimentazione al sistema prima di esaurirsi e necessitare una ricarica, la progettazione delle applicazioni e del sistema operativo deve considerare in modo molto oculato la distribuzione e l'utilizzo di risorse.

L'energia, inoltre, non può essere considerata come una risorsa che si consuma linearmente, allo stesso ritmo con cui viene richiesta.

L’utilizzo simultaneo delle periferiche, antenne e sensori, comporta un riscaldamento dei componenti elettronici, e l’efficienza delle moderne batterie Li-Ion risente molto della variazione di temperatura.
Riscaldandosi, la batteria diminuisce la sua efficienza, scaricandosi più velocemente e ricaricandosi più lentamente.

L’apparato che consuma maggiore energia è sicuramente lo schermo, a causa della luce della retroilluminazione che permette di illuminare i pixel e mostrare colori vivi e nitidi.

A seguire, il processore è in larga misura responsabile del consumo di energia, soprattutto se utilizzato con la massima frequenza di clock prevista dall’architettura.

Le antenne e i sensori, se utilizzati in modo intensivo, possono contribuire notevolmente all’assorbimento, soprattutto il modulo dati che permette di collegarsi ad Internet tramite la connessione mobile.

Gli sviluppatori e progettisti dei sistemi operativi hanno affrontato il problema integrando nel kernel un apposito gestore dell’energia.

Il sistema operativo si preoccupa che in ogni momento venga utilizzata la quantità minima di energia sufficiente a fornire un’esperienza utente soddisfacente.
Questa limitazione avviene su più livelli, sia dal punto di vista del dispositivo, nel sistema operativo e nelle applicazioni, sia dal punto di vista dell’utente.

Il kernel modifica costantemente la frequenza di clock del processore, mantenendola al minimo nei momenti di inattività ed aumentandola quando i processi richiedono maggiore velocità computazionale.

Lo schermo viene spento dopo un periodo di inattività per evitare di sprecare risorse inutilmente.

Il sistema operativo però non può garantire in ogni caso un consumo ottimale della batteria, in quanto gli utilizzatori finali delle risorse sono le applicazioni, che richiedono potenza computazionale al bisogno.

Il kernel può porre dei limiti alle applicazioni nel consumo di risorse, terminando o sospendendo i processi che superano la soglia imposta, ma una grossa responsabilità per l'efficienza energetica del dispositivo spetta agli sviluppatori delle applicazioni.

Una oculata richiesta di risorse computazionali deve essere sempre tra le prorità di chi progetta e sviluppa una app.

Un'applicazione che consuma molta batteria, avrà necessariamente meno successo di una concorrente che gestisce in modo ottimale le risorse, soprattutto se questa viene resa disponibile a pagamento.

I realizzatori dei sistemi operativi mobili, inoltre, hanno affrontato il problema mettendo a disposizione dell'utente degli strumenti per controllare in ogni momento il consumo di corrente del proprio dispositivo.

Nella maggior parte dei casi questo si traduce in interruttori realizzati nell'interfaccia grafica che permettono di "spegnere" ed "accendere" dinamicamente le varie periferiche, lasciando abilitate solo quelle che di cui si ha effettivamente bisogno.

Molti sistemi inoltre permettono all'utente di abilitare una modalità di "risparmio energetico" che attua una serie di politiche drastiche per garantire una maggiore autonomia della batteria.

3.2 Risorse computazionali

Benchè la miniaturizzazione dei processi produttivi di processori e componenti elettronici sia arrivata a risultati incredibili, i dispositivi mobili dispongono di risorse computazionali molto limitate rispetto ai tradizionali computer.

La frequenza di clock è limitata per i motivi precedentemente illustrati, la quantità di memoria centrale non può superare un certo limite per problemi di spazio, e anche la velocità di scrittura della memoria secondaria, costituita prevalentemente da memorie flash, è molto ridotta.

Per questo motivo, l'efficienza algoritmica in spazio e tempo dei programmi destinati ad essere eseguiti su questo tipo di dispositivi diventa cruciale.
Non ci si può permettere di eseguire codice inefficiente e nei casi in cui una computazione richieda una modesta quantità di risorse, il mondo dell'informatica ha adottato diversi stratagemmi per mantenere un'esperienza utente nella norma.

Le interfacce grafiche si sono evolute con animazioni di caricamento sempre più coinvolgenti ed accattivanti, per evitare di indurre nell'utilizzatore che l'elaboratore si

sia bloccato o abbia generato un errore, ma che stia effettivamente calcolando il risultato atteso.

Spesso le computazioni non vengono eseguite direttamente dal dispositivo mobile, ma vengono inviati ad un server remoto i dati di input e viene scaricato il risultato atteso attraverso la rete, per poi essere presentato all'utente.

3.3 Comunicazione

Pur essendo la comunicazione lo scopo della maggioranza delle funzionalità degli smartphone, questi dispositivi spesso non dispongono di un'ampiezza di banda sufficiente per realizzare in modo rapido i compiti a loro assegnati.

I moderni cellulari vengono dotati di diverse antenne che consentono loro di comunicare in un ampia varietà di situazioni.

Il tipo di connessione più importante è sicuramente quello dei dati mobili, realizzato attraverso l'aggancio a torri della rete cellulare, con l'ausilio di una scheda SIM di un determinato operatore.

Durante gli anni si sono succedute diverse generazioni di connessioni mobili su rete cellulare.

Nella prima generazione di GSM la connessione avveniva a commutazione di circuito, con lo stesso modello dei "modem" su linea telefonica fissa. Il sistema si è rapidamente evoluto introducendo con il GPRS la commutazione di pacchetto, e con l'estensione EDGE installata alcuni anni più tardi da alcuni Operatori, in grado di

moltiplicare notevolmente la larghezza di banda disponibile in ricezione.
L'avvento della rete di terza generazione UMTS, ribattezzata per semplicità 3G, con le sue varianti HSDPA e HSPA+ in alcuni Paesi, hanno permesso il raggiungimento effettivo di un'ampiezza di banda di alcuni Mbit, paragonabile a quella disponibile nelle utenze domestiche ADSL.
La diffusione del recente standard di quarta generazione, LTE, garantirà prestazioni paragonabili a connessioni in fibra ottica (fino a 100Mbit).

Le connessioni mobili, tuttavia, presentano diversi problemi che le rende non completamente affidabili.
Oltre all'elevato consumo di batteria, già descritto nel paragrafo precedente, la copertura di rete a banda larga non copre adeguatamente tutto il territorio e, in luoghi chiusi o dove sono presenti intelaiature metalliche può essere addirittura assente.
Visto il costo delle connessioni mobili, inoltre, non tutti i possessori di smartphone sottoscrivono un contratto per usufruire di questa tecnologia.
Questo tipo di connessione è caratterizzato da una latenza doppia rispetto a quella della broadband fissa e da larghezze di banda inferiori e soprattutto variabili nel tempo a seconda della qualità del segnale.

Oltre che alle reti cellulari, i dispositivi mobili si affidano anche alle reti Wi-Fi.
Fin dai primi modelli, gli smartphone hanno montato antenne in grado di collegarsi alle reti WLAN, e queste antenne hanno seguito l'evoluzione di tale tecnologia.
Dalle prime reti di classe **b** con una velocità di trasmissione di 11 Mb/s, la tecnologia ha permesso di raggiungere velocità di 54 Mb/s con le reti di classe **g** e successivamente la velocità di 450 Mb/s con le reti di classe **n**.

Le reti Wi-Fi permettono ai dispositivi di ottenere una capacità di accesso alla rete pari a quella dei tradizionali computer domestici, ma spesso in mobilità, cioè la situazione prevalente di impiego di questi apparati, le reti Wi-Fi non sono disponibili, oppure sono protette da una chiave di sicurezza.

La periferica Wi-Fi, inoltre, viene spesso utilizzata per produrre reti ad-hoc, per condividere con altri dispositivi mobili la connessione fornita dalla rete cellulare.

Oltre a questi due tipi di antenne, gli smartphone integrano anche altri meccanismi di comunicazione senza fili, meno utilizzati ma importanti per scopi differenti.

Il Bluetooth, standard che ha raggiunto recentemente la versione 4.0, viene impiegato soprattutto per collegare i dispositivi mobili tra loro, per lo scambio di file, contatti e rubrica, oppure per agganciarsi ad altre periferiche mobili, come sistemi vivavoce per auto, auricolari, altoparlanti ed altro.
Questa tecnologia, non permette di solito il collegamento ad Internet, ha un limitato raggio di azione ed un'ampiezza di banda minima.

Ultimamente è stata introdotta anche un'altra modalità di comunicazione senza fili: il Near Field Communication (NFC).
Questa tecnologia permette una comunicazione tra dispositivi alla distanza di un paio di centimetri (praticamente in contatto) con una banda molto ridotta.
Si sta diffondendo sempre di più per realizzare soprattutto modalità di pagamento e sostituzione delle carte magnetiche dotate di RFID e smart-card che di solito teniamo nel portafogli.

Verrà utilizzato per pagare al posto della carta di credito, come modalità di autenticazione per l'accesso a zone riservate e per pagare il biglietto nei trasporti pubblici.

Pur disponendo di tutte queste possibilità di comunicazione senza fili, i moderni cellulari non dispongono spesso di una connessione ad Internet di buona qualità, e questa caratteristica deve essere considerata nella realizzazione di applicazioni mobili.

La scienza informatica ha cercato di arginare questo problema limitando la richiesta di scambio dati attraverso la rete dove possibile.
Visto che l'accesso alla rete da parte di più processi contemporaneamente porta inevitabilmente alla riduzione della banda disponibile ad ogni applicazione, di solito viene consentito l'accesso al programma attualmente in primo piano, lasciando ad alcuni appositi servizi in background il controllo periodico di nuovi messaggi, email o dati disponibili, avvisando l'utente attraverso delle notifiche che permettono di portare in foreground l'applicazione interessata.

La compressione dati assume un ruolo molto importante e viene utilizzata in ogni situazione possibile.

L'utilizzo di banda può essere anche ottimizzato durante la progettazione del ciclo di utilizzo medio dell'applicazione.

Se la app richiede risorse che occupano molto spazio per svolgere le sue funzionalità, meglio integrare questi file nel pacchetto che viene scaricato nel momento in cui questa viene installata.

Questo comportamento è preferibile al far scaricare una quantità ingente di dati al primo avvio.

Allo stesso modo, si può scegliere di effettuare download e upload pesanti ma non urgenti solo quando il dispositivo è connesso ad una rete Wi-Fi.

3.4 Piattaforma di sviluppo

Nei tradizionali sistemi software destinati ai computer, i sistemi operativi lasciano molta libertà di scelta sul linguaggio e la piattaforma da utilizzare per poter sviluppare un'applicazione. Pur fornendo solitamente una serie di API e System Call per accedere alle funzionalità base del sistema, gli sviluppatori di applicazioni desktop, e anche web, hanno a disposizione una miriade di linguaggi, tecnologie e framework tra cui poter scegliere per implementare la funzionalità desiderata.

Nei sistemi operativi mobili, pur essendo molto importante la possibilità di realizzare e scaricare applicazioni di terze parti che estendono le capacità del dispositivo, è stato deciso di ridurre notevolmente questa libertà implementativa, imponendo agli sviluppatori una serie di vincoli e barriere oltre la quale l'applicazione non può espandersi.

Solitamente i produttori del sistema operativo rilasciano un kit di sviluppo (SDK), contenente un ambiente di sviluppo pre-configurato necessario a realizzare app, e una documentazione sulle API e librerie da utilizzare per interagire con il sistema e visualizzare elementi grafici.

Nella maggior parte dei casi queste librerie di sviluppo sono limitate ad un solo linguaggio di programmazione, lo stesso con cui viene realizzato il livello di astrazione più elevato del sistema operativo.

Se da un lato questa caratteristica uniforma notevolmente le procedure e i metodi di sviluppo di applicazioni per la stessa piattaforma, dall'altra impedisce agli sviluppatori di adottare la migliore soluzione e tecnologia necessaria ad ogni situazione.

A peggiorare la situazione segnaliamo il fatto che le piattaforme e le tecnologie di sviluppo sui maggiori sistemi operativi mobili, si sono evolute parallelamente ma senza nessuna compatibilità, per cui uno sviluppatore che deve migrare la sua applicazione da una piattaforma ad un'altra, è costretto a riscriverla praticamente da zero, dovendo apprendere metodi, schemi e regole di sviluppo di un sistema completamente nuovo e differente dal precedente.

Per questo motivo il mondo degli sviluppatori per applicazioni mobili è considerato una "nicchia" di persone che sono specializzate in una piuttosto che l'altra piattaforma, generando grossi problemi di interoperabilità.

Capitolo 4: Situazione attuale dei sistemi operativi mobili

Il primo passo che ha portato alla scelta delle piattaforme e degli strumenti da impiegare per realizzare l'applicazione in SMS ITALIA, è stato quello dell'analisi del mercato e dei sistemi operativi mobili più diffusi attualmente.
Sono state prese in considerazione le loro caratteristiche tecniche, economiche e sociali, per individuare la migliore soluzione da impiegare.
Di seguito presenteremo il risultato di questa analisi, mostrando le piattaforme esistenti al momento della ricerca.

4.1 iOS: iPhone e iPad

4.1.1 Storia e mercato

Dal lancio del primo iPhone (detto anche iPhone EDGE o 2G) nel 2007, la Apple ha rilasciato una vasta gamma di dispositivi.
La linea iPhone si è evoluta con iPhone 3G (2008), iPhone 3GS (2009), iPhone 4 (2010), iPhone 4S (2011), iPhone 5 (2012), iPhone 5S e iPhone 5C (2013).
Oltre che alla linea di telefoni, l'azienda californiana ha presentato anche altri prodotti legati dalla stessa piattaforma.
Sulla scia dell'iPhone, la linea di riproduttori musicali si è evoluta nel 2007 con l'iPod Touch, seguito dall'iPod Touch 2G (2008), iPod Touch 3G (2009), iPod Touch 4G (2010), iPod Touch 5G (2012).

Nel 2010 è stata introdotta anche una linea di tablet, dispositivi con le stesse caratteristiche di uno smartphone, ma con uno schermo notevolmente più grande (7 - 10 pollici), una batteria maggiorata e spesso senza la possibilità di effettuare telefonate.

Il primo dispositivo di questa linea è stato iPad, lanciato nel 2010, seguito da iPad 2 nel 2011, iPad mini nel 2012, iPad 3 e 4 nel 2012 e iPad Air nel 2013.

Tutti questi dispositivi condividono l'architettura hardware con processore RISC ARM appositamente progettato.

Il sistema operativo utilizzato da questi dispositivi è denominato iOS, una derivazione di Mac OS X, utilizzato nei sistemi desktop della serie Mac.

Si tratta di una versione modificata di UNIX (BSD), basata sul kernel ibrido XNU, ottenuto dalla fusione dei progetti del kernel monolitico FreeBSD e del microkernel Mach, un approccio che permette di sfruttare i vantaggi di entrambe le strutture architetturali.

Mantenendo stessa architettura e stesso sistema operativo in tutti i dispositivi, Apple ha potuto operare una forte ottimizzazione, ottenendo come risultato un sistema stabile, veloce ed affidabile.

La volontà di controllo sull'esperienza dell'utente non si ferma solamente alla produzione di hardware proprio, ma anche in specifiche scelte architetturali del sistema operativo.

Si sente spesso affermare che il mondo dei dispositivi Apple è “chiuso”. Chi utilizza questo termine si riferisce alle politiche messe in atto dall’azienda per far si che l’utente abbia a disposizione, in ogni situazione, delle funzionalità sicure e controllate.

La maggior parte dei programmi sviluppati, compreso il sistema operativo, adotta una licenza proprietaria e si cerca in ogni modo di impedire la modifica degli stessi da parte di sviluppatori terzi.

Le applicazioni possono essere installate solamente tramite l’App Store, il programma integrato in ogni dispositivo che simula un “supermercato” dove i clienti possono ottenere copie delle applicazioni, gratis oppure a pagamento.

Per essere pubblicate nell’App Store le app devono superare un rigoroso controllo, che permette all’azienda di filtrare contenuti indesiderati (le applicazioni pornografiche ad esempio non sono permesse), comportamenti indesiderati e malware.

I dispositivi possono interfacciarsi con il computer solo attraverso l’apposita applicazione iTunes, che viene utilizzata per trasferire e sincronizzare i dati tra computer e smartphone.
L’utente non dispone di accesso “root” al dispositivo, ma alcuni sviluppatori amatoriali, nel corso degli anni, sono riusciti ad insinuarsi nei sistemi di sicurezza, rilasciando dei programmi in grado di “sbloccare” il dispositivo sfruttando delle falle. Questo processo, noto come “jailbreak” (rottura della prigione), permette di installare software di terze parti non riconosciuto da Apple, come copie illegali di applicazioni a

pagamento e software che permette di effettuare modifiche al sistema altrimenti impossibili.

Il sistema operativo è diviso in 4 livelli di astrazione:

- Core OS Layer

  E' il livello più basso del kernel, si occupa di gestire l'hardware e quindi di fornire ai livelli superiori delle System Call per accedere alle funzionalità tipiche del kernel: socket, file system, autenticazione utenti, sicurezza, gestione dell'energia, firmware e driver delle periferiche
- Core Services

  Questo livello implementa le funzionalità base dell'interfaccia utente: rubrica, telefonate, database SQlite, posizione GPS, accesso ad internet, fotocamera
- Media

  Questo livello si occupa di gestire la riproduzione, il salvataggio e la sincronizzazione di contenuti multimediali: foto, video, animazioni, sincronizzazione della libreria con iTunes
- Cocoa Touch

  Questo livello si occupa di realizzare l'interfaccia grafica attraverso la quale l'utente interagisce con il sistema.

  E' lo strato di astrazione più elevato e si serve delle chiamate di tutti i livelli inferiori per ottenere le funzionalità desiderate.

4.1.2 Piattaforma di sviluppo

Nel 2008, anno successivo al rilascio dell'iPhone, Apple ha rilasciato l'iOS SDK (Software Development Kit).

Gli sviluppatori che intendono rilasciare applicazioni nell'Apple Store devono iscriversi all'iOS Developer Program, pagando 99 dollari all'anno.
Una volta iscritti, si possono pubblicare applicazioni gratis o a pagamento. Lo sviluppatore incasserà il 70% degli incassi derivanti dall'acquisto dell'applicazione nello Store.

L'SDK è compatibile solo con piattaforme che eseguono Mac OS, quindi lo sviluppo di applicazioni native su iOS è vincolato al possesso di un computer della serie Mac.

Le applicazioni per iOS sono scritte in linguaggio Objective-C, lo stesso del framework grafico Cocoa.

Utilizzando il kit di sviluppo si ha a disposizione ogni strumento necessario allo sviluppo, test e debug dell'app.

Tramite l'IDE XCode, si può scrivere, controllare ed eseguire il codice attraverso un simulatore, che come una vera e propria macchina virtuale, simula l'esecuzione della app in un dispositivo iPhone o iPad.

Il simulatore mette a disposizione diversi tipi di emulazione legati ai vari sensori. In questo modo si possono testare le funzionalità legate all'hardware del dispositivo, come i satelliti GPS o il giroscopio, senza la necessità di possedere un iPhone.

Lo sviluppo di applicazioni iOS segue il pattern Model - View - Controller (MVC).

Ognuna di queste componenti dell'applicazione implementa una specifica funzionalità.

Tale separazione semplifica il mantenimento e l'aggiornamento del software, rilegando a pezzi di codice separati compiti logicamente differenti.
In particolare:

- Model: contiene e modella le strutture dati necessarie all'applicazione. Astrae le classi e gli oggetti gestendone la memorizzazione, la creazione e la distruzione degli stessi. (es. memorizzazione e creazione di account utenti con indirizzo email e password)
- Controller: realizza la logica dell'applicazione interagendo con il modello dati (es. metodi di login/logout)
- View: realizza la visualizzazione grafica dei componenti Cocoa, agganciando ogni elemento a determinati metodi del controller (es. posizione e colore del bottone "Login" che richiama il metodo di login nel controller)

4.2 Android

4.2.1 Storia e mercato

Il sistema operativo di Google è attualmente in prima posizione per quanto riguarda il numero di dispositivi venduti.

Worldwide Device Shipments by Operating System (Thousands of Units)				
Operating System	2012	2013	2014	2015
Android	503,690	877,885	1,102,572	1,254,367
Windows	346,272	327,956	359,855	422,726
iOS/Mac OS	213,690	266,769	344,206	397,234
RIM	34,581	24,019	15,416	10,597
Chrome	185	1,841	4,793	8,000
Others	1,117,905	801,932	647,572	528,755
Total	2,216,322	2,300,402	2,474,414	2,621,678

(fig1: L'andamento delle vendite dei diversi tipi di smartphone negli ulitmi anni)

Come già accennato precedentemente, le scelte progettuali che hanno portato alla diffusione di questa piattaforma, sono state in molti casi completamente opposte a quelle della sua maggiore antagonista, iOS.

La prima differenza consiste nella modalità di distribuzione del sistema: Google non produce dispositivi hardware propri, ma affida ad aziende terze il compito di realizzare cellulari che, tramite contratti e concessioni, verranno venduti con una copia di Android.

I produttori in molti casi modificano il sistema che viene fornito da Google, attuando delle ottimizzazioni sull'hardware, aggiungendo o modificando l'interfaccia grafica predefinita ed aggiungendo funzionalità non previste.

Questa scelta ha portato alla realizzazione e alla vendita di un incredibile numero di dispositivi, indirizzati verso diverse nazioni, fasce di prezzo e ambiti di utilizzo.

Google ha stretto accordi commerciali con 599 marchi, i quali hanno prodotto, in tutto l'arco di vita di questo progetto, 3997 differenti modelli[2].

L'aggiornamento ed il mantenimento di un sistema operativo indirizzato ad un numero cosi elevato di dispositivi differenti rappresenta una enorme sfida per l'azienda di Montain View.

Per ragioni di compatibilità, l'azienda ha inevitabilmente dovuto operare un compromesso, realizzando un sistema che non può adattarsi ad una sola architettura e configurazione hardware.
Questo adattamento è implementato introducendo astrazione, che pur migliorando la mantenibilità del codice, porta inevitabilmente a maggiore overhead.

Le ottimizzazioni e personalizzazioni introdotte dalle aziende produttrici, inoltre, si riducono spesso all'aggiunta di ulteriori strati di astrazione, piuttosto che alla modifica vera e propria del sistema, peggiorando le performance e introducendo quello che solitamente viene denominato "bloatware", cioè software aggiuntivo indesiderato che nella maggioranza dei casi non viene utilizzato.

Per questo motivo, anche se Android permette di ottenere eccellenti prestazioni sui dispositivi di fascia medio-alta, negli smartphone economici la fluidità e la velocità del sistema risulta notevolmente ridotta.

Un'altra grande scelta progettuale che differenzia di molto Android da iOS, è la politica di rilascio del software.

[2] http://www.focus.it/tecnologia/quanti-modelli-android-esistono-al-mondo_C12.aspx

Come per altri suoi prodotti, Google adotta una politica “aperta”.
Investendo nei valori del Software Libero e della condivisione della conoscenza.

Gli stessi sviluppatori dichiarano che lo scopo di Android è di favorire l’apertura dei sistemi software per cellulari, e vogliono farlo lasciando libero l’utente di fare cosa vuole con il proprio dispositivo[3].

Il codice di Android viene rilasciato, attraverso il progetto AOSP (Android Open Source Project), con una licenza Apache 2.0[4]

Alcuni parti di Android, in particolare alcuni firmware e driver, purtroppo sono composte da “*blob”* di codice binario e proprietario.
Questo avviene perchè molti produttori di hardware non rilasciano il codice sorgente di driver e firmware adottati su dispositivi Android.

Grazie alla disponibilità del codice, nel tempo sono nate centinaia di versioni modificate di Android, le cosiddette “custom ROM”.

I motivi che giustificano questi fork sono molteplici:

- rimozione del bloatware: vengono eliminate tutte le personalizzazioni e programmi aggiuntivi inseriti dal produttore del dispositivo, per ottenere un sistema più veloce e che occupa meno memoria.

[3] “Android is about freedom and choice. The purpose of Android is promote openness in the mobile world, and we don't believe it's possible to predict or dictate all the uses to which people will want to put our software. So, while we encourage everyone to make devices that are open and modifiable, we don't believe it is our place to force them to do so. Using LGPL libraries would often force them to do just that.” - https://source.android.com/source/licenses.html
[4] http://www.apache.org/licenses/LICENSE-2.0

- modifiche al kernel: vengono apportate modifiche di basso livello per ottimizzare il kernel ad una particolare architettura. Nella maggior parte dei casi vengono applicate patch già esistenti. Spesso si tratta dell'aggiunta di particolari scheduler che ottimizzano l'utilizzo del processore da parte dei processi. Inoltre permettono di far scegliere all'utente diverse modalità di gestione della frequenza del processore, a partire da politiche orientate alla performance che aumentano velocemente la frequenza, fino a quelle che la mantengono costantemente bassa per ottenere il massimo risparmio energetico.

- interfaccia grafica: vengono aggiunti software realizzati ad hoc, spesso da sviluppatori amatoriali, per aggiungere funzionalità e migliorare l'interfaccia grafica di Android. Un esempio tra tutti, che ha riscosso molto successo, è il sistema di notifiche HALO)))[5], creato appositamente per essere inserito nella ROM Paranoid Android[6] e ultimamente sostituito dala sua evoluzione Hover.

- firmware personalizzati: in rete sono disponibili anche "modem" personalizzati, cioè il programma binario che interfaccia il sistema operativo alle antenne del telefono. Vengono rilasciate versioni modificate per adattarsi meglio al tipo di celle telefoniche presenti in una determinata nazione e per ottimizzare la qualità del segnale ed il consumo di batteria.

- accesso root: una delle caratteristiche principali delle ROM modificate è che permettono di accedere al sistema operativo con permessi di amministratore. Le versioni ufficiali di Android sono dotate di utenti con permessi limitati in

[5] https://github.com/ParanoidAndroid/HALO
https://play.google.com/store/apps/details?id=com.paranoid.halo
[6] http://paranoidandroid.co/

modo predefinito. In questo modo si impedisce all'utilizzatore finale di poter scrivere nelle cartelle di sistema. Questo impedisce modifiche accidentali che potrebbero danneggiare il dispositivo e che applicazioni malevole possano influire negativamente sul funzionamento del sistema operativo.

Tra le versioni modificate di Android, la più diffusa e conosciuta è sicuramente CyanogenMod.
Ideata e lanciata dallo statunitense Steve Kondik, questa ROM è distribuita ufficialmente per più di 60 hardware e ha raggiunto la quota di 5 milioni di utenti[7]

Il principale scopo di CyanogenMod è fornire un sistema operativo il più possibile libero e pulito da ogni tipo di personalizzazione apportata dai produttori.

Per questo motivo viene utilizzato come base il codice sorgente rilasciato da Android Open Source Project, apportando principalmente queste modifiche:

- tema grafico: come appare evidente dal nome della ROM, il sistema operativo viene dotato di un'interfaccia color ciano
- launcher: CyanogenMod dispone di un proprio launcher, cioè l'applicazione che si occupa di visualizzare le icone delle applicazioni nella pagina principale del sistema operativo. Questo componente è stato chiamato "trebuchet".
- Google Apps: per problemi di licenza, CyanogenMod viene fornita senza i pacchetti base di Google, compreso il Google Play, l'applicazione attraverso la quale è possibile scaricare tutti i programmi di terze parti disponibili su Android. Comunque è possibile installare questi elementi di corredo semplicemente scaricando un pacchetto aggiuntivo.

[7] http://stats.cyanogenmod.org/

Riteniamo importante, inoltre, citare il progetto Replicant, un fork di CyanogenMod che, a partire dal codice sorgente dello stesso, rimuove ogni traccia di software proprietario.

Benchè questo approccio impedisca di utilizzare Replicant su un gran numero di dispositivi, vista la mancanza di firmware e driver, in questo modo è possibile disporre di un sistema operativo che sia completamente software libero, ottenendo il pieno controllo del proprio smartphone.

4.2.2 Piattaforma di sviluppo

Contestualmente al rilascio della versione beta di Android, il 12 Novembre 2007 è stato rilasciato l'Android Software Development Kit (SDK).

Questo pacchetto di sviluppo, è costituito da un corredo di programmi utili alla realizzazione di applicazioni di terze parti installabili nel dispositivo.

Oltre ad un IDE integrato (una versione modificata di Eclipse, che monta dei plugin appositi), un emulatore basato su QEMU per provare le applicazioni, un tool per scaricare le librerie utilizzate dall'applicazione, l'SDK contiene anche una ricca parte di documentazione ed esempi di codice.

E' compatibile con tutti i maggiori sistemi operativi: GNU/Linux, Mac OS e Microsoft Windows, quindi un ampio numero di utenti possono sviluppare applicazioni per la piattaforma mobile di Google, a differenza di iOS, le cui applicazioni possono essere realizzate solo possedendo un Mac.

Le applicazioni realizzate con l'SDK vengono compilate e diventano un "pacchetto", cioè un file con estenzione .apk pronto per essere installato nel dispositivo.

Questo permette, oltre che al tradizionale metodo di distribuzione attraverso i Google Play Store, di diffondere la propria applicazione anche attraverso metodi alternativi, come ad esempio pubblicare il file su un sito web.

Per installare software al di fuori dello Store, direttamente attraverso il pacchetto apk, l'utente deve semplicemente abilitare questa possibilità nelle impostazioni.

Il metodo più diffuso per pubblicare e scaricare applicazioni, comunque, è quello di pubblicare il proprio lavoro sul Google Play Store.

Per pubblicare applicazioni è necessario registrarsi alla Developer Console di Google, pagando una quota di 25$, che a differenza dell'App Store di Apple, non è annuale ma una tantum.

Google opera molti meno filtri rispetto al contenuto delle applicazioni che richiedono di essere pubblicate (ad esempio le applicazioni pornografiche sono tollerate, a differenza di App Store).

Vengono comunque operati dei controlli per evitare che sul Play Store venga pubblicato del malware che possano danneggiare il sistema operativo od avere effetti indesiderati per l'utente.

Le applicazioni per Android vengono realizzate in linguaggio Java, lo stesso utilizzato per tutta l'interfaccia grafica del sistema.

Volendo rappresentare la composizione di Android, otteniamo, similmente ad iOS, una struttura composta da diversi livelli di astrazione.

Android è basato su kernel Linux, in particolare la versione 2.6 del kernel fino alla versione 3 di Android, e basato su Linux 3.x da Android 4.0

Il kernel Linux è stato modificato dagli sviluppatori Google e in alcuni aspetti è molto differente da quello adottato dalle tradizionali distribuzioni GNU/Linux.

Sono state aggiunte molte funzionalità di basso livello, in particolare rivolte all'ottimizzazione energetica, come ad esempio l'introduzione dei wakeclocks, un meccanismo utilizzato dalle applicazioni per eseguire servizi in background che vengono risvegliate periodicamente per eseguire il loro task.

La struttura del file system non rispecchia quello tradizionale, ma viene adottata una gerarchia adattata al sistema Android[8].

Non è presente il server grafico X, quindi non si può utilizzare il tradizionale approccio per visualizzare componenti grafici e finestre.

Dopo una lunga e controversa discussione sui pro ed i contro del fork del kernel Linux, gran parte di queste modifiche sono state integrate nel ramo principale della versione 3.5 del kernel sviluppato da Linus Torvalds.

[8] tutto il sistema Android, ad esempio, è contenuto nella sottoradice /system

Oltre a linux che si occupa di dialogare con le periferiche del telefonino, tramite firmware e driver, il sistema è fornito anche di un ampio spettro di librerie che garantiscono le funzionalità di base, la maggior parte delle quali sviluppate in linguaggio C, ad esempio la libreria SSL che permette di effettuare comunicazioni cifrate con l'omonimo protocollo crittografico.

Sopra al sistema di base è presente il livello dell'Application Framework.
Questa parte di Android, realizzata principalmente in C++ e C, si occupa di fornire i modelli di astrazione base del funzionamento del telefono, che verranno poi utilizzati dalle applicazioni.
Uno dei framework che compongono questa parte, ad esempio, è quello della geolocalizzazione, che fornisce delle API attraverso le quali una app può richiedere la posizione attuale del dispositivo.

E' questo livello, quindi, che si occupa di instradare tutte le richieste provenienti dalle applicazioni utente, indirizzandole verso il componente che le realizza effettivamente (la libreria SQlite, ad esempio, nel caso dell'accesso alla rubrica del telefono).

Oltre a questo insieme di Framework, è presente quello che viene definito Android Runtime, il livello d'astrazione che mette a disposizione delle applicazioni le librerie e le chiamate di sistema realizzate in linguaggio Java.
Queste richieste vengono eseguite dalla Dalvik Virtual Machine, una componente realizzata completamente da Google e che presenta molte analogie e differenze con la tradizionale Java Virtual Machine (JVM).

Dalvik è stata ottimizzata tenendo in considerazione la scarsa memoria dei dispositivi mobili.

Il codice Java delle applicazioni viene convertito in Java Bytecode, che viene preso in pasto da un tool denominato dx che lo converte ulteriormente in Dalvik Bytecode, un formato che rispetta il particolare set di istruzioni supportato dalla Dalvik Virtual Machine.

Al di sopra di tutti questi componenti, sono presenti le applicazioni vere e proprie eseguite in Android, sia quelle già presenti nel sistema (widget grafici, telefonate, SMS, rubrica), sia quelle installate dal Play Store o da terze parti tramite pacchetto .apk.

Le applicazioni Android realizzate in Java, utilizzano l'astrazione delle Activity per implementare le varie viste, e l'astrazione dei Service, per accedere o implementare servizi in background, da eseguire in secondo livello per, ad esempio, controllare e notificare la presenza di nuovi messaggi in una applicazione di messaggistica.

4.3 Windows Phone, BlackBerry, Amazon Fire, Ubuntu Touch, Firefox OS.

Pur avendo preso in considerazione la realizzazione dell'applicazione per queste piattaforme minoili, la trattazione approfondita di queste piattaforme che hanno quote di mercato minoritarie esula dagli obiettivi di questa tesi.

Capitolo 5: Approcci alla programmazione su piattaforme mobili

Durante la fase di progettazzione dell'applicazione mobile che ho realizzato durante il tirocinio, abbiamo valutato attentamente tutte le modalità e le tecnologie esistenti per lo sviluppo software in questo settore.

Il risultato che è apparso subito evidente è la spaccatura tra due approcci molto diversi ma egualmente validi: programmazione nativa e ibrida.

Di seguito analizzeremo le caratteristiche, i pro e i contro di ognuno di questi approcci, descrivendo poi, le motivazioni che ci hanno indirizzato verso la scelta di utilizzare uno piuttosto che un altro.

5.1 Applicazioni native

Come già descritto nei capitoli precedenti, ogni piattaforma mobile mette a disposizione il proprio ambiente di sviluppo, implementando una serie di API in un determinato linguaggio di programmazione.

Oltre a queste, solitamente, viene rilasciato un kit di utility che contiene tutti gli strumenti necessari per realizzare applicazioni per quel particolare sistema operativo.

Le applicazioni realizzate seguendo questo approccio, sono denominate “native”, in quanto utilizzano le tecnologie messe a disposizione del produttore, mantenendo una forte omogeneità con il resto del sistema operativo e dell’interfaccia.

I vantaggi di questa astrazione messa a disposizione dagli sviluppatori del sistema operativo sono molteplici.

- Librerie

  Si ha a disposizione un grande numero di funzioni, classi ed astrazioni già realizzate e pronte ad essere utilizzate.

  Molte di queste sono già integrate nell’SDK, ma per le esigenze più particolari, si può attingere alla grande quantità di librerie pubblicate da sviluppatori terzi, professionisti e amatoriali.
- Interfaccia

  Utilizzando le API del sistema operativo, si possono posizionare e decorare le proprie applicazioni, con la stessa serie di widget grafici[9] di cui è composta l’interfaccia base del sistema.

  Il cosiddetto “look & feel” dell’applicazione che si stà realizzando, cioè l’insieme di pulsanti, campi di inserimento, menù, popup, sarà omogeneo con il sistema di base e con le altre applicazioni native.
- Ottimizzazione

  Lavorando allo stesso livello d’astrazione dell’interfaccia delle applicazioni base del telefono, le applicazioni native possono godere di tutta la velocità e le prestazioni messe a disposizione dalla piattaforma, comprese le ottimizzazioni operate da ogni singolo sistema operativo.

[9] Con questo termine si indicaon i vari “riquadri” che compongono l’interfaccia grafica: liste, interruttori, menù, popup ecc...

- Documentazione

  Le API native sono sempre documentate in modo eccellente, e gli SDK sono solitamente forniti di utilissimi programmi di esempio, che mostrano il corretto utilizzo delle API per realizzare le funzionalità più comune e istruire l'utente sui pattern ricorrenti della piattaforma.

  Oltre a questo, è possibile trovare in rete una enorme quantità di documentazione, che varia dalle semplici guide per le stesse API delle piattaforme, a pezzi di codice già scritti che implementano semplici funzionalità, a forum dove si possono trovare facilmente le soluzioni ai problemi riscontrati più frequentemente.

5.2 Applicazioni web

Ultimamente, grazie all'evoluzione delle funzionalità e delle performance dei browser mobili, si è assistito alla nascita di applicazioni web ottimizzate per smartphone e tablet.

La realizzazione di siti web consultabili da cellulare è iniziata fin dai primi dispositivi che erano in grado di collegarsi alla rete con delle lentissime connesioni WAP e browser molto primitivi.

Con l'aumento delle prestazioni e la nascita di nuove tecnologie (CSS, AJAX, HTML5), la navigazione da cellulare è cresciuta in maniera esponenziale tanto da rappresentare circa il 15% del traffico web mondiale[10].

[10] http://gs.statcounter.com/#all-browser-ww-monthly-201306-201406

Mentre inizialmente i siti web per cellulari erano versioni completamente separate e con funzionalità fortemente ridotte rispetto a quelli destinati al desktop, l'avvento di CSS3 ha permesso di realizzare i cosiddetti layout "*responsive*", cioè rendere disponibile lo stesso sito web a tutti i tipi di dispositivi.

Modificandosi in base alla dimensione dello schermo, il layout si adatta al dispositivo utilizzato, permettendo di fornire all'utente un'esperienza unificata.

L'arrivo di HTML5 e AJAX, con il miglioramento delle performance e le funzionalità fornite da JavaScript, ha permesso la nascita di complesse applicazioni web.

Mentre prima il web si limitava ad essere uno strumento per ottenere e fruire informazioni, con il cosiddetto Web 2.0 il browser diventa un potente strumento di elaborazione dati: le informazioni vengono create, modificate e condivise in tempo reale.

Tutti i maggiori siti web oggi dispongono di un'ottimizzazione per piattafome mobili. Facebook, il più grande social network del mondo, ha annunciato di aver raggiunto la quota di un miliardo di utenti che si collegano da un dispositivo mobile[11].

Tuttavia, nonostante le potenzialità delle applicazioni web che vengono visualizzate sul browser del dispositivo, queste sono limitate rispetto a quelle che teoricamente un'applicazione nativa è in grado di offrire.

---

11 http://techcrunch.com/2014/04/23/facebook-passes-1b-mobile-monthly-active-users-in-q1-as-mobile-ads-reach-59-of-all-ad-sales/?ncid=rss

I browser dei moderni sistemi operativi mobili non sono ancora in grado di accedere alle funzionalità dell'hardware dei device utilizzando le API di sistema descritte precedentemente.
In particolare, sono negate le numerose potenzialità di interazione con l'utente e con l'ambiente rese possibili dai sensori presenti negli smartphone.

Il W3C sta definendo degli standard per accedere a queste funzioni, ma non sono ancora ufficiali e solo pochi di questi sono stati già implementati nei browser più diffusi.

Possiamo ipotizzare, inoltre, che il trend nei prossimi anni non varierà sensibilmente, in quanto l'accesso a tali sensori e funzionalità avanzate costituisce uno delle principali mezzi di differenziazione fra un produttore e l'altro.

Nonostante queste limitazioni, l'approccio delle applicazioni web ha l'indiscusso vantaggio di uniformare lo sviluppo del software: non è necessario apprendere e conoscere tutte le tecnologie, i linguaggi ed i framework di ogni piattaforma su cui si vuole rendere disponibile il programma.

Utilizzando HTML, JavaScript e CSS, tecnologie molto diffuse e consolidate, si rende automaticamente fruibile l'applicazione a qualunque dispositivo dotato di un browser moderno.

5.3 Applicazioni ibride

Esiste un altro approccio alla programmazione che unisce i vantaggi delle modalità descritte in precedenza.

Recentemente sono state sviluppate delle soluzioni che permettono di sviluppare applicazioni mobili utilizzando le tecnologie web (HTML5, JavaScript e CSS), rendendole disponibili su tutte le piattaforme e disponendo, allo stesso tempo, della possibilità di utilizzare le periferiche del dispositivo.

In questo ambito, l'analisi effettuata durante il tirocinio ha riscontrato l'esistenza di due tecnologie che si contendono la scena: Apache Cordova e Titanium.
La fase di progettazione di Connex ha comportato l'analisi di entrambe le soluzioni per capire quale fosse in grado di soddisfare meglio le esigenze del committente.

5.3.1 Apache Cordova (PhoneGap)

Questo framework, nato da Nitobi ed acquistato nel 2011 da Adobe Systems, che gli ha dato il nome di PhoneGap, permette la realizzazione di applicazioni ibride, sfruttando i moderni standard web.

Il codice sorgente è stato donato alla Apache Foundation, che si occupa del mantenimento e dell'aggiornamento sotto il nome di Apache Cordova.

Pur con nomi differenti, Apache Cordova e PhoneGap rappresentano, di fatto, la stessa tecnologia, a cui ci riferiremo, da qui in poi, con il nome di Cordova.

Cordova permette di installare nel dispositivo un'applicazione nativa che consiste in una semplice WebView, cioè una vista che richiama il browser del sistema operativo per visualizzare pagine web.

Le pagine visualizzate da tale browser integrato vengono installate nella memoria insieme all'applicazione, in quanto contenute nel pacchetto[12].

La app, quindi, consiste in una vera e propria applicazione web che viene realizzata con le moderne e diffuse tecnologie HTML5, JavaScript e CSS3.

Utilizzando solamente la diffusissima tecnologia delle pagine web, Cordova si occupa di rendere compatibile l'applicazione su tutte le piattaforme con una diffusione non trascurabile[13].

Questo framework, inoltre, ha il notevole pregio di mettere a disposizione un canale di comunicazione tra il browser integrato ed il sistema operativo, permettendo di utilizzare le periferiche e le funzionalità tipiche delle applicazioni native, che, come già discusso, risulterebbero altrimenti inaccessibili dal browser tradizionale.

Tale meccanismo viene implementato da Cordova mettendo a disposizione delle API in linguaggio JavaScript che inviano messaggi all'applicazione nativa di base la quale, nell'opportuno linguaggio di programmazione caratteristico di ogni piattaforma, si occupa di richiamare la funzionalità desiderata.

[12] è comunque possibile caricare URL remoti.
[13] http://cordova.apache.org/platforms

Gli sviluppatori di Cordova si sono occupati di realizzare le chiamate a queste API per ogni piattaforma esistente, in modo che il processo risulti completamente trasparente allo sviluppatore.

Le API, inoltre, sono state realizzate sulla base degli standard definiti dal W3C[14].
Se e quando queste venissero introdotte in maniera definitiva ed implementate nei browser, Cordova non avrebbe più motivo di esistere e le applicazioni con tale framework sarebbero da subito compatibili e funzionanti direttamente nel browser del sistema operativo.

Lo scopo stesso di Cordova, dichiarato dagli stessi manutentori, è infatti quello di "scomparire"[15].

Le API sono ricche, complete e ben documentate[16] e permettono di realizzare quasi tutte le funzionalità che si potrebbero implementare con un'approccio di programmazione nativo.

[14] http://www.w3.org/standards/techs/mobileapp#w3c_all
[15] http://phonegap.com/2012/05/09/phonegap-beliefs-goals-and-philosophy/
[16] http://cordova.apache.org/docs/en/3.2.0/index.html

5.3.2 Titanium

Questo framework, rilasciato e mantenuto dall'azienda Appcelerator, può risultare solo apparentemente simile a Cordova, ma in realtà affronta il problema con un approccio completamente diverso[17].

Se da un'analisi superficiale appare che entrambi permettano permettono di scrivere applicazioni in JavaScript, mentre Cordova esegue il codice assieme all'HTML e CSS in un browser, Titanium interpreta il codice JavaScript similmente a come avviene nella programmazione server-side con PHP, Ruby, Perl e node.js.

Gli sviluppatori di Titanium hanno sviluppato un livello di astrazione con delle API in javascript che permette di richiamare le funzionalità dei singoli sistemi operativi.

Ad esempio, piuttosto che dover studiare come si implementa una lista di elementi scorrevoli su diverse piattaforme e diversi linguaggi, uno sviluppatore che utilizzi Titanium può, ad esempio, richiamare la funzione JavaScript messa a disposizione dal framework per indicare la sua intenzione di "posizionare" una lista scorrevole.

Una volta scritto il codice, Titanium si occuperà di interpretarlo e trasformarlo in chiamate effettive ai singoli sistemi operativi, ognuno con il proprio linguaggio di programmazione.

Per realizzare questo meccanismo Titanium mette a disposizione un ricco SDK, basato su una versione modificata dell'IDE Ecplise.

[17] http://www.appcelerator.com/blog/2012/05/comparing-titanium-and-phonegap/

Lo sviluppatore deve installare nella propria macchina i kit di sviluppo di ogni piattaforma su cui vuole rendere compatibile il proprio software.
L'SDK di Titanium si occupa poi di tradurre il codice JavaScript in codice nativo e costruire i pacchetti dell'applicazione per ogni piattaforma, pronti ad essere installati.

5.3.3 Differenze e punti di unione

Mentre Cordova cerca di uniformare lo sviluppo passando attraverso l'astrazione del browser quindi, Titanium non segue un approccio "web", ma cerca piuttosto di uniformare le API di programmazione delle applicazioni native di tutte le piattaforme mobili.

Questi approcci così differenti si traducono naturalmente in vantaggi e svantaggi variegati per ogni soluzione.

Il più grande vantaggio di Cordova è sicuramente la sua semplicità.
Utilizzando l'approcco di programmazione web diffuso e consolidato, anche uno sviluppatore alle prime armi è in grado di realizzare applicazioni in grado di essere installate su tutti i sistemi operativi mobili esistenti, accedendo anche alle funzionalità del dispositivo.

Un altro forte vantaggio è quello di poter contare su un grande numero di plugin, librerie e documentazione, che permettono di estendere facilmente il proprio programma con nuove funzionalità.

Titanium in alcuni frangenti può risultare più efficente di Cordova, perchè il suo codice si traduce effettivamente in codice nativo, ma questa considerazione non può

essere considerata vera in assoluto in quanto dipende molto dal tipo di applicazione che si intende realizzare e da come questa sfrutta le capacità computazionali del device.

Titanium ha il grande vantaggio di realizzare interfacce grafiche completamente fedeli al sistema operativo di destinazione, in quanto vengono sfruttate le stesse librerie grafiche della piattaforma nativa.

Nelle applicazioni Cordova, invece la realizzazione dell'interfaccia grafica resta totalmente appannaggio dello sviluppatore.
Pur ponendosi avvalere di una miriade di librerie grafiche, anche mirate a riprodurre il look&feel delle interfacce per cellulari, se si intende realizzare un'interfaccia che risulti simile a quella originaria del sistema operativo, il risultato non potrà mai essere completamente fedele all'originale.

Riassumendo, non possiamo affermare che un framework sia migliore di un'altro. Entrambi svolgono bene il compito per cui sono stati realizzati.
Se si ha bisogno di realizzare un'applicazione che deve necessariamente avere un'interfaccia grafica uniforme a quella del sistema operativo, Titanium è sicuramente la strada da seguire.

Se invece non si hanno vincoli sull'interfaccia, Cordova è una soluzione preferibile, considerando anche il fatto che man mano che i browser integreranno gli standard di accesso ai sensori nel dispositivo, il codice scritto per Cordova potrà essere utilizzato direttamente nel browser senza alcuna modifica.

Capitolo 6: Analisi dei requisiti

Lo scopo del tirocinio presso SMS ITALIA era quello della realizzazione di Connex, un'applicazione per dispositivi mobili per la distribuzione di contenuti multimediali.

L'analisi dei requisiti dell'applicazione è stata consegnata dal committente, e lo studio della stessa ha rappresentato la prima fase del tirocinio.
Di seguito presenteremo lo scopo ed il funzionamento dell'applicazione.

Connex serve a veicolare contenuti pubblicitari multimediali ed interattivi, realizzati a loro volta con le tecniche di sviluppo di webApp (HTML5 + Javascript + CSS).

Una volta visualizzato il contenuto multimediale, all'utente potrà essere eventualmente consegnato un coupon di sconto utilizzabile presso i punti vendita convenzionati.

L'applicazione, al primo avvio, chiede di autenticare l'utente inserendo un codice numerico inviato al destinatario a cura dell'Azienda, o tramite il proprio numero di cellulare.

L'interfaccia dell'applicazione è minimale e racchiude in modo intuitivo tutte le funzionalità.

Nella parte inferiore dello schermo è stata posizionata una "*tabstrip*", cioè un menù orizzontale che permette di accedere a diverse schede che identificano diverse funzioni.

La prima scheda è quella della Inbox e serve per visualizzare le campagne pubblicitarie ricevute.
In questa scheda è possibile sia visualizzare le campagne con un *tap* sia eliminare le campagne indesiderate attraverso un tasto di modifica.
La visualizzazione della campagna avviene in una nuova vista a pieno schermo.
Questa può richiedere l'interazione dell'utente attraverso animazioni, form da compilare, semplici giochi da completare, ecc.
L'utente ha la possibilità di interrmpere in qualsiasi momento la visualizzazione della campagna, ed eventualmente contrassegnare come indesiderata la campagna ricevuta, attraverso un pulsante in sovra-impressione che rappresenta tre punti, stile mutuato da Google.
Quando l'utente termina l'interazione viene rimandato alla schermata Inbox e gli verrà notificata eventualmente la disponibilità di un nuovo coupon promozionale.

Tali coupon sono visionabili nella seconda scheda dell'applicazione, denominata appunto "Coupon".
Come per le campagne, questi vengono visalizzati in una lista, ed è possibile aprirli o eliminarli.
Il coupon aperto consiste in un immagine, che al cetro può contenere eventualmente un barcode ottimizzato per la lettura ottica degli schermi.
Presentandosi nel punto vendita convenzionato con tale coupon, l'utente può disporre delle agevolazioni, e il personale del punto vendita ha la possibilità di annullare in modo semplice e veloce il coupon stesso.

La terza scheda dell'interfaccia è quella delle "Impostazioni", in cui è possibile modificare il numero di telefono con il quale si è registrati, visualizzare una statistica

sul numero di coupon e campagne ricevute, accedere ad un pannello informativo che spiega il funzionamento dell'applicazione.

Nelle impostazioni, inoltre, si ha la possibilità di riabilitare eventuali categorie pubblicitarie che sono state marcate come indesiderate durante la fruizione.

E' presente anche una quarta scheda di "Debug". Questa è visibile solo in modalità di sviluppo, e contiene un log di tutte le azioni effettuate dall'utente e dall'applicazione, molto utile per analizzare il funzionamento del programma ed individuare bug in caso di errori o comportamenti indesiderati.

Capitolo 7: Scelte progettuali

Considerate ed analizzate le specifiche dell'applicazione, descriviamo di seguito le accortezze, le strategie, le scelte operate e le tecnologie utilizzate per realizzarle.

7.1 Piattaforma

Dato che uno dei requisiti principali era quello che l'applicazione funzionasse almeno per le maggiori due piattaforme mobili esistenti (iOS e Android), è apparsa naturale la scelta di utilizzare una tecnologia ibrida in modo da semplificare e minimizzre l'onerosità non solo della parte iniziale di sviluppo ma anche quelle successive di debugging, aggiornamento e integrazione di nuove funzionalità.

Fra le due principali tecnologie sopra citate (Cordova e Titanium), la scelta che si è rivelata più adatta in relazione alle esigenze del committente, è stata quella di utilizzare Apache Cordova.

Visto che il modo piu semplice per realizzare le campagne multimediali era quello di utilizzare le ben note tecnologie di sviluppo web in HTML, JavaScript e CSS, abbiamo deciso di uniformare tutta l'applicazione a questa tecnologia, visto che Cordova permette da subito, di visualizzare contenuti e animazioni realizzate in HTML5.

7.2 Architettura interna

E’ stato scelto di utilizzare JQuery come framework JavaScript di base, visto che si tratta di una libreria molto leggera che velocizza lo sviluppo e la realizzazione di alcune operazioni ricorrenti come richieste AJAX e modifica del DOM.

L’interfaccia è stata realizzata utilizzando CSS3, realizzando un layout semplice e minimale che risultasse di facile utilizzo e ricordasse il più possibile il look& feel nativo dei rispettivi OS.

Per ogni funzionalità è stata realizzata un’astrazione, in modo da massimizzare la riutilizzabilità del codice nonché la sua leggibilità e quindi manutenibilità, e minimizzando pertanto l’incidenza di bug.

Per memorizzare dati in maniera persistente è stato scelto di utilizzare il localStorage, una funzionalità implementata in HTML5 che mette disposizione un dizionario chiave:valore per memorizzare informazioni in modo permanente, cioè che non si cancellano al caricamento di una nuova pagina.

Ogni funzionalità è stata incapsulata in un “singleton” in modo da simulare il comportamento di una classe, pur non essendo una funzionalità prevista dal JavaScript.

7.3 Richieste di rete

Come già analizzato nella prima parte di questo lavoro di tesi, i dispositivi mobili non dispongono di una connessione ad Internet sempre stabile, veloce ed affidabile.

Questo problema è stato affrontato con attenzione durante la fase di progettazione di Connex, definendo in modo accurato la modalità con cui l'applicazione si collega alla rete.

Quando ha necessità di inviare o richiedere informazioni al server, l'applicazione accoda, utilizzando un'apposita libreria di astrazione appositamente realizzata, in una lista di richieste da effettuale.

La libreria controlla se è presente la connessione, tramite la chiamata *connection.type*.
Se il dispositivo è collegato ad Internet la richiesta viene inviata subito, altrimenti la chiamata, invece di fallire, viene memorizzata.

Quando il dispositivo torna online, la libreria viene notificata grazie all'evento Cordova "*online*". A questo punto le richieste in coda vengono inviate, una alla volta, al server, invocando le rispettive funzioni di callback che avevano innescato la chiamata.

Non tutte le richieste hanno la stessa priorità: alcune azioni richiedono una maggiore interattività, ad esempio nel momento in cui si vuole aggiornare la inbox, la richiesta al server è strettamente correlata ad una modifica dell'interfaccia dell'applicazione.

Altre azioni invece corrispondono a semplici notifiche da inviare al server, che non prevedono un side-effect da parte dell'applicazione.

Le richieste più urgenti vengono inviate immediatamente (in modalità sincrona), mentre le notifiche possono essere posticipate ad un momento in cui non c'è l'esigenza di effettuare richieste urgenti, oppure ad un momento in cui è disponibile una connessione WiFi (modalità asincrona), tramite un meccanismo di priorità implementato nella libreria.

Cordova, infatti, permette anche di conoscere il tipo di rete a cui il dispositivo è collegato. Visto che i diversi tipi di connessione senza fili hanno caratteristiche ben diverse, come già spiegato, l'applicazione può in questo modo, comportarsi di conseguenza.
In particolare, sono disponibili le seguenti modalità:

- Connection.UNKNOWN
- Connection.ETHERNET
- Connection.WIFI
- Connection.CELL_2G
- Connection.CELL_3G
- Connection.CELL_4G
- Connection.CELL
- Connection.NONE

7.4 Blacklist

Il committente ha espresso la necessità di mettere in grado l'utente di bloccare un determinata categoria di contenuti a lui non graditi.

Questa funzionalità è stata implementata aggiungendo una voce al menù nella visualizzazione della campagna, che permette di chiuderla ed aggiungere il contenuto alla blacklist.

Se l'utente utilizza questa funzionalità, il server viene notificato dell'avvenuto inserimento nella blacklist, e la app nasconderà le campagne relative alle categorie indesiderate.

Nel pannello delle impostazioni, comunque, è possibile visualizzare le categorie bloccate e riabilitarle.
Se ciò viene fatto, le campagne che risultavano bloccate tornano ad essere visibili nella inbox.

7.5 Libreria di astrazione per le campagne multimediali

La scelta di utilizzare Cordova è risultata stratagica non solo per questi motivi generici, ma soprattuto per via della natura dell'applicazione.

L'esigenza di visualizzare contenuti multimediali scaricati da remoto, infatti, si coniuga perfettamente con il funzionamento di Cordova.

Le campagne multimediali sono delle semplici applicazioni web, realizzate in HTML5, JavaScript e CSS, quindi è molto facile e veloce realizzarle, perchè sono tecnologie standard e molto diffuse e non c'è bisogno di apprendere da zero un nuovo approccio di programmazione o imparare ad utilizzare un nuovo framework.

Le recenti funzionalità di HTML5, inoltre, permettono di realizzare delle interfacce e delle animazioni di grande prestazioni e qualità. I realizzatori delle campagne, possono avvalersi di librerie javascript e CSS di terze parti, purchè supportate da browser mobile.

In questo modo le possibilità di utilizzo delle campagne multimediali risulta veramente ampio, e limitato solamente dalla fantasia del creativo che le progetta.

Le librerie di Cordova, inoltre, permettono alle campagne di sfruttare le periferiche e le funzionalità del dispositivo, ad esempio, scattando una foto o prelevando un numero dalla rubrica.

Questo non sarebbe stato possibile utilizzando un'applicazione completamente web, o un'applicazione nativa che avesse integrato una semplice WebView.

Per facilitare ulteriormente il compito ai realizzatori delle campagne multimediali, è stata realizzata una libreria di astrazione per accedere alle funzionalità native del device in modo ancora più semplice ed intuitivo.

Agli sviluppatori viene fornita, inoltre, un'ampia documentazione sulle funzioni disponibili, che possibile consultare nell'allegato A di questo lavoro.

7.6 API per la comunicazione tra server e client

Connex dialoga con un servizio server-side per ricevere le campagne, i coupon di sconto e per sincronizzare i propri dati.

Il protocollo di comunicazione è stato definito in accordo con l'azienda.

E' stato deciso di utilizzare richieste HTTPS, abbinando il vantaggio della consolidata tecnologia HTTP incapsulata in un protocollo SSL che garantisce la sicurezza dell'identità degli utenti e dei dati a loro destinati.

Il client effettua richieste AJAX asincrone tramite la libreria JQuery, e il server processa le richieste comunicando con un DBMS di tipo MySql.

La scelta per il formato di scambio dati è caduta in modo naturale su JSON, per la sua standardizzazione *de facto* e la facilità di integrazione con il JavaScript.

Le API sono state orientate alla scalabilità, adottando delle precauzioni che permetteranno al server di gestire facilmente grandi quantità di richieste, se la applicazione dovesse diffondersi in maniera virale.

Per prima cosa, le chiavi dei dizionari JSON sono state ridotte ad una o due lettere, per minimizzare la dimensione delle richieste ed ottimizzare quindi la banda a disposizione.

Queste chiavi vengono espanse nel loro significato "umano", grazie ad un secondo dizionario, presente sul dispositivo dal momento dell'installazione dell'applicazione.

L’overhead è trasurabile in quanto queste operazioni vengono effettuate interamente client-side.

Per instradare al meglio le richieste e rendere possibile una politica di load-balancing molto granulare, è stata implementata anche una specifica richiesta API che ridefinisce gli URL di comunicazione per ogni chiamata.

L’applicazione, appena installata, comunica solamente con il server principale, l’unico attualmente presente.

Nel caso in cui si renda necessario l’instradamento delle richieste di un certo numero di dispositivi verso un nuovo server secondario, per bilanciare il carico, ad una qualsiasi comunicazione il server potrà includere nel body della risposta un flag, che viene controllato ogni volta dall’applicazione.

Una volta inserito questo flag, l’applicazione effettuerà la suddetta chiamata per rinnovare gli URL delle API al server, il quale risponderà con un dizionario, che associa ad ogni richiesta un indirizzo di un server.

Utilizzando questo stratagemma è possibile ottenere il massimo controllo sulle richieste che vengono effettuate dai client.
L’amministratore di sistema potrebbe decidere di spostare solo un tipo di richiesta, magari la più pesante dal punto di vista computazionale, su un nuovo server dedicato, senza causare alcun disservizio.

Oltre a questo, il server potrebbe adottare le tradizionali tecnologie di load-balancing, ottenendo una grande flessibilità e evitando rischi di Denial Of Service per un eccessivo carico di sessioni da parte dei dispositivi.

7.7 SDK ed applicazione di test

Per facilitare il lavoro dei realizzatori delle campagne multimediali, è stata realizzata una seconda applicazione a questo scopo.

Visto il lavoro di astrazione delle componenti e delle librerie JavaScript, è stato possibile riutilizzare gran parte del lavoro svolto su Connex per sviluppare una semplice applicazione dedicata agli sviluppatori.

Utilizzando questa app lo sviluppatore di una campagna può verificare il funzionamento delle campagne multimediali sui dispositivi mobili reali, per controllarne in particolare le funzionalità native (es. fotocamera, accelerometro…)

Nell'applicazione di test è possibile attivare più campagne in corso di test.
I messaggi di debug vengono visualizzati in un'apposita scheda, attraverso la quale è possibile capire la causa di eventuali errori e risolverli.

Capitolo 8: Valutazione

Per verificare la corretta realizzazione di Connex, ci eravamo posti degli obiettivi concreti, da controllare al termine del progetto.

Il primo, naturalmente, è quello del rispetto delle specifiche consegnate dal committente.

Un fattore che abbiamo ritenuto cruciale per la riuscita dell'applicazione, però, è stato quello della facilità di implementazione di campagne pubblicitarie da parte di aziende terze.

Le campagne multimediali, infatti, non verranno implementate da SMS ITALIA, ma fornite dalle stesse aziende che usufruiranno del servizio.

Per questo motivo abbiamo voluto essere certi di aver messo a disposizione delle agenzie creative tutti gli strumenti per consentire di realizzare le campagne in modo semplice ed efficace.

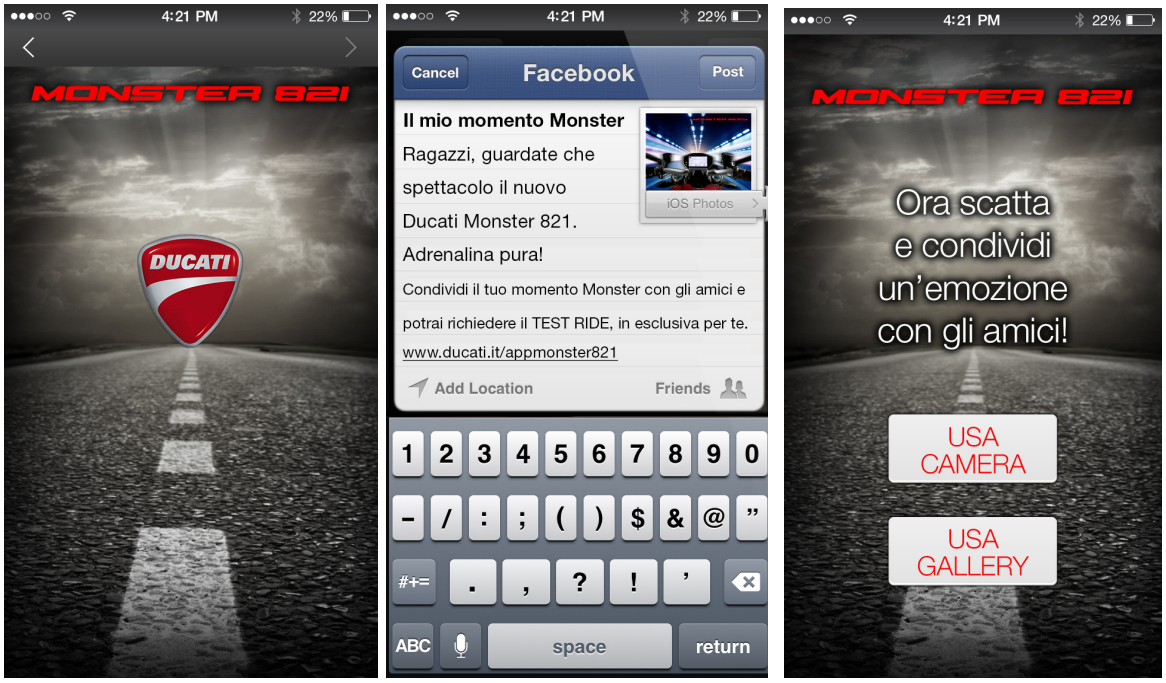
Abbiamo già spiegato i modi con cui abbiamo cercato di ottenere questo risultato: utilizzo delle diffuse tecnologie web (HTML5, JavaScript, CSS), la possibilità di utilizzare librerie Javascript esterne di qualsiasi tipo, la realizzazione di una libreria di astrazione documentata per accedere alle funzionalità del dispositivo e un'applicazione di test per provare il funzionamento delle campagne.

La verifica del primo criterio è stata operata dalla stessa azienda, in quanto il tutor aziendale ha assiduamente supervisionato ogni fase di sviluppo del progetto, validando le singole scelte implementative.

Questo ha fatto sì che il prodotto risultasse pienamente conforme alle specifiche fornite.

Riguardo al secondo criterio, ho avuto la fortuna di poter verificare in modo concreto l'utilizzo reale dell'applicazione per la realizzazione di una campagna-pilota per un prestigiosissimo marchio del territorio emiliano, famoso in tutto il mondo.
La bozza funzionante dell'applicazione, insieme alla libreria e alla documentazione realizzata, sono state già utilizzate con successo da un'azienda terza, che ha pertanto fornito feedback utili al miglioramento di Connex.

I commenti derivati da questo caso d'uso reale sono stati utili per modificare alcuni dettagli e per validare la correttezza delle scelte progettuali implementate.



(*Questi screenshoot provengono dalla campagna-pilota realizzata durante il periodo di realizzazione dell'applicazione, e provata tramite l'apposita applicazione di test abbinata a Connex)*

Capitolo 9: Conclusioni

In questo lavoro di tesi abbiamo descritto la realizzazione di Connex, un'applicazione mobile multipiattaforma destinata alla distribuzione di campagne pubblicitarie multimediali interattive.
Dopo una prima fase in cui abbiamo analizzato la situazione attuale dei maggiori sistemi operativi mobili, le caratteristiche e gli approcci alla programmazione mobile, abbiamo descritto l'analisi dei requisiti e le specifiche all'interno di cui l'applicazione sarebbe dovuta rientrare.
Abbiamo descritto e motivato le scelte progettuali, spiegando in quale modo abbiamo poi implementato le singole funzionalità.

Gli standard HTML5 definiti dal W3C per comunicare con i sensori degli smartphone si stanno diffondendo sempre di più.

Auspico che in un futuro non troppo remoto, come è già successo in buona parte nel mondo del software per desktop, la programmazione di applicazioni mobili si sposti sempre più dall'approccio nativo a quello ibrido, utilizzando linguaggi e API completamente disgregate e disomogenee, per finire all'interno dei browser.

Allegato A: documentazione della libreria di astrazione per gli sviluppatori delle campagne multimediali

La libreria vantage.js mette a disposizione diverse funzioni javascript per interagire con le funzionalità del device.

Tutte le funzioni sono racchiuse nella variabile globale "mse", quindi è sufficiente avere l'accortezza di non sovrascrivere tale variabile nel codice della campagna.

Descrizione delle funzioni:

Scatto fotografia dalla camera del device:

Permette di prelevare una foto dalla fotocamera del dispositivo.

```
mse.camera.getPhotoFromCamera(success, failure);
```

`success`: funzione che viene chiamata quando lo scatto della foto ha successo. riceve come parametro una stringa contenente la foto codificata in base64.

`failure`: funzione che viene chiamata in caso di errore della fotocamera. riceve come parametro una stringa contenente l'errore.

esempio:

```
mse.camera.getPhotoFromCamera(
  function success(imgData) {
    // imgData contiene l'immagine codificata in base64
  },
  function failure(msg) {
    // msg contiene il messaggio di errore riscontrato
  }
);
```

Scelta di una foto dalla galleria:

Permette di prelevare una foto dalla galleria del dispositivo.

```
mse.camera.getPhotoFromGallery(success, failure);
```

`success`: funzione che viene chiamata quando la prelevazione della foto ha successo. riceve come parametro una stringa contenente la foto codificata in base64.

`failure`: funzione che viene chiamata in caso di errore della galleria. riceve come parametro una stringa contenente l'errore.

esempio:

```
mse.camera.getPhotoFromGallery(
  function success(imgData) {
    // imgData contiene l'immagine codificata in base64
  },
```

```
  function failure(msg) {
    // msg contiene il messaggio di errore riscontrato
  }
);
```

Salvataggio di una foto nella galleria:

Permette di memorizzare nella galleria del dispositivo una immagine creata dalla campagna.

```
mse.camera.savePhotoToGallery(imgData, success, failure);
```

`imgData`: stringa contenente l'immagine codificata in base64

`success`: funzione che viene chiamata quando il salvataggio della foto ha successo. riceve come parametro una stringa contenente l'url della foto appena salvata nel file system del device.

`failure`: funzione che viene chiamata in caso di errore. riceve come parametro una stringa contenente l'errore.

esempio:

```
mse.camera.savePhotoToGallery(
  imgData,
  function success(url) {
```

```
    // url contiene il percorso dell'immagine memorizzata
  },
  function error(msg) {
    // msg contiene il messaggio di errore riscontrato
  }
);
```

Share Social

Permette di condividere una frase e una foto attraverso la funzione di condivisione del sistema operativo, e quindi anche attraverso i social network configurati.

```
mse.shareSocial(text, imgData, success, failure);
```

`text`: una stringa contenente il testo da condividere

`imgData`: stringa contenente l'immagine codificata in base64 da condividere, oppure null se non si vuole condividere nessuna foto

`success`: funzione che viene chiamata quando la condivisione ha successo.

`failure`: funzione che viene chiamata in caso di errore. riceve come parametro una stringa contenente l'errore.

esempio:

```
mse.shareSocial(
```

```
  'Ho provato questa app e mi è piaciuta molto!',
  imgData,
  function success() {

  },
  function error(msg) {
    // msg contiene il messaggio di errore riscontrato
  }
);
```

Logging

Invia messaggi al log della applicazione. Questi messaggi compaiono nella console e nella tab "Debug" della applicazione MSE Test

```
mse.log(string);
```

`string`: stringa contenente il messaggio da loggare

esempio:

```
mse.log('richiesta della fotocamera OK');
```

Avvio scaricamento del coupon

Per far si che l'utente riceva il coupon al termine della campagna, la campagna può, in qualsiasi momento, chiamare questa funzione.

```
mse.downloadCoupon(params);
```

params: [facoltativo] un oggetto javascript da passare come parametro al server per individuare il coupon corretto da scaricare.

Fine visualizzazione campagna

Per fare in modo che l'utente esca dalla campagna, una volta terminata la visualizzazione, è necessario richiamare questa funzione.
Nel caso in cui si voglia far scaricare il copun, è necessario chiamare questa funzione successivamente a quella dedicata allo scaricamento del coupon.

```
mse.endCampaign(showCoupon);
```

`showCoupon`: un parametro booleano che indica se l'utente deve visualizzare subito il nuovo coupon scaricato o no.

Bibliografia:

[1] Carli Massimo, Sviluppare applicazioni per Android, Edizioni Apogeo, 2011, pp. 416.

[2] Stark Jonathan, Sviluppare applicazioni per Android con HTML, CSS e JavaScript, Ed. Tecniche Nuove, 2011, pp. 159.

[3] http://cordova.apache.org/docs/en/3.5.0/

[4] https://source.android.com/source/licenses.html

[5] http://docs.appcelerator.com/titanium/latest/

[6] http://www.cs.unibo.it/~bononi/LAM2014/

[7] http://docs.phonegap.com/en/3.5.0/index.html

[8] Programming Android: Java Programming for the New Generation of Mobile Devices

[9] http://wiki.cyanogenmod.org/w/Main_Page

[10] http://redmine.replicant.us/projects/replicant/wiki

[11] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

[12] http://www.w3.org/2007/02/mwbp_flip_cards

Raggiungere il traguardo della laurea non consiste solamente nello scrivere queste poche pagine di tesi.
E' il coronamento di un sogno, per il quale si è combattuto, a volte duramente.
E' fermarsi a riposare dopo aver scalato la vetta, guardare il percorso fatto, e ricordarsi di quei momenti in cui avevi voglia di fermarti.
E' pensare a tutte le persone che ti hanno accompagnato e grazie alle quali sei riuscito ad arrivare in cima.

Ringrazio il profesor Vittorio Ghini, per l'infinita pazienza più volte messa alla prova e per avere a cuore l'insegnamento e il futuro dei suoi studenti.
Ringrazio il professor Renzo Davoli per avermi trasmesso la passione per la programmazione dei sistemi operativi e per aver mostrato ai propri studenti come dovrebbe ragionare un hacker.
Ringrazio il professor Fabio Vitali per avermi trasmesso una grande passione per la programmazione web.
Ringrazio Luca Ferroni per avermi insegnato i motivi per i quali il software vuole e deve essere libero e per avermi trasmesso delle solide basi tecniche, spesso ben superiori di quelle che questo corso di Laurea è stato in grado di offrirmi.
Ringrazio Emanuele Preda, per essere stato non solo un ottimo tutor, ma per aver preso a cuore me non semplicemente come collaboratore , ma come persona, fornendo consigli che si sono rivelati sempre azzeccati, anche in ambiti che esulavano dal proprio compito.

Grazie ai miei genitori che non si sono limitati a permettermi di seguire questo corso di studi, ma mi hanno insegnato a puntare in alto.