ALMA MATER STUDIORUM
UNIVERSITÀ DEGLI STUDI DI BOLOGNA

Seconda Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica

SVILUPPO DI MOBILE APP CON TECNOLOGIE WEB: TIZEN COME CASO DI STUDIO

Elaborata nel corso di: Sistemi Operativi LA

Tesi di Laurea di:
SARTI DAVIDE

Relatore:
Prof. ALESSANDRO RICCI

Co-relatori:
ANDREA SANTI

ANNO ACCADEMICO 2012–2013
SESSIONE II

PAROLE CHIAVE

Dispositivi Mobile

Smartphone

Applicazioni

Tecnologie Web

Tizen

Questa tesi è dedicata a tutta la mia famiglia, che mi è sempre stata accanto durante tutto il periodo universitario, sostenendomi ed incoraggiandomi.
Voglio dedicare questa tesi anche ai miei amici ed ai miei compagni di corso, che mi hanno aiutato nello studio e nella preparazione delle varie discipline durante tutto il percorso di studi, sempre con atteggiamenti positivi e cordiali.
Infine voglio ringraziare il mio relatore, il Dott. Alessandro Ricci, che mi ha aiutato e consigliato durante tutta la preparazione e lo sviluppo della tesi.

Indice

Introduzione della tesi **xi**

1 Introduzione ai dispositivi mobile **1**
1.1 Storia 1
1.2 Dispositivi Mobile 3
1.2.1 Cellulari 3
1.2.2 Tablet 7
1.2.3 PDA 10
1.3 Caso di interesse: Smartphone 12
1.3.1 Confronto con i telefoni cellulari 12
1.3.2 Caratteristiche e funzionalità 13
1.3.3 Sistemi operativi per Smartphone 14

2 Applicazioni per Smartphone **39**
2.1 Introduzione 39
2.2 Applicazioni native 40
2.2.1 Linguaggi di programmazione 40
2.2.2 Tools di sviluppo 42
2.2.3 Componenti chiave 45
2.2.4 Distribuzione 56
2.3 Applicazioni indipendenti dal sistema operativo 62
2.3.1 Applicazioni Web 62
2.3.2 Applicazioni Platform Indipendent 65
2.4 Applicazioni basate su tecnologie Web 66
2.4.1 Prospettiva 66
2.4.2 Tecnologie Web 67
2.4.3 Analisi dell'approccio 72

2.5 Confronto sulle diverse tipologie di Applicazioni 72
2.5.1 Vantaggi 72
2.5.2 Svantaggi 73
2.5.3 Valutazione applicazioni basate su tecnologie Web . . 75

3 Caso di studio Tizen **77**
3.1 Introduzione 77
3.2 Caratteristiche 77
3.2.1 Interfaccia grafica 78
3.2.2 Architettura della piattaforma 81
3.2.3 Sicurezza della piattaforma 83
3.2.4 Application Packpage Manager 86
3.2.5 Filtri 86
3.3 Tools di sviluppo 87
3.3.1 Tools generici 87
3.3.2 Tools per applicazioni native 88
3.3.3 Tools per applicazioni basate su tecnologie Web . . . 90
3.4 Applicazioni native 92
3.4.1 Tipologie di applicazioni native 92
3.4.2 Ciclo di vita di un'applicazione 93
3.4.3 Il Multitasking 93
3.4.4 Multithreading 96
3.4.5 Two-phase Construction 97
3.4.6 Politica di Object Ownership 98
3.4.7 Gestione delle eccezioni 98
3.4.8 Limitazioni della libreria C 99
3.5 Applicazioni basate su tecnologie Web 99
3.5.1 Web Runtime 100
3.5.2 Caratteristiche di sviluppo 103
3.5.3 Elementi di programmazione 108

4 Sviluppo applicazione con Tizen **129**
4.1 Introduzione 129
4.2 Progettazione 129
4.2.1 Struttura 130
4.2.2 Obbiettivi 130
4.3 Gestione Interfaccia 133

4.4 Funzionalità sviluppate 134
4.4.1 Batteria 136
4.4.2 GPS 137
4.4.3 Connessioni 139
4.5 Valutazioni finali 140

5 Conclusioni **143**
5.1 Futuri sviluppi 144

Introduzione della tesi

Lo scopo di questa tesi è quello di valutare l'utilizzo di tecnologie Web, per la creazione di applicazioni per dispositivi mobile, come alternativa allo sviluppo di applicazioni tramite linguaggi nativi.

Tra i vari dispositivi mobile esistenti, quello che trova maggior interesse nello sviluppo della tesi è sicuramente lo smartphone, il più recente tra questi dispositivi, che rispetto agli altri dispositivi è caratterizzato da una maggior complessità dovuta a funzionalità e capacità più elevate. Quindi in questa tesi verrà analizzato l'aspetto legato alle applicazioni utilizzate da questo dispositivo mobile.

Si è deciso di strutturare la tesi in diversi capitoli, che verranno illustrati qui in seguito, al fine di creare un percorso concettuale per arrivare ad analizzare l'argomento chiave della tesi, le applicazioni mobile basate su tecnologie Web, perciò nei primi due capitoli verrano trattati argomenti riguardanti il mondo dei dispositivi mobile, con particolare riguardo per lo smartphone, con lo scopo di dare una visione dell'ambiente che circonda questo argomento di tesi. Negli ultimi capitoli si entrerà nel cuore della tesi, dove verrà trattato l'argomento chiave nel dettaglio, con la specifica analisi di Tizen come caso di studio. Inoltre si è deciso di approfondire gli aspetti legati a questo argomento di tesi sviluppando un piccola applicazione, con lo scopo di andare a sperimentare le nozioni acquisite durante tutto questo percorso di studio.

Capitolo 1

Introduzione ai dispositivi mobile

Questo primo capitolo introduttivo punta a fornire una visione del mondo legato ai dispositivi mobile, così da porre le basi per iniziare una trattazione specifica sull'argomento di interesse della tesi. Le informazioni raccolte in questo capitolo sono state ricercate e rielaborate da diverse fonti di informazione [20] [19] [12] [13].

1.1 Storia

Oggi giorno la tecnologia ha portato l'uomo a sviluppare sepre più avanzati dispositivi, che gli permettono di soddisfare il suo bisogno di comunicare e scambiare informazioni per questioni di carattere sociale e lavorativo. In questi ultimi anni abbiamo avuto sotto gli occhi uno sviluppo frenetico e costante della tecnologia, che è stata sempre più alla portata del pubblico, con innovazioni soprattutto nel campo mobile.

Se si pensa all'invenzione del primo telefono fisso, che fu accreditato a Alexander Graham Bell, ma brevettato provvisoriamente da Meucci nel 1871, solo dopo circa 100 anni si è riusciti ad ottenere il primo dispositivo mobile, il telefono cellulare inventato da Martin Cooper direttore del reparto ricerca e sviluppo della Motorola, che fece la prima telefonata nel 3 aprile 1973. Ci vollero una decina di anni prima che si avesse uno sviluppo capillare da parte di questa tecnologia, indubbiamente a causa degli alti costi che lo ponevano in una categoria di bene di lusso. Ma dopo questo primo

periodo di assesto, questa tecnologia ebbe uno sviluppo molto rapido, questi dispositivi aumentarono progressivamente e molto rapidamente le loro funzionalità e migliorarono le loro caratteristiche fisiche quali dimensione, design, durata della batteria, potenza del segnale ...

Per introdurre l'aspetto di connettività alla rete internet, non ci volle molto tempo, già nel 1996 il Nokia Communicator 9000 fu il primo telefono cellulare a consentire un accesso ad internet. Successivamente nel 1999 naque in Giappone il primo servizio commerciale di web mobile basato su browser. Nel 2005 è stato introdotto il nome di dominio di primo livello .mobi, che fu dedicato specificatamente ai siti per dispositivi mobile.

Anche nel campo dei dispositivi mobole legati al concetto di computazione e gestione dei dati si ebbe un evoluzione significativa, infatti nel 1986 nasce il primo palmare, un dispositivo mobile chiamato anche PDA ideato come un computer portatile di dimensioni ridotte e con le funzionalità basilari di un calcolatore. Un importante aspetto di questo dispositivo è il concetto di connettività, per l'esigenza di sincronizzare dati con personal computer, tramite tecnologie quali collegamento a infrarossi, collegamento seriale, USB, Bluetooth. La funzionalità di connessione wirless alla rete fu associata anche a questo dispositivo.

L'unione del cellulare al dispositivo PDA diede vita all'ormai famoso Smartphone, telefono intelligente. Questo dispositivo segnò una rivoluzione nel concetto di telefonia mobile, perché da quel monento un solo dispositivo mobile abbinò le funzionalità di telefono cellulare a quelle di gestione di dati personali.

Il primo smartphone nasce nel 1992, chiamato Simon e progettato dall'IBM. Questo nuovo dispostivo è l'anello di congiunzione tra semplice dispositivo di telefonia e computer, poiché introduce concetti quali le applicazioni, il multitasking e la gestione delle risorse del dispositivo, concetti che prima erano proprietà esclusive dei computer.

Dai più semplici dispositivi, che permettevono di trasmettere segnali vocali da una postazione fissa siamo passati ad avere dei dispositivi sempre più piccoli, in grado non solo di creare uno scambio di segnali vocali, ma anche di scambiare messaggi di testo, immagini e con il passare del tempo hanno acquisito anche la capacità di incorporare funzionalità, quali la connessione alle reti internet, che permette di aumentare notevolmente le capacità di questi piccoli dispositivi portatili, di scambiare dati e reperire informazioni su più larga scala e con maggiore velocità e facilità. Infatti

questi dispositivi mobile, col passare del tempo assomigliano sempre più a veri e propri calcolatori, sia a livello hardware, sia per funzionalità, sia per grafica.

1.2 Dispositivi Mobile

In quests sezione si vuole dare una visione dei principali dispositivi mobile, presenti oggi sul mercato, per delimitare i confini del mondo nel quale ci stiamo per addentrare. Esistendo varie tipologie di dispositivi mobile è utile descrivere le varie caratteristice e funzionalità di ogni dispositivo, per poter capire al meglio le differenze tra questi prodotti tecnologici, che ci circondando ogni giorno. Conoscere le differenze e le potenzialità di ogni dispositivo è essenziale per avvicinarsi vantaggiosamente a queste tecnologie mobile, cosi da evitare di compiere scelte sbagliate quando ce ne vogliamo servire.

1.2.1 Cellulari

Il cellulare o telefonino, chiamato più correttamente telefono cellulare è un apparecchio radio mobile, che si collega alla rete telefonica fissa e alla rete dati tramite centrali di smistamento. Questo dispositivo permette di avere un collegamento telefonico, quando ci si trova all'interno di aree coperte dal raggio di una o più stazioni radio base a cui ci si può appoggiare, per ricevere questo servizio. La disponibilità di questo servizio, per meglio dire la potenza o la presenza di copertura è misurata in base a ben noti livelli di campo, che noi chiamiamo comunemente tacche.

1.2.1.1 Caratteristiche e funzionalità

Il cellulare ha una serie di caratteristiche che variano in base al produttore e al modello a cui ci riferiamo. Per quanto riguarda la struttura fisica, ne esistono di varie tipologie, come quelli classici monoblocco, quelli con lo sportellino, a scorrimento ... La batteria ricopre un ruolo importante, poiché in questi dispositivi il consumo energetico è un aspetto sempre critico, che richiede una buona gestione da parte del produttore per assicurare un prodotto di qualità.

I telefoni cellulari possono adottare diverse tecnologie per l'immissione dei dati e l'iterazione con l'utente, infatti tra le varie tecnologie utilizzate troviamo schermi tattili (Touchscreen), tastiere compatte o estese, in alcuni casi questi dispositivi adottano entrambe le tecnologie di input. L'antenna è la risorsa più importante di questo dispositivo e può essere interna, esterna o periferica. In fase di progettazione è importante il posizionamento di questo componente, per garantire un corretto funzionamento e rispettare tutti i criteri di affidabilità della ricezione.

Questo dispositivo può fornire vari servizi, che dipendono da modello a modello, alcuni basilari come l'invio di un SMS e il collegamento con gli altri dispositivi cellulari, altri che possono essere scelti dal produttore o derivanti dalla tecnologia che incorpora il cellulare, come l'invio di MMS, la video chiamata, fotocamera, videocamera, GPS, radio, televisione, ecc ...

L'evoluzione tecnologica di questo dispositivo ha portato ad un aumento dei componenti integrati, un aumento delle prestazioni, delle risorse come la memoria, ma di particolare interesse è l'evoluzione che ha caratterizzato l'aspetto della connettività, riguardante sia la rete telefonica, che la rete internet. Infatti tutti capiscono intuitivamente cosa può essere una fotocamera o una videochimata, ma se si parla di 3G, pochi sono a conoscenza di cosa sia, perciò è di maggiore interesse andare ad analizzare la parte riguardante le connessioni che effettua il dispositivo sulla rete.

Di seguito illustriamo i passi evolutivi fondamentali che hanno caratterizzato la crescita di questo dispositivo mobile.

Tecnologia 1G

Come si evince dal nome stiamo parlando della primissima tecnologia che sfruttavano i cellulari per comunicare tra di loro. La rete 1G prevede uno scambio di messaggi secondo uno standard analogico e ha utilizzato vari protocolli di comunicazione come l'AMPS, il TACS e il ETACS. Una grossa problematica di questa tecnologia era la sicurezza, che era a livelli molto bassi e quindi rendeva possibile, intercettazioni e hacking.

Lo standard analogico richiedeva apparecchi di grosse dimensioni, per via delle dimensioni del componente elettronico adibito alla ricezione/trasmissione del dispositivo e come si può intuire facilmente, una batteria di dimensioni elevate per far fronte agli alti consumi dell'apparato hardware. L'evoluzione impone la riduzione delle dimensioni e dei consumi, quindi questa tecnologia

era destinata ad essere superata, infatti divenne obsoleta con la comparsa dello standard GSM, che portò all'utilizzo della tecnologia 2G.

Tecnologia 2G

Questa seconda generazione di tecnologia per la connettività dei cellulari è stata commercializzata nel 1991 con l'introduzione dello standard GSM. La differenza principale e più grande dalla precedente generazione, consiste nel fatto che si è abbandonata la rete analogica per passare ad una rete digitale.

Questo ha portato ovviamente ad importanti benefici quali:

- Completa cifratura delle trasmissioni nell'etere, per previnire le intercettazioni.
- Miglior efficienza spettrale.
- Possibilità di usufruire di servizi dati, come gli SMS.

L'utilizzo di segnali digitali incrementa le capacità del sistema, poiché il segnale vocale viene compresso più efficentemente e questo porta alla condivisione da parte di un numero maggiore di chiamate della stessa banda, per non parlare della minor quantità di potenza necessaria per la trasmissione e tramite i filtri digitali è possibile avere una qualità del suono maggiore.

L'utilizzo della tecnologia GSM nacque in Europa ed è stata vincente e ha riscosso gran successo in tutto il mondo servendo oltre l'80 % degli utenti di telefonia mobile.

La rete 2G si è successivamente evoluta in due versioni, la 2.5 e la 2.75, che ne hanno aumentato notevolmente le prestazioni, poiché le nuove versioni hanno adottato evoluzioni della tecnologia GSM, che sono rispettivamente la GPRS e la EDGE.

Tecnologia 3G

È la terza generazione di tecnologia per la connettività e standard per i telefoni cellulari, che introduce innovative funzionalità, infatti altre al traffico dati voce, si ha la presenza di dati non voce, che sarebbero quelli riguardanti il download di dati via internet, invio e ricezione di mail, ecc ...

Il primo paese ad introdurre questa tecnologia fu il giappone nel 2005. Questa tecnologia è stata implementata in diversi standard:

1. **UMTS (W-CDMA)** che è lo standard principale per paesi europei, dove viene utilizzato il GSM. Di fatti è l'evoluzione diretta dello standard GSM, con maggiore velocità di trasferimento, teoricamente fino a 21Mb/s, dovuto all'accesso multiplo al canale di tipo W-CDMA.

2. **CDMA 2000** evoluzione del CDMA e usato principalmente in America, Giappone e Korea.

3. **TD-SCDMA** meno noto degli altri fu sviluppato in Cina da Datang e dalla Siemens.

Tecnologia 4G

Ultima generazione, che incorpora nuove tecnologie e standard studiate per l'utilizzo di applicazioni multimediali avanzate e collegamenti dati ad elevata banda passante. Questa nuova generazione è ancora in fase di sviluppo e le norme per la costruzione di questa futura rete sono affidate ad un gruppo di società chiamato NGMN (Next Generation Mobile Networks). Questa nuova rete sarà basata su un'architettura a commutazione di pacchetto, che consentirà una traslazione verso una rete IP che andrà a sostituire le vecchie 2G e 3G.

Altre tecnologie

Senza stare troppo a dilungarsi nei dettagli su tutte le altre forme di connessione, che possono essere adoperate da questi dispositi, mi limiterò a citarle con lo scopo di cercare di dare una visione generale sulle tecnologie che circondano questi dispositivi.

- **Wirless Application Protocol** - conosciuto come WAP, protocollo utilizzato per la connessione in internet, ideato da Alain Rossman.

- **I-Mode** - un servizio molto utilizzato in Giappone per il collegamento ad internet, un protocollo per lo scambio di dati a pacchetto su piattaforme mobile.

- **Voice over IP** - conoscito come VoIP, che consente di effettuare telefonate sfruttando la connessione internet o qualsiasi rete dedicata a commutazione di pacchetto, che utilizzi il protocollo IP senza connessione per il trasporto dati.

- **Unlicensed Mobile Access** - conosciuto come UMA, che consente ai telefoni dual mode, di trasferire in modo trasparente all'utente, una conversazione da rete GSM/UMTS alla rete Wi-Fi.

Questi dispositivi offrono inoltre varie tecnologie per connettersi tra di loro, ad altri dispositivi mobile o a computer, senza l'appoggio della rete, quali Bluetooth, Wi-Fi, Infrarossi.

Figura 1.1: Dispositivo Mobile: Cellulare

1.2.2 Tablet

Questo dispositivo conosciuto anche come Tablet Puro o Tablet PC è caratterizzato da dimensioni compatte ed utilizza come unico sistema di input lo schermo, tramite l'utilizzo di una penna o delle dita, senza così aver bisogno

di device esterni come la tastiera o il mouse, detto in maniera più semplice è un dispositivo unicamente Touch.

Il primo tablet fece la sua comparsa nel 2000 grazie a Microsoft che presentò Concept, che si basa su Windows XP ed tuttoggi il suo sviluppo è ancora in atto, l'ultima tipologia di tablet infatti è stata rilasciata nel 2011 con il primo tablet ibrido tra la prima e la seconda generazione di tablet.

1.2.2.1 Caratteristiche e funzionalità

Un tablet si può vedere come una via di mezzo tra un cellulare ed un computer, infatti permette di poter essere usato come un telefonino e usare le funzionalità wirless per i browser come il 2G, 3G, 4G e Wi-Fi, di effettuare videochiamate e di usare applicazioni per accedere a Social Network, per gestire caselle di posta elettronica. Può anche essere usato come navigatore GPS, per leggere e-book, per scaricare applicazioni come giochi o programmi di utilità, ecc ...

Concludendo è un dispositivo ricco di funzioni, con alte potenzialità.

Possiamo inserire tutti i tablet in 3 inisiemi di generazioni, che segnano l'evoluzione di questo dispositivo in questo decennio.

Prima generazione

I tablet appartenenti a questa generazione sono stati ideati partendo dal modello del personal computer, infatti montano architetture Intel x86, con un sistema operativo per pc, generalmente Windows XP Tablet PC Edition, che viene adattato per l'utilizzo dello schermo come sistema principale di input.

Questi dispositivi hanno una connettività cablata e Wi-Fi molto simile a quella di un Pc fisso o di un portatile, inoltre hanno molte funzionalità legate a sistemi operativi per computer, come ad esempio è possibile collegargli una chiavetta USB per la navigazione in internet.

Questa generazione è caratterizzata da una elevata compatibilità con Personal Computer e relative periferiche.

Seconda generazione

I tablet di questa generazione hanno debuttato nel 2010 sono basati

sul modello degli Smartphone, infatti utilizzano processori con architettura ARM e montano sistemi operativi tipici degli Smartphone, quali Android, iOS, ecc ...

È importante denotare che questa seconda generazione di tablet può essere suddivisa in due macro-insiemi, per quanto riguarda la caratteristica di connettività, poiché per collegarsi alla rete hanno due modalità ben differenti per approcciarsi a questo aspetto. Questo dipende da come il costruttore ha deciso di progettare un dato modello di tablet.

- Tablet dotati di connettività mobile possono utilizzare una SIM Card per connettersi alla rete internet, per effettuare e ricevere telefonate e SMS.

- Tablet dotati di connettività Wi-Fi, che per accedere alla rete internet sfruttano una tecnologia chiamata tethering, che consiste nell'accesso alla rete internet tramite un collegamento ad uno smartphone utilizzando la connettività del dispositivo a cui si è connessi. In parole povere questa tecnologia consente di usare una risorsa esterna proveniente da un altro dispositivo. Questa soluzione è attualmente utilizzata dai diversi prodotti come per esempio i tablet BlackBerry.

Queste due soluzione hanno relativi vantaggi e svantaggi.

I dispositivi con una connettività mobile natia sono completamente indipendenti, ma essendo definito a priori l'hardware, la tecnologia legata alla connettività non evolve nel tempo. I dispositivi che fanno uso della connettività Wi-Fi e del tethering possono ottenere aggiornamenti relativi alla tecnologia di connettività e di conseguenza anche dell'aggiornamento del dispositivo.

Questi dispositivi sono caratterizzati da una scarsa compatibilità con i Pc e le relative periferiche.

Terza generazione

I Tablet di questa generazione sono degli ibridi derivanti dall'unione delle precedenti generazioni. Questi dispositivi sono caratterizzati da due o più sistemi operativi, che permetteno così di sfruttare funionalità software delle precedenti generazioni e anche dal lato hardware sono previsti accessori come tastiere removibili.

Il primo tablet di questa nuova generazione è Microsoft Surface, che monta un unico sistema operativo, generalmente Windows RT o Windows 8 Pro.

Figura 1.2: Dispositivo Mobile: Tablet

1.2.3 PDA

Questo dispositivo conosciuto meglio con il nome di Computer Palmare o con il desueto termine palmtop è di fatto un computer di piccole dimensioni, ideato per essere tenuto sul palmo della mano. Originariamente era concepito come agenda elettronica, con la classica presenza di funzioni di utilità quali orologio, calcolatrice, calendario e rubbrica. La sua evoluzione è iniziata nel 1986, quando uscì il primo palmare Organizer II prodotto da Psion. Il termine PDA fù adottato nel 1992, quando l'evoluzione di questi dispositivi li portò ad essere sempre più simili a quelli che conosciamo oggi giorno.

1.2.3.1 Caratteristiche e funzionalità

Questi dispositivi sono dotati della capacità di collegarsi e sincronizzarsi con i Pc, tramite tecnologie quali infrarossi, collegamento seriale, USB o Bluetooth. Nello specifico quest'ultima tecnologia (il Bluetooth) è stata introdotta con lo scopo di permettere un collegamente a dispositivi esterni

come telefoni cellulari e GPS per aumentare l'usabilità e le potenzialità di questo dispositivo.

Sui PDA si possono installare diversi tipi di software, per aumentarne le funzionalità, come ad esempio fogli elettronici, client di posta elettronica, lettori MP3, giochi, ecc ... Ultimamente i PDA stanno diventando sempre più potenti ed accessoriati, infatti gli ultimi modelli integrano le funzionalità di connessione telefonica sfruttando le principali tecnologie usate in questo campo come GSM, GPRS, EDGE, UMTS, HSPDA.

Il maggiore limite di questi dispositivi è riscontrato nella risorsa hardware riguardante la memoria disponibile, che raramente supera i 128 MB. Questo limite si sta cercando di risolvere e superare facendo ricorso all'utilizzo di memory card ed alcuni produttori hanno lanciato sul mercato dei dispositivi con un hard disk interno con capacità variante tra i 2 e gli 8 GB.

Per quanto riguarda i sistemi operativi utilizzati dai PDA, si può notare una stretta affinità con i sistemi operativi utilizzati dagli smartphone, infatti i PDA montano sistemi quali:

- Android
- iOS
- Palm OS
- Windows Mobile
- BlackBerry OS
- GNU/Linux
- Symbian

Come ci si può aspettare, la maggior parte dei PDA esistenti sono dotati di una connessione wirless e di un browser, che gli permette di navigare su internet e di trarre tutti i benefici derivanti dalla possibilità di collegarsi alla rete.

Figura 1.3: Dispositivo Mobile: PDA

1.3 Caso di interesse: Smartphone

Uno smartphone piò essere visto come l'unione di un PDA ad un telefono cellulare e viceversa.

Questo dispositivo abbina le funzionalità di un telefono cellulare a quelle di gestione di dati personali tramite l'utilizzo di software, chiamati apllicazioni o in gergo App, che sono strutturate ad hoc per venire incontro alle più svariate necessità e ai servizi che possono interessare gli utenti di questi dispositivi mobile. Il cuore di questo dispositivo sono per l'appunto le App, che rivestono un ruolo di principale interesse, per tutti gli utilizzatori di questo dispositivo mobile, fino a tal punto che esiste un vero e proprio mercato che ruota attorno a questo aspetto ed offre migliaia di diverse applicazioni gratuite o a pagamento.

1.3.1 Confronto con i telefoni cellulari

Con l'introduzione di questi dispositivi, si è avuto un notevole aumento delle performance e delle funzioni rispetto ai telefoni cellulari delle precedenti generazioni, grazie all'utilizzo di processori sempre più potenti e simili a quelli dei Pc e a memorie molto più capienti. Questo importante investimento in

termini di hardware è stato necessario per supportare l'evoluzione di tutte quelle funzionalità che i telefoni cellulari non avevano e per rendere questo disposito sempre veloce ed agevole nell'utilizzo.

1.3.2 Caratteristiche e funzionalità

Le funzionalità di questo dispositivo sono molteplici, dato che sono di fatto l'unione di PDA a telefoni cellulari, perciò non starò a dilungarmi troppo sulle funzionalità basilari presenti su questi dispositivi, ma è importante cirtarne alcune degne di nota.

Oggigiorno gli smartphone utilizzano connessioni GSM, GPRS, EDGE, UMTS, HSPDA, HSUPA, LTE ed utilizzano tecnologie Wi-Fi e Bluetooth per comunicare con altri dispositivi. Possono aumentare le proprie funzionalità installando applicazioni ufficiali o di terze parti come software di qualsiasi utilità, giochi, temi, ecc ... Alcune funzionalità aggiuntive possono essere sfruttate tramite l'uso di sensori presenti nel dispositivo, come ad esempio accellerometro, magnetometro, sensore di prossimità, ecc ... Alcuni smartphone offrono anche la possibilità di tethering in Wi-Fi verso altri dispositivi mobile, portatili, Pc fissi. Và comunque specificato che tutti gli smartphone, come la maggior parte dei telefoni cellulari permettono il tethering cablato, ossia effettuare il tethering attraverso un collegamento fisico dei dispositivi, tramite l'utilizzo di cavi USB, FireWire o HDMI.

Per quanto riguarda il sistema operativo, questi dispositivi ne possono adottare diversi, che variano in base alle scelte del produttore, quelli più utilizzati sono:

- Android
- Apple iOS
- Windows Phone
- BlackBerry OS
- Symbian OS

La caratteristica che differenzia maggiormente questi dispositivi mobile, dagli altri citati precedentemente, consiste nei componenti hardware che

utilizzano, infatti per supportare tutte queste funzionalità garantendo comunque una buona efficienza, si adoperano componenti hardware di alto livello, infatti guardando un generico smartphone di fascia media presente oggi sul mercato possiamo notare che questi dispositivi mobile sfruttano processori multi-core, in genere dual-core, con velocità molto elevate nell'ordine di 1.2 GHz, si avvalgono di memorie per il salvataggio dei dati da 8-16 GB, più tutto il set dei vari sensori che incorporano. Tutto questo utilizzo di risorse hardware di alto livello porta con sé problematiche critiche legate al consumo di energia.

Figura 1.4: Dispositivo Mobile: Smartphone

1.3.3 Sistemi operativi per Smartphone

I sistemi operativi per dispositivi mobile quali gli Smartphone sono sistemi operativi che controllano il dispositivo mobile, con lo stesso principio con cui sistemi come Windows, Mac OS, Unix controllano computer desktop o laptop. Tuttavia questi sistemi si trovano ad affrontare problematiche maggiori, che sono legate a vari fattori come risorse di memoria e di processore, che sono molto più limitate in questi dispositivi. Altri fattori sono le diverse tecnologie di connessione alla rete, che sono state citate precedentemente, i nuovi metodi d'immissione dei dati come il touchscreen o le minitastiere e anche le ridotte dimensioni dei display.

Proseguiamo dunque all'analisi dei principali sistemi operativi per gestire questo dispositivo mobile.

1.3.3.1 Android

Android è un sistema operativo open source per dispositivi mobile, di proprietà di Google. Nonostante sia un sistema open source, l'influenza di Google all'interno del sistema si nota molto, infatti molti aspetti del sistema sono ottimizzati per l'iterazione con prodotti Google.

Questo sistema è basato su un kernel Linux ed è caratterizzato da una struttura open source con licenza Apache, che gli permette di modificare e distribuire liberamente il codice sorgente.

Caratteristiche tecniche

Android è basato su kernel Linux 2.6 e dalla versione Android 4.0 da kernel Linux 3.x . A livello hardware, Android utilizza un'architettura ARM, ma grazie alla nascita del progetto Android x86 ideato per il supporto alla Google Tv, può anche utillizzare l'architettura x86. Al di sopra del kernel risiedono set di librerie conteneti API scritte in C e C++ e un middleware. La parte software del sistema è in esecuzione su un framework contenente librerie Java basate su Apache Harmony. Android sfrutta tecnologie quali un database SQLite e un supporto alla grafica tramite SGL e OpenGL ES 2.0 e monta un motore grafico basato sulla tecnologia WebKit. Questo sistema è dotato di una memoria flash suddivisa in partizioni, tra le quali troviamo la partizione system per i file di sistema e altre per gestire al meglio il sistema.

Android è molto simile alle distribuzioni per computer desktop di Linux, anche per quanto riguarda la gestione dei permessi, infatti l'utente di questo sistema, non avrà mai i privilegi di superutente, per motivi di sicurezza principalmente, però potrà comunque accedere ai dati solo per la lettura di informazioni. È importante sottolineare che l'accesso come root (superutente), può comunque essere effettuato sfruttando le falle del sistema, ma questo viene spesso usato dalla comunità open source per migliorare le capacità dei propri dispositivi. La comunità XDA, si è dedicata al rilascio di Firmware, che possono essere installati sul dispositivo, per aumentare i permessi e poter accedere a funzionalità avanzate e gestire risorse come CPU e App di sistema.

Il porting di applicazioni Linux o librerie per Android risulta difficoltoso, poiché un tipico sistema Android non possiede X Windows System nativo, non supporta il set completo standard di librerie GNU e nel caso del C++ vi è solo una implementazione parziale delle STL. L'interfaccia utente è basata sul concetto di direct manipulation, che permette di manipolare oggetti presenti sullo schermo, tramite input mono/multi-touch. La homescreen è simile al desktop di un computer e solitamente ha un tasto associato per potervi accedere da qualsiasi scermata in cui si è arrivati, di fatti è anche la scermata che ci si presenta all'avvio del nostro dispositivo. Questa schermata può essere suddivisa in varie sezioni e contiene icone e widget delle varie applicazioni installate sul dispositivo, personalizzabili a vari livelli dall'utente. Android utilizza la Dalvik Virtual Machine con un compilatorie just-in-time per l'esecuzione di Dalvik Dex-Code (gli eseguibili), che vengono tradotti da codice bytecode Java.

La Dalvik Virtual Machine - La Dalvik Virtual Machine è in grado di eseguire il codice contenuto all'interno di file di estensione .dex ottenuti a partire dal byte-code Java. I dispositivi Android utilizzano questa Virtual Machine al posto della Java Virtual Machine per via della necessità di risparmiare quanto più spazio possibile, per la memorizzazione ed esecuzione delle applicazioni. Infatti un file .dex ha dimensioni ridotte rispetto ad un file .jar di Java, si parla di circa il 50 % di risparmio, che è una delle prime ottimizzazioni introdotte rispetto all'essecuzione di byte-code Java. La Dalvik Virtual Machine continua ad avvalersi del Garbage Collector le la gestione della memoria, inoltre dalle versione 2.2 implementa il Just-In-Time compiler, che si tratta di un meccanismo di compilazione chiamato anche traduzione dinamica, attraverso al quale la Dalvik Virtual Machine riconosce determinati pattern di codice Java e li traduce in frammenti del codice nativo C/C++ durante l'esecuzione, per aumentare le prestazioni e rendere l'applicazione più efficiente. Un aspetto molto importante della Dalvik Virtual Machine riguarda il meccanismo di generazione del codice, che viene detto register based a differenza di quello della Java Virtual Machine, che viene detto stack based. Questa strategia permette di ridurre del 30 % circa il numero di operazioni da eseguire, che permette di ottenere un minor tempo di esecuzione delle istruzioni, al prezzo di una maggiore elaborazione in fase di compilazione o trasformazione. Ulteriore vantaggio della Dalvik Virtual Machine è quella di permettere un'efficace esecuzione di più processi

in contemporanea, infatti ogni applicazione lanciata in esecuzione sarà rappresentata come un processo Linux avente una Virtual Machine associata, così da rendere ogni processo indipendente dagli altri, compresi quelli di sistema. Questo approccio porta ad avere vantaggi a livello di prestazioni.

Architettura

È basato su un'architettura che comprende tutto un set di elementi per l'esecuzione di applicazioni mobile di ultima generazione, tra cui un sistema operativo, un insieme di librerie native per le funzionalità di sistema della piattaforma, un'implementazione della Virtual Machine e un insieme di librerie Java. Si tratta di un'architettura a layer dove i livelli inferiori offrono servizi a quelli superiori, con più alto livello di astrazione [22].

Questo sistema operativo è formato da un'architettura suddivisa in 4 livelli, come mostrato in figura 1.5, che andiamo ad analizzare partendo dal livello più vicino all'utente, per poi proseguire avvicinandosi sempre più al livello hardware.

Applications Layer - In questo livello risiedono tutte le applicazioni per l'utente, infatti troviamo un insieme di applicazioni base che permettono all'utente di sfruttare le funzionalità del sistema Android. Si trovano applicazioni come calendario, mappe, browser, contatti, ecc ... Le applicazioni presenti su in questo livello consentono di utilizzare funzionalità come ad esempio mandare email attraverso un client, chiamare e mandare SMS. Potrà essere arricchito successivamente con applicazioni che l'utente deciderà di aggiungere al dispositivo, tramite opportuni canali di download, per aumentarne le capacità o per semplice svago.

Application Framework Layer - In questo livello troviamo tutti i gestori di applicazioni e risorse base del sistema, che hanno il compito di mettere in atto ciò che viene richiesto dalle applicazioni del livello superiore.

Al fine di compiere il loro lavoro, questi gestori sono in possesso di un set di API, che possono essere liberamente utilizzate dagli sviluppatori nelle applicazioni che si intende creare. Tra i gestori più importanti che possiamo trovare in questo livello sono degni di nota i seguenti gestori.

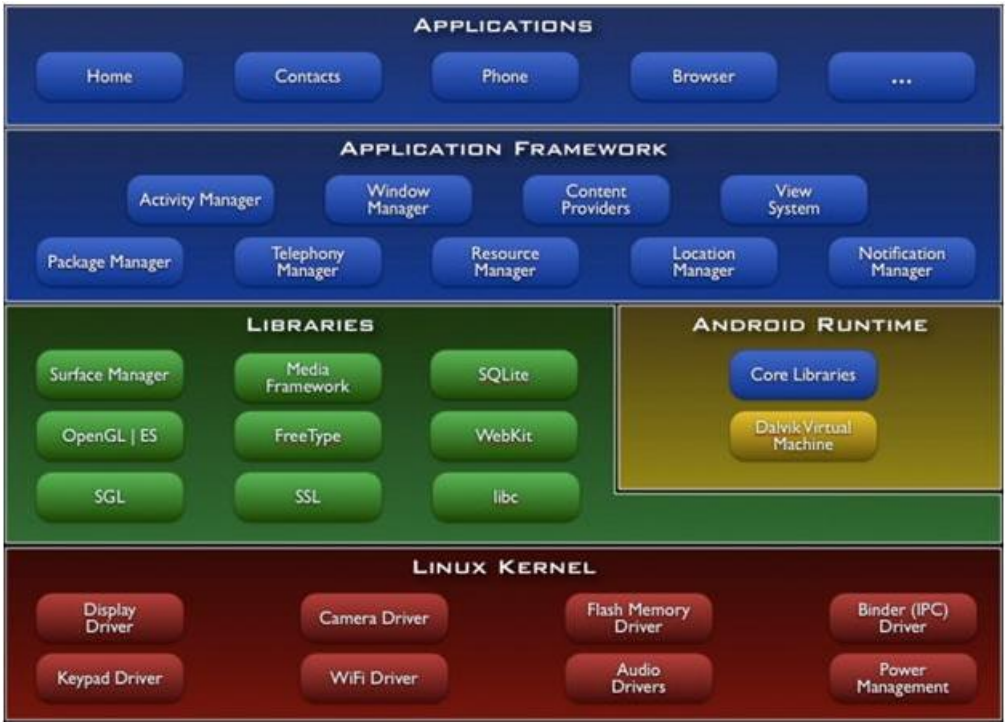

Figura 1.5: Architettura sistema operativo Android [1].

- **Activity Manager** - ha il compito di gestire il ciclo di vita di ogni Activity mandata in esecuzione sul dispositivo. Il ciclo di vita di ogni Activity dopo la propria creazione è composta da quattro stati in cui si può trovare in un determinato istante. L'Activity potrà essere in esecuzione, in pausa, fermata o distrutta.

- **Content Providers** - ha il compito di gestire la condivisione dei dati tra le varie applicazione in esecuzione sul dispositivo. Questo componente ha a disposizione varie funzioni che può utilizzare per gestire al meglio tutti i dati che dovranno essere condivisi, come Insert(), Update(), Delete(), Query().

- **View System** - ha il compito di gestire la parte riguardante le interfacce associate alle applicazioni in esecuzione, che verranno gestite tramite una struttura gerarchica ad albero singolo.

- **Resource Manager** - si occupa di gestire tutti i dati che vengono utilizzati dalle applicazioni che non sono di tipo codice, come ad esempio immagini, suoni, testi, ecc ...
- **Telephony Manager** - che come si evince dal nome gestisce tutta la parte riguardante le chiamate vocali del dispositivo.
- **Location Manager** - si occupa di gestire la localizzazione dell'applicazione, tramite le risorse che sono a disposizione nel dispositivo, quali il GPS, il provider locale, ecc ...

Librariers Layer - In questo livello risiedono tutte le librerie native del sistema scritte in C/C++, che contengono al loro interno vari set di istruzioni per utilizzare le funzionalità base, al fine di gestire le risorse che offre il dispositivo. Alcune di queste librerie sono:

- **Surface Manager** - contente le funzionalità per la composizione di finestre sullo schermo.
- **SGL** - libreria per la grafica 2D.
- **Open GL-ES** - libreria per la grafica 3D.
- **Media Framework** - contiene tutte le informazioni per registrare e riprodurre formati video, audio e immagini.
- **Free Type** - libreria dedicata al rendering dei font.
- **WebKit** - libreria dedicata ai browser per il rendering delle pagine web.
- **Libc** - libreria scritta in linguaggio C, con il compito di aumentare le prestazioni del sistema.

In questo livello si può individuare un insieme di librerie, che è chiamato Android Runtime Librariers scritte in linguaggio Java. In questo insieme di librerie risiede la Dalvik Virtual Machine e un set di librerie di sistema. La Dalvik Virtual Machine è il componente principe del sistema operativo Android, poiché è il corrispettivo della Java Virtual Machine per i software scritti in linguaggio Java.

Linux Kernel Layer - Questo livello è quello più vicino alla parte hardware del dispositivo ed è il cuore del sistema operativo, dove svolge funzioni vitali per il funzionamento di quest'ultimo. Il Kernel Linux sopperisce alla necessità di fornire gli strumenti di basso livello per la virtualizzazione dell'hardware sottostante attraverso la definizione di diversi driver.

In questo livello troviamo vari componenti che svolgono funzioni come:

- Astrazione delle risorse hardware.
- Gestione della memoria.
- Impostazioni di sicurezza.
- Gestione della alimentazione.
- Driver Hardware, programmi per controllare le risorse hardware del dispositivo.

1.3.3.2 iOS

iOS è un sistema operativo proprietario sviluppato da Apple, che fà la sua comparsa nello scenario mobile il 9 gennaio 2007 ed essendo proprietario, come tutti i prodotti Apple può essere mandato in esecuzione solo su dispositivi mobile prodotti da quest'ultima, perciò questo sistema operativo potrà essere utilizzato solo da dispositivi iPhone, iPad, iPod e Apple TV.

Caratteristiche tecniche

Questo sistema operativo è una derivazione di Unix, com'è stato per il sistema operativo Mac OS per i Macbook ed utilizza un mircokernel XNU Mach basato sul sistema open source Darwin OS.

Il kernel di iOS è particolare, poiché è un ibrido composto da un microkernel XNU Mach unito ad un kernel monolitico FreeBSD, dove il primo si occupa di gestire la memoria, i dati di input e output, la comunicazione tra i processi del sistema, il multitasking e la gestione della memoria virtuale, mentre il kernel monolitico si occupa di gestire gli utenti con i relativi permessi, il virtual file system, lo stack di rete. Questo dispositivo solo dalla verisone 4.0 è stato dotato di un sistema di multitasking, per far fronte al consumo energetico del dispositivo.

Inizialme il sistema operativo iOS era stato progettato con un'architettura a 32 bit, ma dalla versione iOS 7, l'architettura del sistema è passata a 64 bit, questo ha comportato una trasformazione di tutte le librerie e dei framework a 64 bit, che possono essere utilizzati sia da applicazioni a 32 che da 64 bit. Questo sistema operativo supporta processori basati su architetture RISC ARM ed utilizza tecnologie quali OpenGL ES 1.1 e OpenGL ES 2.0, gestite da un processore video PowerVR. Rispetto agli altri sistemi operativi, iOS si circonda di numerose applicazioni base, già fornite dal dispositivo, per garantire una noteve User Experience. Questo sistema operativo integra il Safari web browser, per la navigazione e il supporto alle applicazioni Web.

Architettura

Questo sistema operativo ha un'architettura che è strutturata su quattro livelli, come mostrato in figura 1.6, che andremo qui di seguito a mostrare ed analizzare partendo dal livello più vicino all'utente, per poi proseguire verso il livello più vicino all'hardware del dispositivo.

Cocoa Touch Layer - In questo primo livello dell'architettura iOS troviamo una serie di tecnologie e framework, che offrono una serie di funzionalità importanti per l'iterazione del dispositivo con l'utente. Come per Cocoa di Mac OS X, questo livello è basato su un'architettura software Model-View-Controller. Questo livello incorpora un set di Cocoa API presenti anche su Mac OS X, principalmente scritte in linguaggio Objective-C, che offrono servizi di sistema ad alto livello.

Senza ulteriore indugio andiamo a presentare queste tecnologie e framework che il livello Cocoa Touch mette a nostra disposizione. In questo livello troviamo le seguenti tecnologie.

- **AirDrop** - è un componente che permette di condividere qualsiasi tipo di dato come foto, documenti e file, con gli altri dispositivi nelle vicinanze. Questo componente può essere utilizzato facendo rifermento alla classe UIActivityViewController.

- **Text Kit** - è un componente formato da un set di classi di alto livello, designate per la formattazione del testo e degli stili di scrittura, si occupa di gestire tutti i vari font utilizzati dalle applicazioni.

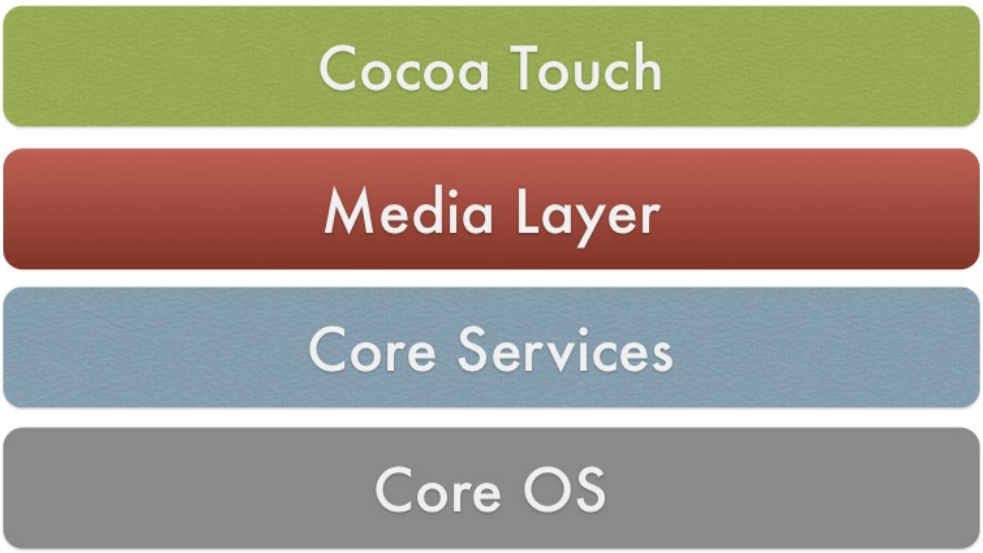

Figura 1.6: Architettura sistema operativo iOS [2].

- **UIKit Dinamics** - è un componente per la gestione dei comportamenti dinamici e delle caratteristiche legati alle interfacce delle applicazioni. Questo componente permette di utilizzare i seguenti comportamenti dinamici.
  - Attachment, classe UIAttachmentBehavior.
  - Collision, classe UICollisionBehavior.
  - Gravity, classe UIGravityBehavior.
  - Push, classe UIPushBehavior.
  - Snap, classe UISnapBehavior.
- **Multitasking** - è un componente molto importante per la gestione dei vari processi, con l'obbiettivo di minimizzare i consumi della batteria, che sono dovuti all'esecuzione di questi processi. Principalmente gestisce il passaggio delle applicazioni da un stato di foreground ad uno di background e quando un'applicazione non ha più istruzioni da esegure, ne viene sospesa l'esecuzione

andando a porre l'applicazione in uno stato di freeze-dried, che la porterà a non eseguire più istruzioni di codice, ma occuperà comunque una porzione di memoria, per un eventuale riutilizzo futuro.

- **Auto Layout** - è un componente che facilita la programmazione delle interfacce grafice specificando diverse regole per la visualizzazione degli elementi all'interno dell'interfaccia.
- **Storyboards** - è un componente che tiene traccia di tutte le interfacce facilitando la comprensione da parte dello sviluppatore, della loro iterazione.
- **UI State Preservation** - è un componente che ha il compito di preservare la memoria, che se dovesse giungere a livelli critici mette in moto un'operazione di terminazione delle applicazioni in eccesso, secondo regole associate alla priorità dei processi.
- **Apple Push Notification Service** - è un componente che ha il compito di mandare notifiche all'utente quando si verificano malfunzionamenti nell'esecuzione delle applicazioni.
- **Local Notification** - altro componente per le notifiche, indipendente delle applicazioni.
- **Gesture Recognizers** - è un componente che riconosce i diversi gesti che l'utente compie per interagire con il dispositivo.
- **Standard System View Controllers** - è un componente che produce view controller, per le varie view utilizzate dai framework presenti in questo livello.

Oltre alle tecnologie precedentemente citate, in questo livello risiedono importanti framework, che andremo ad illustrare qui in seguito.

- **Address Book UI Framework** - definisce le interfacce di sistema base, per la creazione e gestione dei contatti.
- **Event Kit UI Framework** - fornisce view controller per la gestione delle interfacce riguardanti il calendario e i relativi eventi associati.

- **Game Kit Framework** - supporta Game Center fornendo caratteristiche come aliases, classifiche, matchmaking, sfide, achievements.

- **iAd Framework** - gestisce la parte riguardante gli avvisi e le notifiche in formato banner.

- **Map Kit Framework** - si occupa della gestione dell'inserimento e dell'utilizzo di mappe all'interno delle applicazioni.

- **Message UI Framework** - supporta la composizione di messaggi SMS e di email all'interno delle applicazioni associando un view controller all'interfaccia dell'applicazione.

- **Twitter Framework** - evoluto in Social Framework, per supportare le interfacce delle applicazioni che interagiscono con i social network.

- **UIKit Framework** - fornisce le basi per implementare la grafica e le animazioni delle applicazioni.

Media Layer - Nel livello Media troviamo tutte le librerie per l'implementazione di funzioni per la gestione dei flussi video e audio del sistema, infatti utilizza librerie OpenAL per la gestione dell'audio e OpenGL ES per quanto riguarda la gestione del video. In questo livello troviamo anche il framework Quartz Core e le interfacce Core Animation, che permettono di creare animazioni ed effetti visivi. Gli aspetti audio e video vengono gestiti tramite l'utilizzo di diverse tecnologie e framework. Partendo dalla parte delle tecnologie presenti su questo livello, ci troviamo ad avere tecnologie grafiche, video e audio.

Iniziamo illustrando le tecnologie per la grafica.

- UIKit graphics.

- Core Graphics.

- Core Animation.

- Core Image.

- OpenGL ES and GLKit.
- Text Kit and Core Text.
- Image I/O.
- Assets Library.

Per quanto riguarda la parte video troviamo le seguenti tecnologie.

- Media Player.
- AV Foundation.
- Core Media.

Le tecnologie audio presenti in questo livello sono:

- Media Player.
- AV Foundation.
- OpenAL.
- Core Audio.

Tutte queste tecnologie hanno associato un framework per il loro utilizzo, ma risulta di particolare rilevanza la funzionalità dell'AirPlay. L'AirPlay permette di creare un flusso di dati audio e video tra il dispositivo mobile e la AppleTV utilizzando vari framework presenti in questo livello.

Core Services Layer - Il Core Service, si occupa di fornire importanti sevizi di sistema. I servizi offerti da questo livello riguardano il collegamento peer-to-peer, la condivizione di dati, la protezione dei dati, SQLite, supporto all'XML, l'Automatic Reference Counting (ARC), che semplifica la gestione del ciclo di vita degli oggetti Objective-C. Altro servizio importante è l'iCloud Storage, che permette di salvare i propri dati e documenti in modo centralizzato in una locazione remota, per potervi accedere da qualsiasi computer o dispositivo iOS, senza bisogno di sincronizzarsi o dover trasferire i file. Il vantaggio di questa tecnologia è che in caso di smarrimento, malfunzionamento, rottura del dispositivo, i dati non andranno persi e vi si potrà accedere da qualsiasi altro dispositivo iOS o computer.

Anche questo livello è ricco di framework, che citeremo di seguito.

- Accounts Framework.
- Address Book Framework.
- Ad Support Framework.
- CFNetwork Framework.
- Core Data Framework.
- Core Foundation Framework.
- Core Location Framework.
- Core Media Framework.
- Core Motion Framework.
- Core Telephony Framework.
- Event Kit Framework.
- Foundation Framework.
- JavaScript Core Framework.
- Mobile Core Services Framework.
- Multipeer Connectivity Framework.
- Newsstand Kit Framework.
- Pass Kit Framework.
- Quick Look Framework.
- Safari Services Framework.
- Social Framework.
- Store Kit Framework.
- System Configuration Framework.

Core OS Layer - Il Core OS rappresenta il cuore del sistema operativo, qui è dove vengono gestiti i file system e vengono implementate tutte le funzioni di sicurezza del dispositivo. Questo livello si occupa di tutti gli aspetti di gestione delle risorse, come la batteria, la CPU, la memoria e come si può immaginare, l'ottimizzazione di questo livello è necessaria.

All'interno del livello Core OS può essere individuato una sezione System, che include la parte riguardante lo sviluppo del kernel, i driver e le interfacce di basso livello UNIX del sistema operativo. Il System gestisce la memoria virtuale, i file system, i threads, la rete, le comunicazioni tra i vari processi.

In questo livello troviamo importanti framework per la gestione dei vari aspetti.

- **Accelerate Framework** - contiene interfacce per ottimizzare il processamento dei segnali digitali (DSP), l'algebra lineare e l'elaborazione di immagini.
- **Core Bluetooth Framework** - permette di interagire con gli accessori Bluetooth a basso consumo energetico.
- **External Accessory Framework** - supporta la comunicazione con gli accessori hardware collegati al dispositivo iOS gestendo le informazioni ed inizializzando la sessione di comunicazione.
- **Generic Security Services Framework** - si occupa dei servizi di sicurezza del dispositivo utilizzando interfacce, che seguono le specifiche definite nei moduli IETF RFC 2743 e RFC 4401.
- **Security Framework** - offre servizi di sicurezza per la protezione dei dati delle applicazioni presenti sul dispositivo, tramite interfacce per la gestione di certificati, chiavi pubbliche e private. Utilizza tecnologie di criptazione dei dati, presenti nella libreria Common Crypto.

1.3.3.3 Windows Phone

Windows Phone è il sistema operativo per smarthphone della Microsoft. Il lavoro per la creazione di questa nuova generazione del sistema Windows

Mobile è iniziata nel 2004, ma la prima versione è stata rilasciata il 15 Febbraio del 2010. La mancata retrocompatibilità con le versioni precedenti di Windows Mobile è stata dovuta al fatto che questo sistema è stato sviluppato molto velocemente, come ha dichiarato Terry Mayerson, ingegnere per lo sviluppo del sistema Windows Phone, che disse Con lo sviluppo degli schermi capacitivi, senza necessità di pennino e l'adozione di particolari scelte di hardware per Windows Phone 7, abbiamo dovuto interrompere la compatibilità delle applicazioni Windows Mobile 6.5.

Microsoft non essendo un produttore di smartphone distribuisce il proprio sistema operativo ad altri produttori di smartphone con cui instaura legami di partnership, che sono andati ad aumentare gradualmente con l'evoluzione di questo sistema.

- Nel 2010 con il lancio di Windows Phone 7, i partner furono HTC, Dell, Samsung e LG.
- Nel 2011 si aggiunsero Nokia, Acer, Fujitsu e ZTE.
- Nel 2012 con l'arrivo di Windows Phone 8, i partner furono Nokia, HTC, Samsung e ZTE, mentre venne a mancare un partner importante, LG.

Caratteristiche tecniche

Questo sistema operativo supporta l'architettura CPU ARMv7 ed utilizza DirectX per fornire accellerazione GPU all'interfaccia utente. Utilizza un kernel Windows NT e condivide con il sistema operativo Windows 8 caratteristiche quali driver, gestione della rete, sicurezza, componenti multimediali e grafiche. Altri aspetti in comune con Windows 8 sono l'utilizzo di tiles (mattonelle), che rappresentano le applicazioni installate, l'integrazione con i social network, l'aggiornamento del sistema tramite Windows Update, la possibilità di fare screenshots, l'integrazione di Skydrive, il look screen con sblocco verticale, ecc ... Supporta il multitasking e può operare su architetture di processori diversi, per chi utilizza sistemi embedded come ARM, x86, ecc ... Dal lato hardware supporta i processori multi-core fino a 64 core e le schede MicroSD.

Questo sistema operativo è incompatibile con le versioni precedenti di Windows Mobile, poiché si è deciso di riprogettare interamente il sistema

operativo Windows Mobile per smartphone prodotto da Microsoft. L'interfaccia utente è formata da una schermata contenente live tiles, che sono collegamenti ad applicazioni, funzioni o oggetti individuali, che gli utenti possono spostare, aggiungere e ridimensionare a proprio piacere. La loro caratteristica principale è l'aggiornamento in tempo reale, infatti per esempio il tile di un account mail mostrerà il numero di messaggi non letti.

Molte caratteristiche sono organizzate in hub, che combinano contenuti locali ed online mediante l'integrazione del sistema con i principali social network, per esempio l'hub Foto conterrà sia le foto fatte dalla fotocamera che quelle caricate su un social network. Ci sono diversi hub, che suddividono i dati presenti sul dispositivo, per citarne qualcuno, c'è l'hub Foto, Musica e Video, Contatti, Giochi, ecc ...

Windows Phone dispone di un sistema di riconoscimento vocale progettato da TellMe, che permette di effettuare ricerche nei vari hub dello smartphone. Nella versione Windows Phone 7, il dispositivo si appoggiava ad un software di nome Zune, per la sincronizzazione dei contenuti con gli altri dispositivi che montano lo stesso sistema. Questa applicazione è anche usata per scaricare applicazioni sullo smartphone e aggiornare il sistema operativo, ma dalla versione Windows Phone 8 non è stata più necessaria, poiché il collegamento dello smartphone ad un altro dispositivo viene visto come il collegamento di una chiavetta USB, tranne per l'eccezione di collegamento a computer Mac, che richiedono ancora un software per la gestione dei dati, di nome Windows Phone Connector.

Architettura

L'architettura di questo sistema operativo è strutturata come segue.

Come si vede in figura 1.7, l'architettura di questo sistema operativo è suddivisa in diversi livelli. Troviamo framework importanti per l'utilizzo delle applicazioni, che possiamo esaminare qui in seguito.

- **Silverlight** - è un framework per la creazione di applicazioni, che utilizza il linguaggio di markup XAML per lo sviluppo delle interfacce grafiche. Questo framework è un cross-browser con tecnologia cross-platform implementata dal .NET Framework. In questo framework risiedono API simili al DOM per i browser, che permettono di programmare le iterazioni con Javascript.

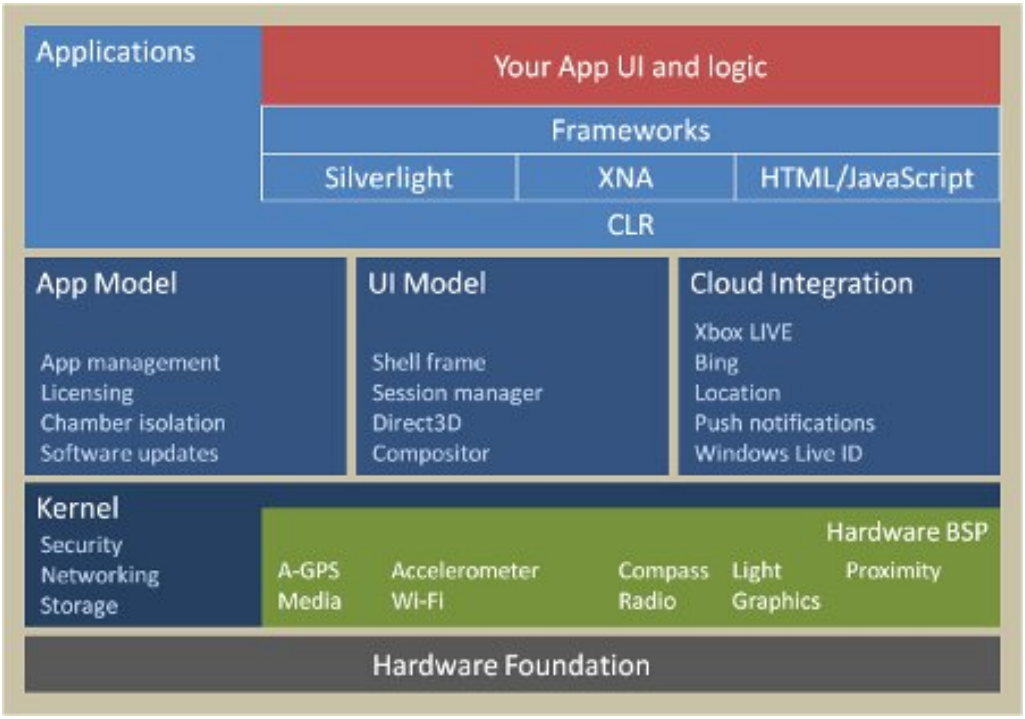

Figura 1.7: Architettura sistema operativo Windows Phone [3].

- **XNA** - è un framework pensato per la creazione di videogiochi, che fornisce funzionalità per semplificare lo sviluppo di videogiochi sia 2D che 3D. Questo framework è basato su DirectX 9 e .NET Framework.

- **HTML/JS** - framework contenente tecnologie che possono essere usate dalle applicazioni, nel caso specifico l'HTML e il Java Script.

Il componente CLR è contenuto nel .NET Framework ed è la Virtual Machine del sistema incaricata di compilare ed eseguire just-in-time le applicazioni presenti sul dispositivo. Si serve anche di un Garbage Collector, per alleggerire il carico di lavoro riguardante la gestione della memoria.

1.3.3.4 Blackberry

È un sistema operativo mobile proprietario sviluppato da Research in Motion, erede del sistema operativo EPOC. Il sistema operativo supporta dispositivi di input specializzati adottati da RIM, come la rotellina trackball e più recentemente il trackpad e touchscreen.

Caratteristiche tecniche

Questa piattaforma è nota per il suo supporto natio della posta elettronica aziendale tramite MIDP 1.0 e più recentemente con un sottoinsieme di MIDP 2.0, che tramite attivazione della Wireless permette la sincronizzazione con Microsoft Exchange, Lotus Domino, Novell GroupWise e per la sincronizzazione di e-mail, calendario, note, contatti, attività. Tutte le funzioni di questo sistema opertativo come il calendario, i contatti, le note e le attività sono ottimizzate da un software all'interno del sistema, chiamato BlackBerry Enterprise Server. Per quanto riguarda gli accessi internet e tutti i servizi legati all'utilizzo della rete, si può utilizzare un software alternativo al BlackBerry Enterprise Server chiamato BlackBerry Internet Service.

Questo sistema operativo è dotato di multitasking e supporta le tecnologie WAP 1.x e 2.0. Questo sistema utilizza un'architettura ARM e monta un processore Intel XScale ARM e utilizza registri a 16 x 32 bit, le istruzioni che adotta hanno un'architettura RISC. I dispositivi di ultima generazione BlackBerry utilizzano processori con firmware RedBoot, un bootstrap open source progettato per disporre di sistemi con architettura ARM. Le applicazioni su questo dispositivo possono essere sviluppate per utilizzare il compilatore Java Virtual Machine, oppure possono adottare un compilatore presente nel BlackBerry Enterprise Server chiamato MDS runtime. Entrambi i compilatori si occupano della gestione della memoria tramite un Garbage Collector. Questi dispositivi montano una piattaforma QNX che gli permette di poter utilizzare qualsiasi processore che sia dotato di un componente chiamato Memory Management Unit. Al momento alcuni dei processori con Memory Management Unit sono PowerPC, ARM, StrongARM, MIPS e SH-4.

Gli aggiornamenti del sistema operativo sono messi a disposizione dai vari operatori telefonici, che permettono di aggiornare il sistema operativo

senza l'utilizzo di una connessione ad un Pc.

Architettura

Questo sistema operativo viene eseguito su una piattaforma QNX mostrata in figura 1.8, che è progettata come un sistema microkernel composto da tanti piccoli componenti, che si occupano di uno specifico servizio, chiamati Server. Questi componenti sono adibiti a ricoprire anche le funzionalità che negli altri sistemi, vengono assolte dai driver.

Questa architettura porta significavi vantaggi, perché permette di avviare il sistema solo con le parti che si necessitano, senza dover per questo ricompilare l'intero sistema ogni volta che si intende modificare la lista dei componenti richiesti in esecuzione.

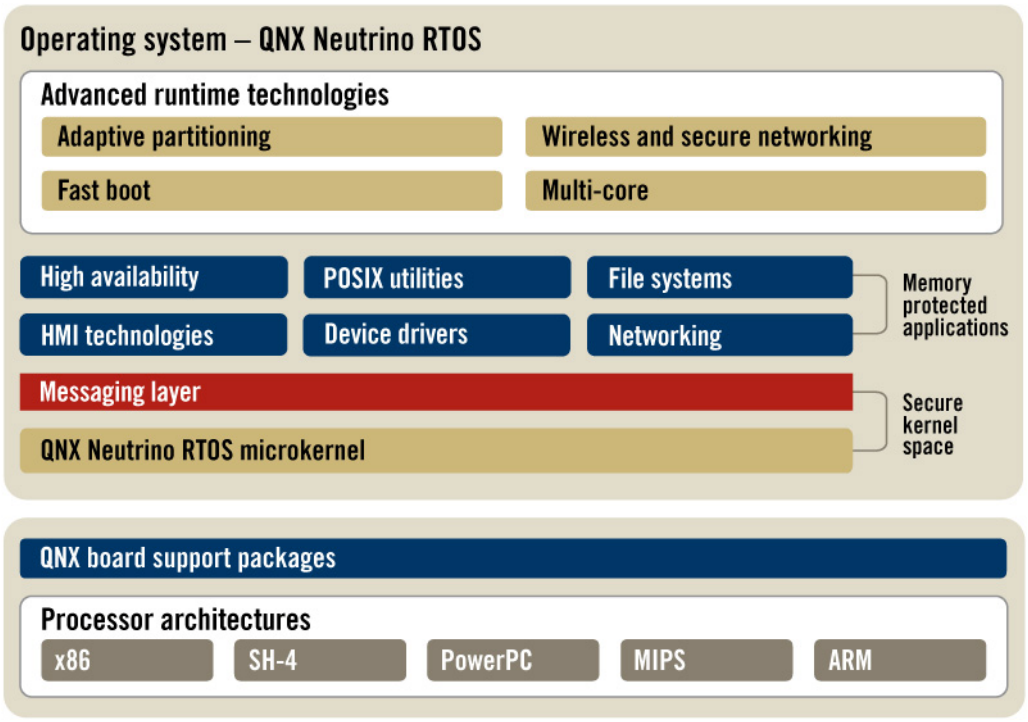

Figura 1.8: Sistema operativo QNX [4].

1.3.3.5 Symbian

È un sistema operativo open-source sviluppato della Symbian Foundation, attualmente supportato da Accenture, ma dopo l'abbandono da parte di Nokia, non è più supportato da nessun dispositivo mobile delle nuove generazioni.

La documentazione delle API è pubblica, perciò chiunque può sviluppare applicazioni per Symbian.

Caratteristiche tecniche

Questo sistema operativo supporta le funzionalità di multithreading e multitasking preemptive. Symbian tratta il multitasking seguendo l'approccio utilizzato da EPOC, che a sua volta è stato ispirato dal sistema operativo VMS sviluppato per computer e server. La gestione della memoria avviene tramite tecniche specifiche di Symbian, che sono molto efficienti nella gestione dello spazio su disco, poichè riducono sensibilmente i memory leaks, errori dovuti alla cattiva gestione della risorsa. Il sistema operativo è ottimizzato per i dispositivi che necessitano di un basso consumo e per i sistemi basati su ROM. Questo sistema operativo è basato su eventi, quindi la CPU viene automaticamente abilitata quando il dispositivo avverte la comparsa di eventi attivi, questo viene fatto tramite un linguaggio di programmazione chiamato ad oggetti attivi. Questa strategia porta ad un consumo minore di energia, al fine di assicurare una durata maggiore delle batterie.

Le applicazioni e il sistema operativo stesso vengono progettati seguendo un stile architetturale MVC. Per quanto riguarda l'interfaccia, non c'è una versione ufficiale e si trovano interfacce di varie tipologie, perché questo aspetto è stato affidato ai produttori dei dispositivi mobile, che volevano adottare Symbian come sistema operativo. Alcuni dispositivi che montano questo sistema, come ad esempio i Nokia N92 e N96 hanno la funzionalità di TV Mobile, tramite il supporto della tecnologia HSPDA. Alcuni dispositivi a partire dalla versione 9.1 fino a prima della 9.3 sono dotati di accelleratore grafico 3D.

Architettura

L'achitettura di questo sistema operativo è strutturata su 5 livelli, come mostrato in figura 1.9, che andremo ad analizzare partendo da quello più vicono all'utente, per poi prosegure avvicinandosi sempre più all'hardware del dispositivo.

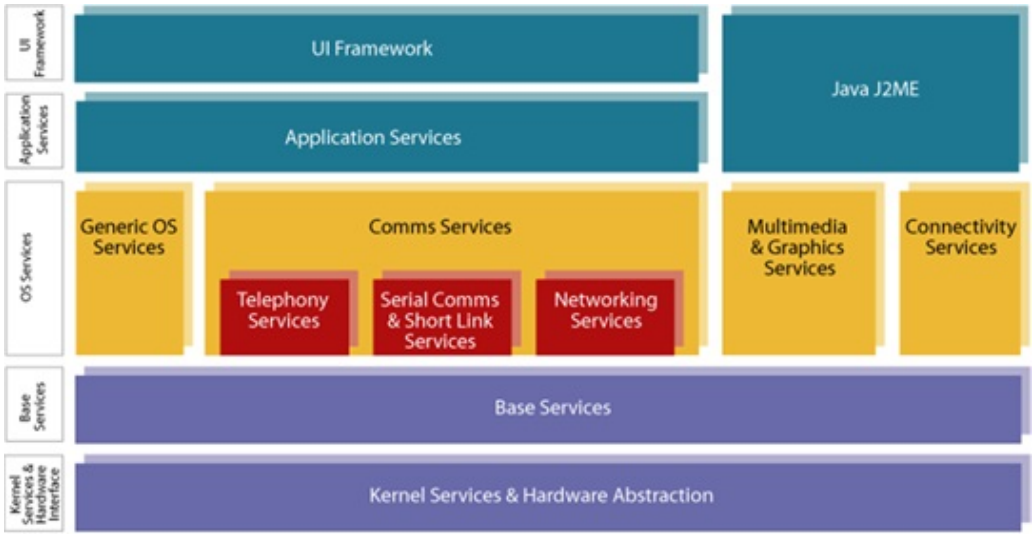

Figura 1.9: Architettura sistema operativo Symbian [5].

In questa architettura è interessante il ruolo ricoperto da Java ME. Siccome Java non si adatta perfettamente a questa specifica architettura a strati adottata da Symbian, l'implementazione di Java ME si basa sui seguenti aspetti.

- Una Virtual Machine con supporto specifico per ogni livello che integra il sistema Java seguendo il profilo MIDP 2.0 .
- Un set di librerie standard MIDP 2.0 .
- Implementazione del linguaggio CLDC 1.1
- Implementazione dei servizi Input/Output.
- Implementazione dei servizi di utilità.

- Un set di pluggins di basso livello che implementano interfacce per la comunicazione tra le librerie, il linguaggio CLDC ed il sistema nativo.

Il supporto Java fu incluso nel sistema operativo fin dalla prima versione, ma dalla versione 7 di Symbian, il sistema si è basato su Java ME e solo dalla versione 8 è stata introdotta la Java Virtual Machine.

UI Framework Layer - In questo livello risiedono i framework e le librerie per gestire le interfacce che verranno utilizzate per interagire con l'utente. I framework adibiti alla parte grafica delle interfacce sono:

- Animation Framework
- Front End Processor Framework
- Grid Framework

L'architettura delle interfacce del sistema sono gestite da Uikon Framework e il Control Enviroment Framework contiene una gerarchia di interface control. Questi due framework definiscono i comportamento base delle interfacce grafiche per l'utente. Il framework Uikon è stato creato per migliorare il precedente Eikon Framework, una libreria per le interfacce utente, questo per via della necessità di creare un facile supporto per lo sviluppo di queste interfacce.

The Application Service Layer - Questo livello supporta le interfacce utilizzate nelle applicazioni fornendo servizi utilizzabili nelle varie applicazioni, come servizi base di sistema, servizi più generici mirati alle applicazioni stesse, servizi di sincronizzazione dati. In questo livello sono inclusi anche un certo numero di motori applicativi chiave utilizzati da applicazioni come il calendario, l'agenda, la fotocamera, che possono essere usati ed estesti tramite l'utilizzo di opportune licenze. Per esempio alcune applicazioni per condividere i propri dati con le altre applicazioni si basano su un motore server-based.

Infine in questo livello si trovano servizi più generici associati a tecnologie presenti sul dispositivo come ad esempio i servizi di mail, messaggistica, browser, ecc ...

The OS Services Layer - Questo livello è il middleware del sistema Symbian ed ha al suo interno server, framework e liberie, che sono necessari per svolgere tutte le funzionalità offerte dal sistema operativo. I servizi offerti da questo livello possono essere catalogati in 4 macrogruppi.

- Servizi generici del sistema operativo.
- Servizi di comunicazione.
- Servizi per la grafica e il multimedia.
- Servizi di connettività.

Questo livello offre servizi e tecnologie specifiche che sono indipendenti dalle applicazioni del sistema operativo, ma che possono essere utilizzate da quest'ultime. In questo livello si trovano tutti i framework e le librerie utilizzati per fornire i servizi sopra citati, di particolare importanza è la C Standard Library, che offre supporto per il porting dei software.

The Base Services Layer - Questo livello è fondamentale per il sistema operativo, perché contiene i servizi di più basso livello, come il File Server e la User Library, che grazie al microkernel di Symbian vengono posizionati all'interno dello spazio dedicato all'utente. Questo è in contrasto con le architetture dei sistemi operativi monolitici, dove i File System Service e la User Library sono offerti come servizi prodotti dal Kernel.

In questo livello troviamo altri importanti framework come l'ECom Plug-in Framework, che si occupa di implementare gli standard di gestione delle interfacce usati in tutto il sistema e il DBMS framework. Questo livello contiene anche la Cryptography Library per i sistemi di protezione del sistema.

The Kernel Services and Hardware Interface Layer È il livello più vicino all'hardware del dispositivo ed include il Kernel del sistema operativo, i componenti che virtualizzano le risorse hardware tramite interfacce e i driver per tali componenti hardware. In questo livello il kernel utilizzato è EKA1, che dalla versione 8 del sistema operativo è stato aggiornato alla versione 2, un kernel real time a core singolo definito anche nanokernel. Anche se nella versione 8, l'EKA2 poteva essere scelto come opzione,

nelle versioni successive si è deciso di abbandonare lo sviluppo di questo nanokernel e continuare a sviluppare il kernel EKA1.

Capitolo 2

Applicazioni per Smartphone

In questo capito si andranno ad analizzare i vari aspetti legati allo sviluppo e alle tipologie di applicazioni utilizzate da dispositivi mobile, nel caso specifico, applicazioni per smartphone.

Per la stesura di questo capito sono state ricercate ed analizzate informazioni raccolte da produttori e sviluppatori di sistemi operativi per smartphone, oltre ad articoli riguardanti gli argomenti trattati nel capitolo [20] [19] [12] [15] [22] [16] [21] [17] [14].

2.1 Introduzione

Gli smartphone sono dei dispositivi mobile che fanno un ampio uso di applicazioni, al pari di computer che utilizzano software per offrire servizi e funzionalità all'utente. A differenza dei software che vengono installati su computer, che dispongono di risorse hardware molto più potenti e performanti rispetto ad uno smartphone, le applicazioni si ritrovano in un ambiente molto più limitato, con problematiche maggiori, perciò per loro natura necessitano di essere semplificate il più possibile, al fine di ottenere maggiore leggerezza e velocità di esecuzione.

Il problema principale dei dispositivi hardware è la gestione del consumo della batteria, questo implica alle applicazioni un ulteriore occhio di riguardo verso l'utilizzo delle risorse hardware che il dispositivo mette a loro disposizione, come la CPU per esempio. Per questo l'aspetto legato allo sviluppo di applicazioni per dispositivi mobile è in continua evoluzione e ogni sistema operativo per smartphone supporta lo sviluppo di applicazioni con

linguaggi, architetture e funzionalità improntate a migliorare continuamente le performance sui dispositivi che si servono di quel particolare sistema operativo.

Il mondo che si è creato attorno alle applicazioni si suddivide in due macro categorie di applicazioni per dispositivi mobile, le applicazioni native e le applicazioni indipendenti.

2.2 Applicazioni native

Le applicazioni native sono software sviluppati ad hoc per una determinata piattaforma e contengono una gran quantità di dati (immagini, testi), che una volta scaricati risiedono sul dispositivo mobile dell'utente. Le applicazioni native essendo sviluppate per un determinato sistema operativo possono sfruttare appieno le funzionalità hardware del dispositivo, come fotocamera, giroscopio, GPS, ecc ... Queste applicazioni posso essere ottenute in genere visitando specifici App Store, che offrono sia applicazioni gratuite che a pagamento, certificate dal produttore del sistema operativo che utilizzano, ma possono anche essere progettate da sviluppatori che decidono di utilizzare canali alternativi per la distribuzione delle loro applicazioni. Lo sviluppo di applicazioni native, varia a seconda del sistema operativo per cui si sta sviluppando l'applicazione nativa. Infatti le applicazioni native di ogni sistema operativo hanno diverse caratteristiche e sono compatibili solo sul sistema per cui sono state progettate.

2.2.1 Linguaggi di programmazione

Di seguito andremo ad analizzare i vari linguaggi di programmazione utilizzati per sviluppare le applicazioni native per i loro sistemi operativi di riferimento.

2.2.1.1 Android

Il linguaggio di programmazione per le applicazioni Android è Java, ma a differenza di un applicazione Java, queste applicazioni non dispongono di un main, perché sono pensate per essere dei componenti indipendenti che rispondono agli eventi generati sul dispositivo, tutto ciò per ottimizzare

l'utilizzo delle risorse a disposizione, che rispetto ad un computer sono molto limitate.

C'è da dire che si può utilizzare anche la libreria nativa C associata alle JNI, per sviluppare applicazioni in linguaggio C nativo, rinunciando alla Dalvik Virtual Machine ed è stato riscontrato che per le applicazioni che necessitano di molti accessi alla memoria e di effettuare complessi calcoli, questa scelta di sviluppo porta ad un miglioramento prestazionale molto alto. Questa ricerca è stata presentata nell'International Conference on Control [24].

2.2.1.2 iOS

Il linguaggio di programmazione è l'Objective-C ed è un linguaggio di programmazione dinamico, che rappresenta un'estensione del linguaggio C con la particolarità di essere orientato all'utilizzo di oggetti. Una caratteristica di questo linguaggio di programmazione è che la compilazione e il linking viene effettuato il più possibile a runtime. Questo linguaggio garantisce una completa compatibilità con il C.

2.2.1.3 Windows Phone

Per sviluppare applicazioni native per Windows Phone possiamo utilizzare C++ e C#.

2.2.1.4 BlackBerry

Lo sviluppo delle applicazioni native per BlackBerry può essere effettuato sfruttando diversi linguaggi di programmazione. In base a quale linguaggio si vuole sceglie dovranno essere scaricati opportuni tools per consentire lo sviluppo dell'applicazione nel linguaggio desiderato.

Il linguaggio di programmazione per Blackberry è Java, ma dalla comparsa della versione 10 di questo sistema, si possono utilizzare linguaggi di programmazione come Java, C/C++/Qt, ActionScript.

2.2.1.5 Symbian

I linguaggi di programazione per le applicazioni per sistemi operativi Symbian sono il C++ e Qt.

2.2.2 Tools di sviluppo

Per lo sviluppo di applicazioni native vengono offerti vari tools che aiutano gli sviluppatori, nel processo di sviluppo e di test delle applicazioni.

2.2.2.1 Android

Per inziare a sviluppare applicazioni native per Android bisogna dotarsi di vari strumenti di sviluppo e di simulazione. Come prima cosa bisogna scaricare l'Android SDK, che si può trovare sul sito ufficiale di Android. L'SDK andrà installato come plug-in su piattaforme di sviluppo, come Eclipse o Netbean. Questo plug-in conterrà tutte le librerie necessarie allo sviluppo di applicazioni native per Android.

Per testare il funzionamento dell'applicazione, non si dovrà necessariamente installarla sul dispositivo mobile, infatti esistono plug-in chiamati AVD (Android Virtual Device), che virtualizzano i dispositivi mobile in commercio e ne simulano il comportamento. Se per esempio utiliziamo Eclipse come piattaforma di sviluppo, il plug-in per la simulazione dei dispositivi mobile smartphone che montano Android come sistema operativo è l'ADT.

L'SDK Android permette una volta completata l'applicazione di produrre il relativo file .apk, che è un file autoinstallante utile per la distribuzione dell'applicazione.

2.2.2.2 iOS

L'ambiente di sviluppo è caratterizzato da un SDK che dispone di una documentazione di ottimo livello. Và comunque precisato che per sviluppare queste applicazioni occorre avere un Mac, perché iOS è un sistema operativo per smartphone proprietario con compatibilità esclusiva con gli altri sistemi e dispositivi Apple, che si interfacciano solo tra di loro, non c'è da stupirsi poiché da sempre Apple ha sempre avuto una tendenza a far prodotti a livello proprietario, al fine di affermare il proprio standard.

Il software utilizzato per la produzione di applicazioni è l'X-Code, che adotta interfacce molto intuitive ed offre una serie di funzionalità, che permetto di sviluppare più rapidamente l'applicazione che si vuole creare. Dentro l'ambiente di sviluppo sono presenti strumenti molto utili, come l'Interface Biulder che permette agevolmente di sviluppare la parte riguardante le

interfacce grafiche, anche se impone delle limitazioni importanti, infatti gli oggetti dovranno essere prima creati e poi collegati a questo strumento, altra limitazione è che il codice dei vari oggetti grafici, non viene generato automaticamente, ma è tutto a carico dello sviluppatore. Un altro strumento importante è il simulatore, che ci permetterà di testare le nostre applicazioni in locale, senza perdere tempo a caricarle sul dispositivo, senza rischio di danneggiare quest'ultimo. Come si può ben pensare, il simulatore non riesce ad emulare perfettamente il dispositivo sul quale andrà in funzione la nostra App, perciò per le applicazioni si dovranno sempre effettuare anche test sul dispositivo fisico, ma almeno in fase di sviluppo si potranno avere i primi riscontri. Inoltre il simulatore non copre tutte le risorse presenti sul dispositivo fisico, infatti non troveremo risorse come il GPS o le notifiche Push.

2.2.2.3 Windows Phone

La piattaforma per lo sviluppo di applicazioni native Windows Phone è Visual Studio Express, che può essere installato gratuitamente sul proprio PC.

Per lo sviluppo di applicazioni native per Windows Phone, si possono anche utilizzare framework importanti come Sirverlight e XNA, che incorporano importanti funzionalità per lo sviluppo delle applicazioni. Generalmente si consiglia un utilizzo di Sirverlight per lo sviluppo di applicazioni di utilità, XNA è un framework utilizzato per la creazione di giochi. Dalla versione Windows Phone 7.1 'e possibile unire il codice dei due framework Sirverlite e XNA per la produzione dell'applicazione. Per la creazione di interfacce si utlizza il linguaggio di programmazione XAML.

Una volta installata la piattaforma di sviluppo occorre dotarsi di un opportuno SDK, che offre vari strumenti per lo sviluppo di applicazioni. Per facilitare lo sviluppo delle applicazioni vengono offerti anche una serie di tool aggiuntivi, che possono sempre essere scaricati gratuitamente. Uno di questi tool è Microsoft Expression Studio che offre vari strumenti.

- Microsoft Expression Blend 4, per la progettazione di interfacce.
- Microsoft Expression Design 4, per la progettazione della parte grafica.
- Microsoft Expression Encoder 4, per la codifica di file video e audio, la loro riproduzione e trasmissione online.

- Microsoft Expression Web 4, per la progettazione e pubblicazione di siti Web

Per la simulazione dei dispositivi smartphone che montano Windows Phone come sistema operativo, ci viene in contro il tool Windows Phone Emulator. Il tool Microsoft Zune permette di sincronizzare i contenuti multimediali con dispositivi Zune e anche con smartphone Windows Phone. Application Deploiment è un tool che permette di trasferire le applicazioni create al simulatore o al dispositivo fisico.

2.2.2.4 BlackBerry

Per lo sviluppo di applicazioni Blackberry, si possono utilizzare piattaforme di programmazione come Eclipse, Visual Studio o NetBeans, integrandole con opportuni plug-in che possono essere scaricati gratuitamente, come il BlackBerry Java SDK se si vuole utilizzare Java come linguaggio di programmazione, oppure il BlackBerry Native SDK per la programmazione in C/C++.

Se si decide di utilizzare il Java SDK, al suo interno troveremo importanti tools per lo sviluppo delle applicazioni come:

- RAPC, per la compilazione di file con estensione .java e .jar . Produce in uscita file con estensione .cod .
- JavaLoader, per aggiornare e aggiungere le applicazioni sul dispositivo mobile.
- BlackBerry Signature Tool, per inviare richieste di codici di firma necessari per l'utilizzo di alcune API.
- Preverify Tool, per verificere le classi dell'applicazione.
- JDWP, per eseguire debug.

BlackBerry da la possibilità di utilizzare una piattaforma di programmazione alternativa sviluppata ad hoc per gli sviluppatori di applicazioni per questo sistema operativo. La piattaforma in questione è BlackBerry Enterprise Service/BlackBerry Device Service (in base alla versione BlackBerry per la quale si sviluppa) e offre tools per sfruttare le varie tecnologie utilizzabili su questo sistema operativo.

- Cascades, per la programmazione in C/C++.
- BlackBerry Java, per la programmazione in Java.
- WebWorks, per utilizzare l'HTML5.
- Adobe AIR, per programmare in ActionScript.

Tra i tools utilizzabili troviamo anche BlackBerry Smartphone Simulator, che permette di simulare dispositivi smartphone e testare le applicazioni create prima di caricarle sul dispositivo fisico.

2.2.2.5 Symbian

Per sviluppare applicazioni Symbian possiamo utilizzare due piattaforme, Carbide.c++ e Qt Creator. Carbide.c++ offre tutti gli strumenti necessari per sviluppare applicazioni Symbian, inoltre si possono estenendere le sue potenzialità installando Symbian SDKs, che dà pieno accesso a tutte le API Symbian e incorpora tools per il test di applicazioni e la simulazione dei dispositivi smartphone che montano il sistema operativo Symbian. La piattaforma Qt Creator è utilizzata per lo sviluppo di applicazioni con interfacce grafiche per Symbian, all'interno troveremo importanti tools già integrati nella piattaforma.

- Qt Designer, per la creazione di interfacce grafiche.
- qmake, per creare applicazioni compatibili con diversi sistemi operativi.
- Qt Linguist, permette di incorporare informazioni generali come autori, sviluppatori, ecc ...
- Qt Assistant, permette di visualizzare tutta la documentazione.

2.2.3 Componenti chiave

Nello sviluppo di applicazioni native è da considerare che ogni sistema adotta dei componenti chiave, che ricoprono dei ruoli specifici all'interno dell'applicazione. Questo porta al risultato che le applicazioni native dei vari sistemi vengono instradate verso un certo stile di programmazione, poiché sono obbligate ad utilizzare certi componenti necessari per la loro esecuzione.

2.2.3.1 Android

Un'applicazione Android si compone di vari elementi per il suo corretto funzionamento, alcuni necessari, altri svolgono specifiche funzioni che in base all'applicazione che si vuole sviluppare possono essere incorportati o meno. Android mette a disposizione una certa gamma di componenti che si possono utilizzare per lo svilluppo di un'applicazione nativa.

View

Questo componente fornisce le aree rettangolari adibite al disegno degli elementi grafici e della cattura e gestione degli eventi generati dall'interazione dell'utente con il componente visualizzato sul dispositivo, come ad esempio bottoni, caselle di testo, tabelle, etichette.

Le due classi principali per la gestione di questa risorsa sono android.view.View e android.view.ViewGroup. Nasce qui il concetto di layout, per la gestione dell'impaginazione dei controlli all'interno di un contenitore di View. Utilizzare i layout facilita la progettazione di interfacce grafiche semplificando il posizionamento dei controlli inseriti nell'interfaccia.

Activity

Questo componente ricopre un ruolo fondamentale all'interno di ogni applicazione, poiché è quell'entità che si occupa del salvataggio dello stato di un'applicazione e del successivo ripristino. Ogni applicazione può avere più Activity associate, per le diverse fasi del proprio ciclo di vita, ma và comunque ricordato che può essere attiva una sola Activity alla volta, che sarà detta essere in foreground, primo piano, tutte le altre saranno invece nello stato di background, sullo sfondo, in attesa di essere nuovamente riutilizzate. Tutte le Activity in background possono essere terminate dal sistema operativo, per esempio se la memoria diventa insufficiente o anche per altre ragioni, perciò è chiaro che ogni applicazione Android può cambiare stato in ogni momento e può essere chiusa o interrotta in ogni istante. Per stabilire in caso di necessità, quale Activity terminare prima per liberare memoria, ad ogni Activity viene assegnata un certo livello di priorità. Per gestire al meglio i cambi di stato, ogni Activity dispone di un buon numero

di callback che gli permetto di rispondere ad eventi come la sospensione, la ripresa, l'interruzione.

Generalmente un'Activity è un'azione specifica che l'utente può fare, la getione dell'interazione tra le varie Activity e la loro esecuzione è un aspetto fondamentale per la creazione di un'App. Ogni Activity ha un ciclo di vita ben definito, come si può vedere in figura 2.1.

Service

Questo componente rappresenta un processo che viene eseguito in background, senza alcuna interfaccia, che svolge attività nascoste agli occhi dell'utente, solitamente senza dovervi interagire e di fatti ha un comportamento molto simile ai processi deamon per Linux.

Un service ha una priorità più alta rispetto ad un Activity in stato di inattività, questo per far terminare dal sistema prima le Activity inattive, in caso di necessità come l'esaurimento della memoria, ma nel caso in cui venga ugualmente terminato un Service, appena le risorse torneranno disponibili, verrà automaticamente riavviato.

Service e Activity sono risorse molto simili tra loro, l'unica differenza risiede nell'interazione che si ha con l'utente, perciò in fase di sviluppo tutto quello che avrà bisogno di interfacce e di un'iterazione con l'utente verrà espresso con Activity, il resto delle operazioni verrà fatto tramite Service. Avendo un comportamento da background, un Service è un componente meno complesso rispetto un Activity e ha un differente ciclo di vita, come si può vedere in figura 2.2.

Intent

Questo componente è fondamentale, perché descrive azioni specifiche e offre funzionalità molto utili durante lo sviluppo di un'applicazione. Ogni Intent è in pratica una funzione ben definita che viene richiamata da qualunque applicazione necessiti quel servizio, perciò l'insieme di tutti gli Intent può essere visto come una libreria di comandi del nostro sistema operativo. Possiamo avere Intent per inviare sms, inviare e-mail, chiamare casa, ecc ... Di fatto un Intent è un'operazione che viene ritenuta dagli sviluppatori, di carattere generale, quindi facilmente riutilizzabile in diverse applicazioni, senza dover stare a riscrivere ogni volta il codice.

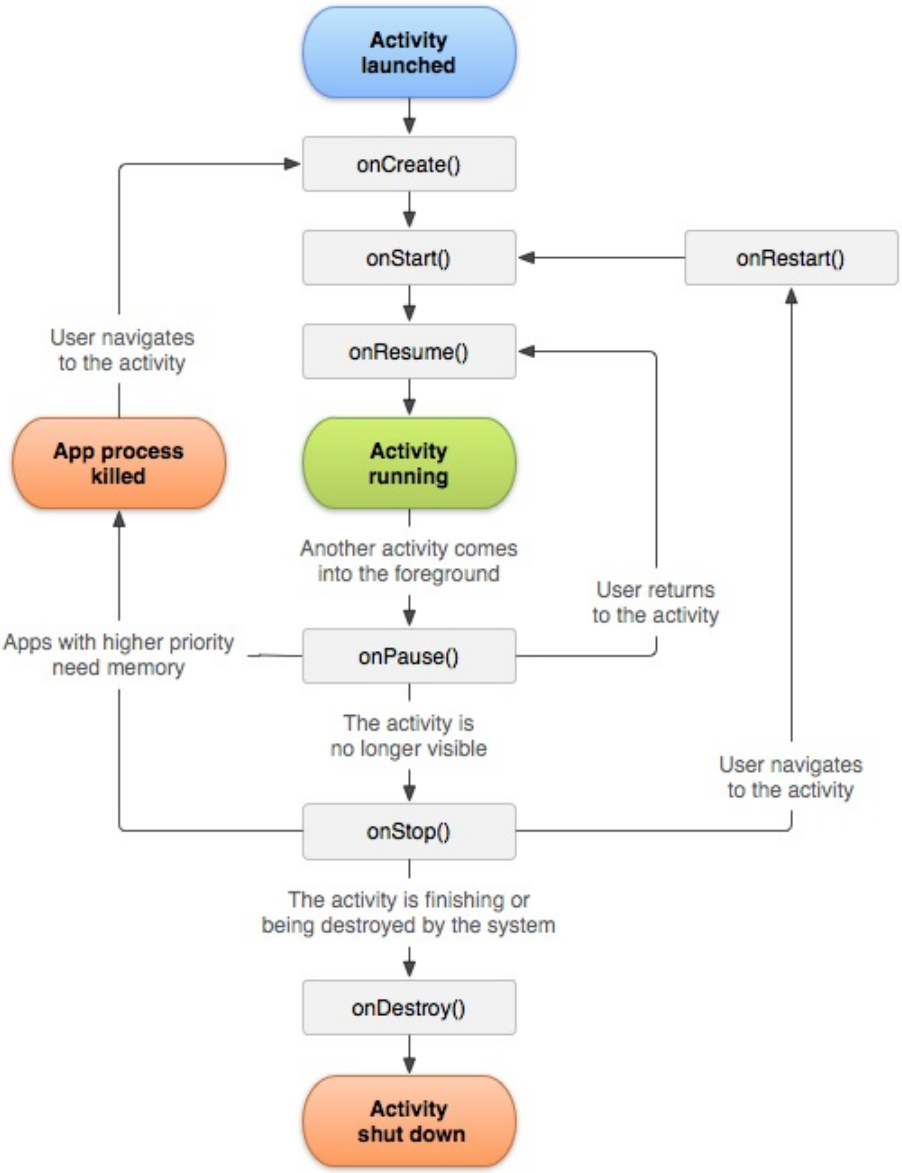

Figura 2.1: Ciclo di vita di un Activity [6].

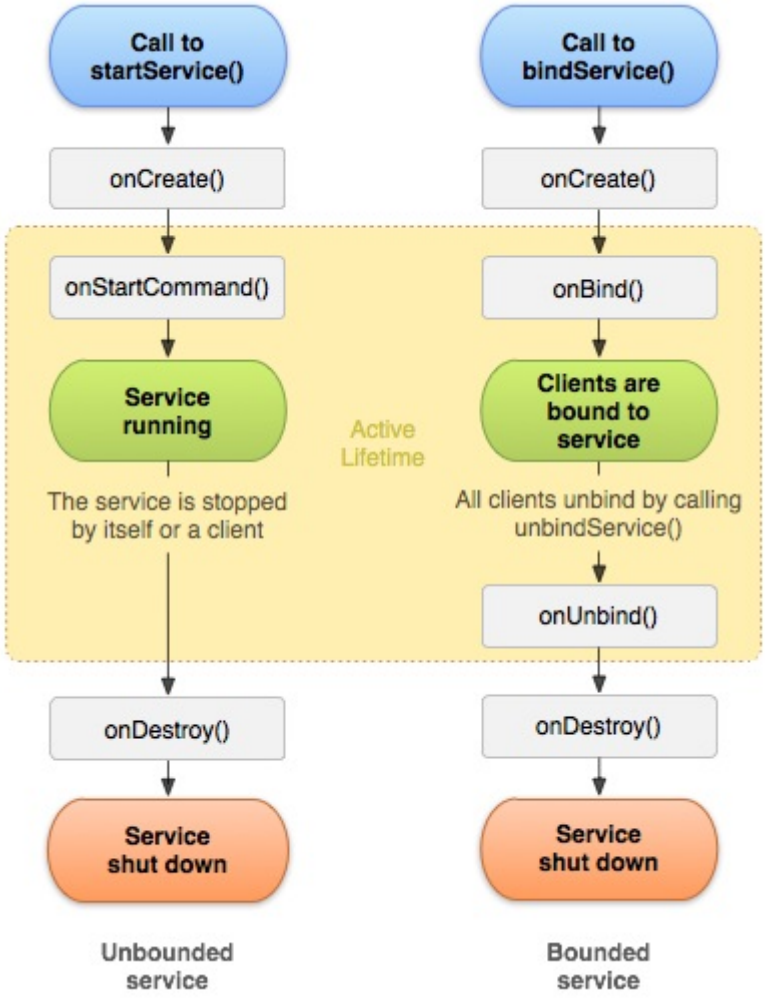

Figura 2.2: Ciclo di vita di un Service [6].

Riassumendo un Intent può essere utilizzato per:

- Trasmettere informazioni al verificarsi di eventi.
- Aiutare Activity e Service al completamento di un compito.
- Lanciare in esecuzione Activity o Service.
- Supportare le iterazioni fra le varie applicazioni presenti sul dispositivo.

L'implementazione di questa risorsa trasforma il dispositivo Android in un insieme di componenti indipendenti, ma facenti parte di un sistema interconnesso.

Content Provider

Questo componente rappresenta un contenitore di dati, che condivide informazioni globali per le varie applicazioni presenti sul dispositivo. Poiché Android è un sistema Linux-based, ogni applicazione ha uno spazio dedicato in memoria protetto, inaccessibile dalle altre applicazioni, perciò è necessario avere un tramite per lo scambio di dati tra le varie applicazioni, questo è il lavoro del Content Provider.

Esistono diversi Content Provider nativi sul sistema, con diverse caratteristiche, per venir in contro alle varie esigenze che si hanno, alcuni permettono la sola lettura dei dati, altri permettono di aggiornare, di modificare e di cancellare i record, come ad esempio per l'applicazione che gestisce i contatti. Un database SQLite è un esempio di content provider che possiamo utilizzare sul nostro dispositivo Android, comunque Android fornisce una serie di Content Provider nativi, presenti nel packpage android.provider dell'SDK.

Resource

Questo componente rappresenta tutte quelle risorse utilizzate dall'applicazione che non siano linguaggio di programmazione, come immagini, suoni, video. Questi componenti vengono compilati all'interno dell'applicazione e saranno presenti nella cartella res della nostra App. Il compilatore dopo

aver compilato ed impacchettato tutte le risorse genera una classe R, dove all'interno sono presenti tutti gli identificatori che possono essere adoperati per richiamere le risorse in punti specifici dell'applicazione. Grazie a questa tecnica, si ha un risparmio di spazio, rispetto al Java standard, poiché non si dovranno memorizzare le stringhe di referenza.

Broadcast Reciver

Questi componenti rimangono in ascolto per ricevere i messaggi e le notifiche provenienti dal sistema o da altre applicazioni, al fine di gestire la risposta a tali eventi. Possono gestire altri componenti applicando modifiche al verificarsi di eventi specifici, ma anche mandare in esecuzioni servizi specifici quando un certo evento si verifica. Tipici messaggi che posso ricevere sono rappresentati dagli Intents Broadcast.

Android Manifest

Questo componente è di importanza chiave in ogni applicazione, all'interno troviamo tutte le informazioni riguardanti le altre risorse utilizzate dalla nostra applicazione, in pratica definisce i contenuti, i comportamenti e i permessi dell'applicazione. La parte riguardante i permessi è molto importante, poiché durante l'installazione dell'applicazione sul dispositivo, il file AndroidManifest.xml presente all'interno dell'applicazione verrà consultato dal dispositivo, per erogare i vari permessi e per utilizzare le risorse che sarebbero altrimenti non accessibili.

Tra i principali permessi troviamo:

- READ_CONTACTS che permette di leggere i dati dei contatti dell'utente.
- WRITE_CONTACTS che permette di modificare i dati dei contatti dell'utente.
- RECIVE_SMS che permette di visualizzare i messaggi ricevuti.
- INTERNET che permette di utilizzare la connessione internet.
- ACCESS_FINE_LOCATION che permette di utilizzare il GPS.

2.2.3.2 iOS

Per creare applicazioni iOS ci si ispira al MVC (model-view-controller) design pattern, che rappresenta il design pattern più utilizzato dai programmatori di applicazioni per sistemi operativi iOS. Questo design pattern è strutturato come mostrato in figura 2.3.

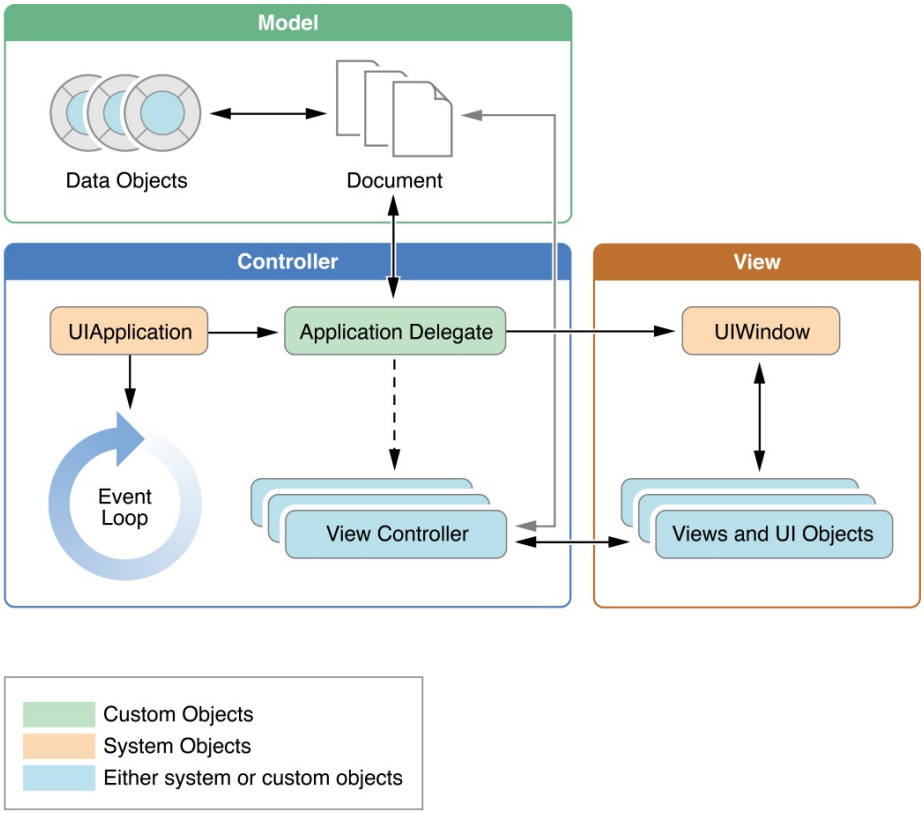

Figura 2.3: iOS Model-View-Controller [7].

Ogni oggetto della nostra applicazione ricoprirà un ruolo ben definito dal pattern, secondo il MVC sono disponibili 3 ruoli:

- **Modello** - un oggetto con questo ruolo contiente dati specifici dell'applicazione e definisce le varie procedure per la manipolazione di quest'ultimi.

- **Vista** - un oggetto con questo ruolo ha il compito di far visualizzare all'utente i dati di un certo modello e di far interagire l'utente con questi dati. Perciò gli oggetti con questo ruolo, si occuperanno anche dell'interfaccia dell'applicazione.

- **Controllore** - un oggetto con questo ruolo ha il compito di gestire le iterazioni tra gli oggetti vista e modello, inclusa la loro inizializzazione. Un controllore andrà ad eseguire le varie modifiche richieste da una vista su di un modello e si occuperà di mostrare i dati di un modello ad una vista, in pratica è il controllore ad eseguire e gestire tutte le operazioni tra gli oggetti degli altri ruoli.

Ogni applicazione iOS quando viene mandata in esecuzione segue una certa logica che descrive il suo ciclo vitale, come mostrato in figura 2.4.

Ora andiamo ad analizzare i componenti che potranno essere utilizzati ed estesi dallo sviluppatore, per comporre l'applicazione nativa per iOS.

View

Questo componente individua un'area rettangolare sullo schermo del dispositivo, al fine di ospitare al suo interno elementi grafici o anche altre view tramite un sistema d'incapsulamento creando strutture di tipo padre-figlio dove il padre viene chiamato superview e i figli subviewes. Il compito di questo componente è quello di mostrare all'utente sullo schermo, tutti i componenti grafici e gestire l'interazione dell'utente con i vari componenti.

View Controller

È un componente fondamentale per ogni applicazione iOS, in quanto gestisce l'iterazione tra le View e il modello dati. Inizialmente questo componente nasce con già una View come attributo, ma non c'è limite alle View che gli si possono aggiungere, infatti possono essere così implementate gerarchie di View.

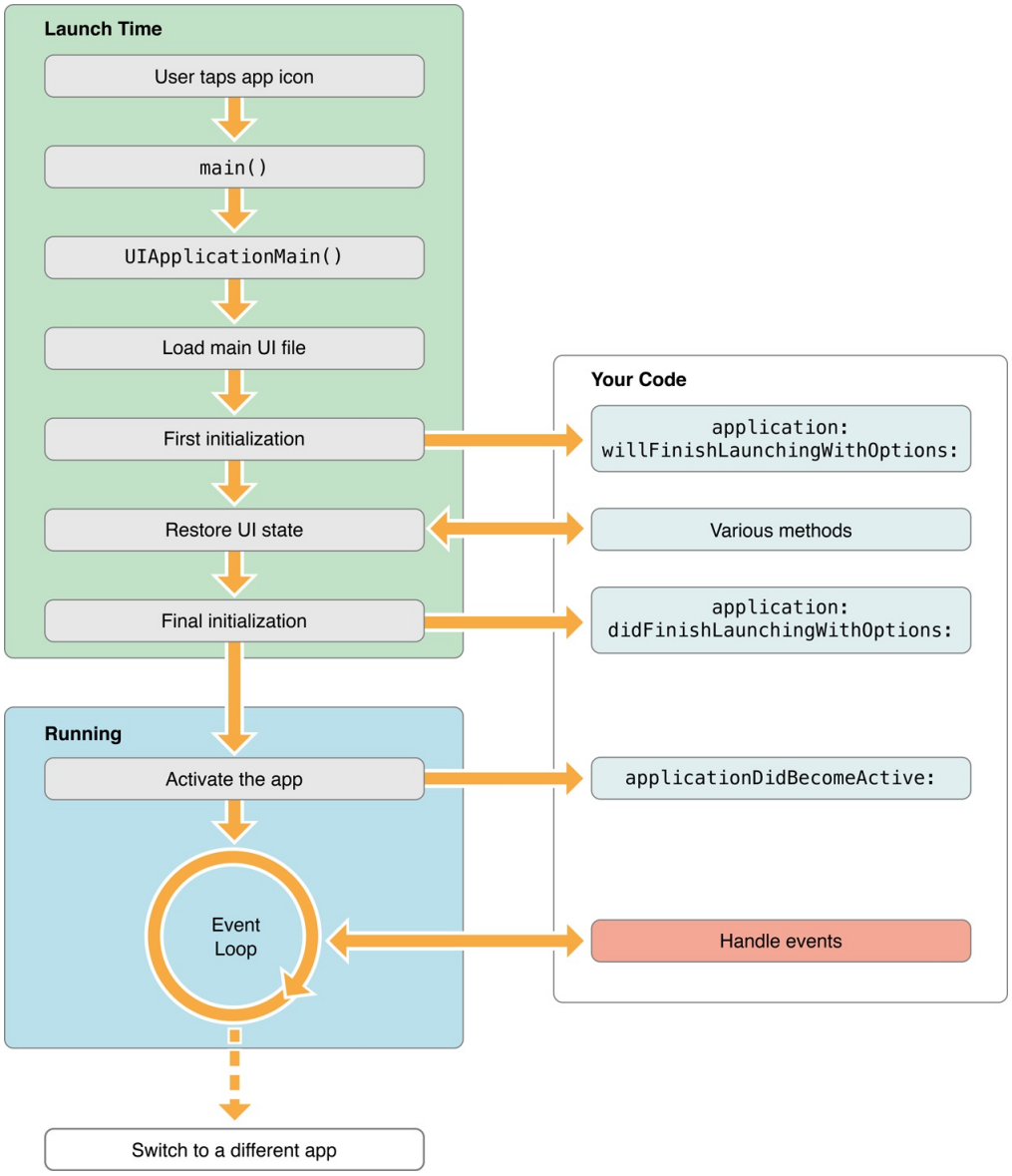

Figura 2.4: Ciclo di vita di un'applicazione iOS [7].

Application Degate

Questo componente svolge un ruolo molto importante, perché si occupa di gestire i cambi di stato delle applicazioni, in sostanza un cambio di stato è ad esempio il passaggio da foreground a background dell'applicazione. In alcuni casi questi eventi sono gesti della UIApplication, ma nel caso non sia possibile la sua gestione, l'evento viene gestito da questo componente.

Componenti di Sistema

Alcuni dei componenti che verranno a trovarsi nell'applicazione che si intende sviluppare sono forniti già dal sistema e non si possono modificare. Questi componenti vengono istanziati al momento della compilazione del codice sorgente dell'applicazione e verrà riscontrato il loro contributo solo nell'eseguibile dell'applicazione.

- **UIApplication** - questa classe è adibita al controllo e coordinameto delle applicazioni presenti sul sistema iOS. Quando un'applicazione viene mandata in esecuzione, viene creato automaticamente un oggetto singleton di classe UIApplication, che svolge il ruolo di coordinatore e gestore degli eventi che vengono generati.

  Per gestire l'instradamento degli eventi si avvale di altri due componenti, UIControl e UIWindow. Di fatto questo componente rappresenta l'applicazione.

- **UIControl** - questa classe è adibita al controllo di tutti gli elementi che compongolo le applicazioni, come ad esempio pulsanti e cursori. È una classe che definisce la struttura e i comportamenti base delle interfacce. Il ruolo principale di un oggetto appartenente a questa classe è quello di definire l'interfaccia e la sua implementazione base per la preparazione di richieste d'intervento in previsione di eventi futuri.

- **UIWindow** - questa classe gestisce e coordina le view e la loro visualizzazione sul display del dispositivo. È una classe che definisce un oggetto chiamato Windows, responsabile di visualizzare le view sullo schermo e di inviare gli eventi generati alle rispettive view.

2.2.4 Distribuzione

Per quanto riguarda l'aspetto della distribuzione delle applicazioni native agli utenti di un determinato sistema operativo, la loro distribuzione può essere effettuata sia tramite canali ufficiali, sia privatamente scegliendo lo strumento che viene ritenuto più adatto ad offrire questo servizio, in questo caso si parla di applicazioni di terze parti.

Per quanto riguarda il confronto tra i vari sistemi operativi è di più alto interesse l'aspetto legato alla distribuzione tramite canali ufficiali.

2.2.4.1 Android

Per arrivare a distribuire un'applicazione per via ufficiale, si consiglia di seguere una certa procedura che andremo a descrivere qui in seguito.

Release Process

Dopo aver sviluppato e testato il corretto funzionamento di tutti gli aspetti e delle funzionalità dell'applicazione, si prosegue preparando la release Candidate Build, una versione funzionante della nostra applicazione. A questo punto occorrerà apportare qualche modifica al file AndroidManifest.xml, alcune di queste modifiche saranno richieste specificatamente dall'Android Market, altre possono essere effettuate dallo sviluppatore, per questioni di buona programmazione. Per esempio, alcuni aspetti importanti da controllare nel file AndroidManifest.xml sono la correttezza del nome dell'applicazione, la versione dell'applicazione, l'identificativo dell'applicazione utilizzato per gli aggiornamenti della stessa, la corretta impostazione l'uses-sdk dell'applicazione, bisogna disabilitare l'opzione debuggable e verificare che tutti i permessi impostati nel file AndroidManifest.xml siano corretti. Dopo aver fatto tutti questi controlli e per sicurezza ritestato tutti i vari funzionamenti dell'applicazione, si può passare alla fase successiva.

Generate Package

In questa fase generiamo il package per Android, il sopracitato file .apk, che andrà ufficializzato tramite una firma digitale. Questa procedura può facilmente essere eseguita poiché l'SDK contiene anche gli strumenti per eseguire questo procedimento tramite l'utilizzo di opportuni wizard. In questa

procedura ci saranno delle impostazioni da inizializzare, si possono trovare sul sito ufficiale tutte le informazioni per completare correttamente la procedura.

Publishing

Una volta ottenuto il file .apk possiamo procedere alla pubblicazione ufficiale della nostra applicazione sull'Android Market. Questa fase consiste tramite un account Google, di fornire i propri dati nel form dedicato allo sviluppatore di applicazioni Android, che consiste nell'inserimento di tutti i dati richiesti, si verifica di essere in possesso della licenza per lo sviluppo di applicazioni Android, che è conosciuta come Android Market Developer Distribution Agreement.

Dopo questa piccola serie di passaggi burrocratici viene permesso di caricare la propria applicazione, con varie modalità e opzioni che si possono sceglie e una volta completato con successo anche questo passaggio, l'applicazione verrà posizionata dentro l'Android Market e pronta per essere scaricata su qualsiasi dispositivo Android che la richieda.

2.2.4.2 iOS

Per iniziare a sviluppare applicazioni per iOS occorre come prima cosa essere iscritti al developer program e scaricare Xcode.

Una volta terminato lo sviluppo dell'applicazione ed effettuati i vari test sul simulatore per verificare che l'applicazione funzioni correttamente, si può passare alla distribuzione dell'applicazione nel market online ufficiale, l'AppStore.

Per far si che la pubblicazione sullo store dell'applcazione vada a buon fine è bene segure questa procedura.

Invio dell'applicazione ai Betatester

Prima di procedere all'invio dell'applicazione ai betatester, si cosiglia di leggere le regole che Apple ha definito per lo sviluppo delle sue applicazioni, come ad esempio:

- Vietato inserire loghi Apple.

- Vietato utilizzare l'UDID.
- Vietato il salvataggio di dati personali senza autorizzazione.

Tutte le regole da seguire per far si che un'applicazione segua le politiche di comportamente della Apple, si trovano all'interno del HIG: Human Interface Guidelines, che si può consultare sul sito ufficiale dell'Apple.

Prima di inviare l'applicazione ai betatester occorrere richiedere gli UDID specifici, relativi ai dispositivi sui quali verrà testata l'applicazione. In più bisogna creare un certificato ad hoc relativo al Distribution Provisioning Profile, che dovrà essere inserito in XCode per poi ricompilare l'applicazione con l'ausilio del certificato ed ottenere un file con estensione .ipa . Una volta ottenuto il file .ipa della nostra applicazione potremmo inviare tale file ai betatester per l'approvazione.

Pubblicazione sull'AppStore

Una volta ricevuto il file .ipa, i betatester inseriranno questo file sul portale iTunes Connect, in modo da installare l'applicazione sui loro dispositivi, ma prima di poter esaminare questa applicazione occorrerà inviare i file binari dell'applicazione tramite iTunes Connect, infatti fino a quando non verranno inviati i file binari, lo stato della revisione verrà segnalato come Waiting For Upload. Dopo questo passaggio potrà essere monitorato lo stato di revisione dell'applicazione sempre da iTunes Connect, che potra variare in 4 stati:

1. Upload Received.
2. Waiting For Review.
3. In Review.
4. Ready For Sale (applicazione approvata).

Una vota arrivati al quarto stato, l'applicazione verrà automaticamente caricata sull'AppStore e sarà disponibile per tutti gli utenti di iOS.

2.2.4.3 Windows Phone

Dopo aver sviluppato e testato il corretto funzionamento di un'applicazione per Windows Phone, si può distribuire l'applicazione tramite i canali ufficiali Windows Phone Store e Windows Store. Per prima cosa occorre registrare un account da sviluppatore per ricevere la licenza allo sviluppo di applicazioni Windows Phone. Questo account può essere di due tipologie, uno dedicato agli sviluppatori privati e l'altro per le aziende. Se si vuole creare applicazioni che sfruttino strumenti pubblicitari per generare introiti, come per esempio pop-up, allora occorre registrare un account Microsoft Advertising pubCenter. Prima di inviare l'applicazione a Windows Phone Dev Center per l'approvazione è consigliato consultare le linee guida presenti nel documento App certification requirements for Windows Phone.

Con l'invio dell'applicazione è richiesta anche la spedizione di un packpage XAP, che viene creato automaticamente durante la compilazione dell'applicazione e contenente tutte le informazioni tecniche riguardanti l'applicazione creata. Il packpage XAP è richiesto ogni volta che si andrà a modificare, ad aggiungere e ad aggiornare l'applicazione che si vuole distribuire sugli store. Quando si inviano questi file occorrerà anche fornire una serie di informazioni, riguardanti la distribuzione dell'applicazione.

Una volta che l'applicazione è stata approvata verrà caricata automaticamente sugli store, a meno che lo sviluppatore non specifichi nelle opzioni una pubblicazione manuale.

2.2.4.4 BlackBerry

Dopo aver creato un'applicazione nativa per il sistema operativo BlackBerry, ci si trova difronte alla necessità di distribuire l'applicazione in via ufficiale sui canali di distribuzione adibiti a questo scopo. Possiamo intraprendere varie strade.

BlackBerry World

Questo è il canale ufficiale di distribuzione per le applicazioni BlackBerry. Come prima cosa bisogna creare gratuitamente un proprio account presso il BlackBerry World vendor portal. Si consiglia di leggere le linee guida e i criteri imposti da BlackBerry, che devono essere seguiti per far si che l'applicazione riceva l'approvazione ufficiale.

Una volta sviluppata e testa l'applicazione con il simulatore, si può mandare la release dell'applicazione per farla approvare. Quando si invia la release al portale l'applicazione occorre assegnargli una specifica licenza fra le seguenti:

- Free, applicazione scaricabile gratuitamente.
- Paid, applicazione acquistabile.
- 7-day, viene richiesto un pagamento periodico, il primo download può essere gratuito.
- 30-day, viene richiesto un pagamento periodico, il primo download può essere gratuito.

Se viene scelta una licenza a pagamento dovrà essere scelto un modello di licenza tra i seguenti, per fornire agli utenti delle License Key al fine di utilizzare l'applicazione.

- Static, non viene richista alcuna License Key.
- Single, dove viene fornita una sola License Key.
- Dynamic, dove viene utilizzato un sito per generare le License Key.
- Pool, dove vengono fornite una serie di License Key diverse per ogni applicazione scaricata.

BlackBerry World offre la possibilità di gestire le proprie applicazioni distibuite sul portale tramite l'utilizzo di SKU, un gestore di applicazioni, che permette di aggiornare, di modificare e di rimuovere le applicazioni caricate sul portale, molto utile per adattare le applicazioni ai cambiamenti che avvengono sul sistema operativo BlackBerry, come ad esempio il rilascio di una nuova versione.

Servizi di distribuzione

I servizi di distribuzione offerti da partners di BlackBerry supportano gli sviluppatori negli aspetti di marketing, analisi statistiche e fatturazione delle applicazioni. Aiutano gli sviluppatori anche nella distribuzione dell'applicazione nativa attraverso BlackBerry World.

I principali partner sono:

- GetJar.
- Handango.
- Handmark.

Enterprise Application Distribution

Tramite l'utilizzo del tool BlackBerry Enterprise Service, per versioni BlackBerry 7.1 o più vecchie, del tool BlackBerry Device Service, per le versioni più recenti è possibile sia inviare, sia modificare, sia aggiornare le applicazioni sul canale BlackBerry World. È possibile anche limitare la distribuzione in ambito aziendale.

Nel caso si utilizzi BlackBerry Enterprise Service verrà richiesto l'invio di un file in formato .zip, nel caso si utilizzi BlackBerry Device Service verrà richiesto un file in formato .bar .

2.2.4.5 Symbian

Dopo aver sviluppato e testato il corretto funzionamento dell'applicazione, per distribuirla sui canali ufficiali, che nel caso del sistema Symbian sono quelli forniti da Nokia, l'unico produttore ad utilizzare Symbian come sistema operativo per i propri smartphone. Dopo aver registrato un account presso il Nokia Publish ed aver ottenuto la licenza di sviluppatori Nokia, si può inviare l'applicazione sviluppata attraverso uno strumento chiamato Nokia Publish self-serve tool.

Una volta stabiliti tutti i parametri come prezzo, dispositivi mobile supportati, paesi e lingue, si può caricare l'applicazione nel Nokia Publish self-serve tool ed aspettare l'approvazione di Nokia Quality Assurance, che in caso l'applicazione rispetti tutte le direttive specificate nel Nokia Publisher Guide verrà caricata nel canale di distribuzione ufficiale, il Nokia Store.

2.3 Applicazioni indipendenti dal sistema operativo

Queste applicazioni nascono dall'esigenza di avere un certo livello di indipendenza dal sistema operativo che richiede di mandarle in esecuzione, per far fronte al problema legato alla portabilità delle applicazioni native, che per loro natura sono strettamente legate al sistema operativo d'origine.

L'idea che sta alla base di queste applicazioni è che una volta sviluppate possano essere adoperabili da un qualunque dispositivo mobile richieda il loro utilizzo. Perciò si è cercato di trovare un aspetto, una tecnologia, che fosse comune a tutti i sistemi operativi per dispositivi mobile. L'annello di congiunzione tra i molteplici sistemi operativi è rappresentato dalle tecnologie Web, che essendo presenti già da svariato tempo nell'ambito informatico dei computer hanno affermato degli standard riconosciuti a livello mondiale e sono tutt'oggi in continua evoluzione. Da qui sono emerse due metodologie per utilizzare queste tecnologie al fine di produrre applicazioni con alta portabilità sui vari sistemi operativi.

2.3.1 Applicazioni Web

La soluzione adottata da queste applicazioni garantisce un'elevata portabilità, poiché l'applicazione non è presente sul dispositivo mobile, ma viene eseguita su un server remoto che interagisce con il dispositivo mobile, questo permette di rendere indipendente l'applicazione dal sistema operativo che ne richiede l'esecuzione.

Queste applicazioni sfruttano i browser presenti sui diversi sistemi operativi, per interagire con l'utente. Ogni sistema operativo utilizza browser diversi, tra i quali possiamo citare i seguenti.

- Google Chrome.
- Mozilla Firefox.
- Dolphin Browser.
- Opera Classic.
- Next Browser.

- Safari.
- Atomic Web Browser.
- Browser Plus.
- QQBrowser.
- Terra Explorer.
- UC Browser.
- Surf Cube.

Ogni browser offre proprie caratteristiche e funzionalità e in base al sistema operativo su cui viene utilizzato avremo anche prestazioni differenti, perciò l'utente può scegliere per il proprio smartphone il browser che ù si addice alle proprie esigenze. Infatti l'istallazione dei browser sul dispositivo può essere facilmente effettuata dagli AppStore presenti sul sistema operativo.

Un applicazione Web di fatto è utilizzabile attraverso la rete internet o una qualsiasi rete privata, in base a com'è stata progettata. Queste applicazioni adottano un'architettura client-server, come mostrato in figura 2.5.

Queste applicazioni sono caratterizzate da diverse strutture ed architetture logiche, in base alla scelta degli sviluppatori. La struttura di un'applicazione Web può essere descritta generalmente su 3 livelli.

- **User Level** - questo livello può essere visto come il terminale dell'applicazione, si occupa di visualizzare i dati e le interfacce dell'applicazione utilizzando il browser del dispositivo. Questa funzione può essere offerta tramite pagine HTML con CSS, oppure tramite framework specifici.
- **Server Level** - questo livello è caratterizzato da un motore applicativo che risiede sul server, che utilizza linguaggi di programmazione dinamici lato server, per offrire i servizi richiesti dall'applicazione. Tra i linguaggi utilizzati possiamo citare PHP, ASP, Java, ecc ...

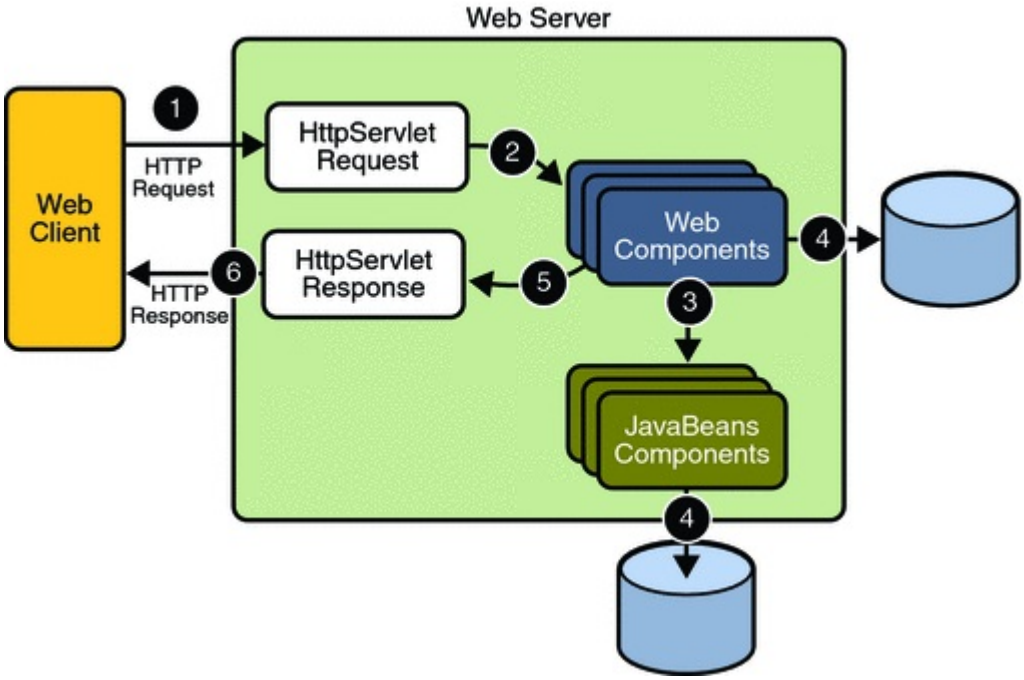

Figura 2.5: Tipica struttura per applicazione Web [8].

- **Database Level** - questo livello è composto da un database associato all'applicazione, per gestire tutti i dati utilizzati dalla stessa. Tipici database utilizzati sono MySQL, Oracle, ecc ...

Un'applicazione Web in base a com'è stato progettato il sistema può avere diverse tipologie di cicli di vita, un esempio è quello mostrato in figura 2.6.

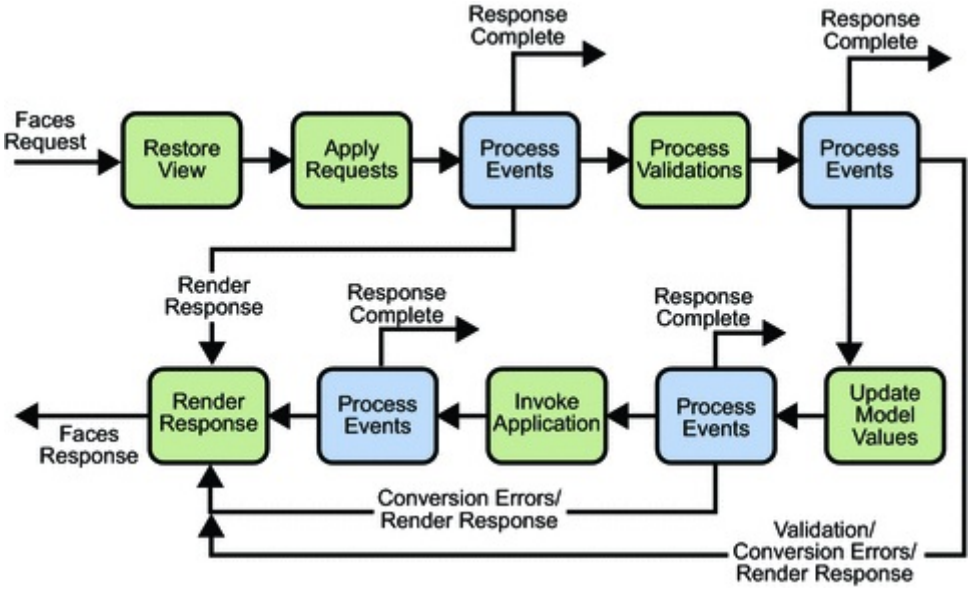

Figura 2.6: Ciclo di vita di un'applicazione Web [8].

2.3.2 Applicazioni Platform Indipendent

Queste applicazioni vengono sviluppate attraverso dei framework, che permettono il loro sviluppo tramite linguaggi di programmazione tipici delle tecnologie Web. Lo sviluppo di applicazioni attraverso questi framework non permette di avere un'unica applicazione compatibile con più sistemi operativi, ma offre la possibilità di creare varie versioni della stessa applicazione, a partire dal medesimo codice sorgente, ognuna compatibile con un diverso sistema operativo. Una volta creata l'applicazione, il framework ha il compito di tradurre il linguaggio con cui è stata scritta l'applicazione, nel

linguaggio nativo utilizzato dal sistema opertativo scelto. Le applicazioni risultanti da questa operazione di traduzione sono applicazioni ibride, perciò non possono essere considerate né applicazioni native né applicazioni Web, ma avranno buona parte di codice nativo e una parte di codice legato alle tecnologie Web. Questo lo si può constatare perché queste applicazioni non utilizzeranno solo i framework nativi, come ad esempio il framework adibito alle interfacce grafiche e a differenza delle applicazioni Web, durante la trasformazione in codice nativo, l'applicazione andrà ad utilizzare le API del sistema scelto, per garantire le stesse funzionalità che sono state espresse nel codice sorgente.

Fra i framework più utilizzati, sia open-source che privati, per lo sviluppo di queste applicazioni troviamo:

- PhoneGap.
- Appcelerator Titanium.
- MotherApp.
- Rhomobile.
- Tersus.
- Unity.
- Flex.

2.4 Applicazioni basate su tecnologie Web

L'utilizzo delle tecnologie Web per la creazione di applicazioni, che non dipendo dal sistema operativo ha portato a considerare la loro utilizzo anche nella creazione di applicazioni per un determinato sistema operativo. Queste applicazioni nascono con l'idea di sfruttare le tecnologie Web come alternativa all'utilizzo del linguaggio nativo proprio di un sistema operativo.

2.4.1 Prospettiva

Queste applicazioni utilizzano le tecnologie Web presenti sul dispositivo, per esprime attributi e funzionalità che le caratterizzano. L'utilizzo di tecnologie

Web non impone la necessità di utilizzare una connessione internet o ad una rete locale, infatti queste applicazioni possono essere lanciate in esecuzione sul dispositivo e tutti i dati necessari per il loro funzionamento sono presenti nella memoria del dispositivo mobile.

Come ci si può aspettare, le tecnologie Web non possono coprire tutte le funzionalità offerte dal dispositivo mobile, perciò questo approccio porta allo sviluppo di applicazioni ibride composte da parti di codice sviluppate in linguaggio nativo e le restanti con i linguaggi utilizzati per sfruttare le potenzialità delle tecnologie Web presenti sul dispositivo, come ad esempio HTML, CSS e Javascript. Questa prospettiva di sviluppo per applicazioni mobile apre la strada a numerosi aspetti, che devono essere valutati in merito ai vantaggi che può portare questo approccio, perciò risulta essere un interessante ambito di ricerca e innovazione in questo mondo in continua evoluzione.

2.4.2 Tecnologie Web

Come già detto queste applicazioni utilizzano tecnologie Web, che parallelamente all'evoluzione dei dispositivi mobile hanno avuto una certa evoluzione passando da standard, che col tempo sono diventati obsoleti, agli standard che vengono utilizzati attualmente. È interessante quindi fare un'analisi dello stato evolutivo in cui ci troviamo [23].

2.4.2.1 Standard obsoleti: il WML

Il WML è stato incorporato nel primo standard WAP 1.1 ed è stato il primo standard per web mobile. Oggi è un linguaggo estremamente sconsigliato, va considerato come un linguaggio del passato, poiché tutti i telefoni cellulari sono in grado di comprendere questo linguaggio, ma nel caso dei moderni smartphone, che montano solitamente browser basati sul WebKit come iPhone e Android, non riescono a visualizzare questo formato, proprio per il fatto che è una tecnologia superata ormai da tempo.

Un documento WML è un file XML, con diversa estensione e sotto certi punti di vista è simile all'HTML, ma si differenzia per molto altro. Il WML non supporta immagini GIF, JPG, PNG, non si serve di Javascript, ma ha un proprio linguaggio di scripting chiamato WMLScript. Il WML è stato eliminato dallo standard WAP nel 2002, ma continua ad essere usato

come linguaggio di ripiego per dispositivi mobile di vecchia produzione. Oggigiorno solo i siti delle compagnine telefoniche sono obbligate ad avere anche una versione WML, per garantire una piena compatibilità con tutti i dispositivi mobile in circolazione. Infatti difficilmente ci troveremo ad utilizzare questa tecnologia, ma se proprio ci viene richiesto sarà per un puro fatto di compatibilità universale.

2.4.2.2 Standard Attuali

Gli standard attuali si trovano racchiusi in vari gruppi, per ragioni dovute a chi li ha sviluppati, che ha dato una propria visione di come dovrebbe essere lo standard mondiale da utilizzare.

Standard per il Web mobile del W3C

- XHTML Basic 1.0 e 1.1
- CSS Mobile Profile

Standard per il Web mobile della OMA

- XHTML Mobile Profile 1.0, 1.1 e 1.2
- Wireless CSS o WAP CSS

Standard Web generici dei produttori di dispositivi

- XHTML 1.0
- HTML 3.2 e 4.0
- CSS 2.1

Standard Web generici ancora in via di definizione

- HTML 5.0
- CSS 3.0

Esistono anche varie estensioni che utilizzano i produttori legate all'X-HTML e ai CSS usati.

2.4.2.3 Come gestire più Standard

Tutti gli standard sono simili e compatibili tra di loro, ad eccezione delle nuove funzionalità presenti su HTML 5 e CSS 3, in più tutti i browser basati su HTML implementano un meccanismo di tolleranza per gestire tag ed attributi, che sono alla base dell'evoluzione di HTML.

2.4.2.4 HTML5

Questa nuova versione della tecnologia HTML, rispetto la precedente versione HTML4 introduce cambiamenti, maggiore flessibilità e nuove funzionalità riguardanti controlli e API, che sono stati dettati dalle necessità venutesi a creare intorno allo scenario delle applicazioni mobile. Le novità introdotte da questa nuova versione dell'HTML sono state pensate per supportare la memorizzazione di grosse quantità di dati e per supportare le applicazioni basate su tecnologie Web, anche in assenza di connessione alla rete internet migliorando così l'indipendenza fra la struttura, i contenuti e i fogli di stile utilizzati da questa tecnologia. Le nuove funzionalità introdotte da questa versione dell'HTML sono state estese tramite l'introduzione di nuove API JavaScript riguarti i seguenti aspetti.

- Salvataggio dati su dispositivi mobile.
- Utilizzo in assenza di connessione.
- Operazioni in backgroud.
- Supporto multimediale.
- Supporto Canvas per le animazioni.

- Generare grafica 2D.
- Generare grafica 3D.
- Gestione dati provenienti da sensori multimediali.
- Geolocalizzazione.

Le modifiche introdotte riguardanti la struttura hanno portato ad abbandonare l'utilizzo delle sezioni DIV, poiché da queste sezioni sono stati implementati ed evoluti nuovi elementi chiamati header, nav, section, article, aside, e footer come mostrato in figura 2.7.

È facilmente intuibile che l'elemento header rappresenta la parte superiore della pagina e l'elemento footer rappresenta quella inferiore. Gli elementi restanti sono più specifici, il nav è adibito a contenere i link per la navigazione, article si occupa di mostrare i contenuti della pagina, il section riassume i contenuti presenti, mentre aside si occupa di ciò che sta intorno ai contenuti, come per esempio sidebars, menù, ecc ...

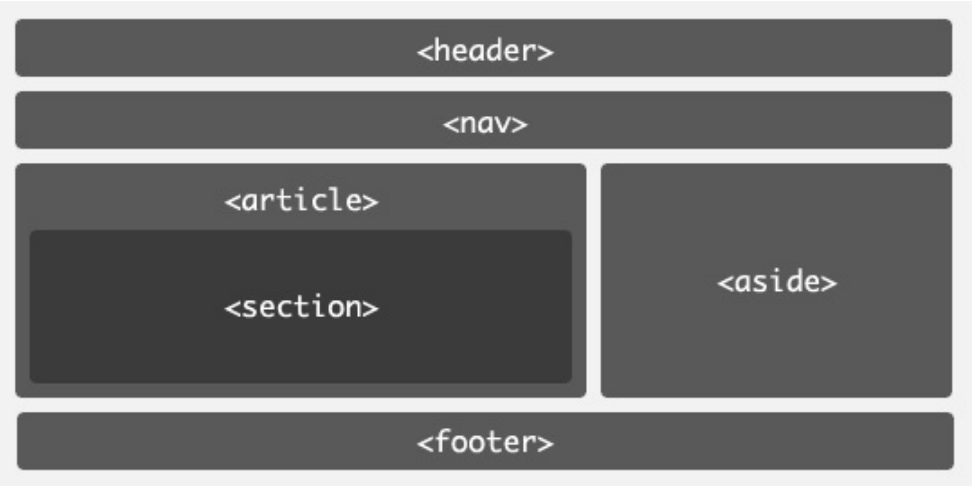

Figura 2.7: Struttura HTML5 [9].

Per quanto riguarda la gestione del multimedia sono stati introdotti tag appositi come *video* e *audio*. Questi tag sono caratterizzati da attributi e funzionalità specifiche, come ad esempio *video* permette tramite l'attributo

POSTER di visualizzare un'immagine durante il caricamento del filmato, *audio* introduce l'attributo CURRENTTIME e metodi che possono essere invocati come play() e pause().

Con la nuova versione dell'HTML5 è stato esteso e potenziato il DOM (Document Object Model), per la gestione e la visualizzazione dei documenti. Il DOM da la possibilità ai documenti di contenere un linguaggio indipendente dalla sintassi, infatti si posso utilizzare differenti sintassi per rappresentare un documento HTML, come l'HTML5 e l'XHTML5 e questo porta un notevole vantaggio.

2.4.2.5 CSS 3

Questa nuova versione della tecnologica CSS sviluppata da W3C, si introduce una maggiore modularità fra i vari aspetti riguardanti proprietà e metodi, che CSS propone per aggevolare lo sviluppo di siti web. Questa tecnologia si pone come obbiettivo di supportare le altre tecnologie Web, come ad esempio l'HTML5 mettendo a disposizione funzionalità riguardanti i seguenti ambiti.

- Creazione di effetti grafici di alto livello.
- Controllo delle caratteristiche dei contenuti.

Questa nuova versione è caratterizzata da una piena compatibilità con le versioni precedenti, infatti tutti gli aspetti e le funzionalità delle precedenti versioni vengono mantenute inalterate, di fatto vengo aggiunte molte proprietà per aumentare le capacità di questa tecnologia. La caratteristica che rappresenta maggiormente questa versione è la modularità, infatti W3C suddivide le specifiche aree dei CSS in moduli, che hanno per l'appunto una vita evolutiva propria permettendo di sviluppare e potenziare i vari settori in maniera completamente indipendente, in base anche alle esigenze che vengono a crearsi negli utenti di questa tecnologia.

Per informarsi sullo stato dell'arte di questa tecnologia, si possono trovare tutte le informazione riguardanti il progresso dei vari moduli, sul sito ufficiale [11].

2.4.3 Analisi dell'approccio

Questo modo di concepire le applicazioni per i dispositivi mobile risulta essere molto interessante, in quanto permette di utilizzare sia aspetti legati al sistema operativo, sia aspetti legati alle tecnologie Web che il sistema operativo supporta.

Dato che le tecnologie Web sono continuamente in costante evoluzione, per via del loro forte utilizzo nei vari ambiti del mondo informatico, la loro integrazione nell'ambito delle applicazioni per dispositivi mobile può rappresentare importanti sbocchi evolutivi nello sviluppo di applicazioni, con funzionalità sempre più ampie e di più alto livello. L'integrazione di tecnologie Web porta ad ottenere vantaggi riguardanti la portabilità di queste applicazioni, anche se parziale, ma è comunque degno di nota considerare che buona parte delle applicazioni sviluppate seguendo questo approccio, si potrà riutilizzare anche nella produzione della medesime applicazioni per sistemi operativi mobile differenti, semplicemente andando a sviluppare quelle parti relative al linguaggio nativo che l'applicazione necessita.

Queste applicazioni offrono ampie possibilità di ricerca, poiché nell'ambito delle caratteristiche prestazionali e dei consumi legati ad ogni applicazione, che sono gli aspetti cardinali che influenzano lo sviluppo delle applicazioni, bisogna sempre valutare se l'utilizzo di queste tecnologie porterà ad avere dei miglioramenti riguardo una specifica operazione, rispetto all'esclusivo utilizzo di codice nativo.

2.5 Confronto sulle diverse tipologie di Applicazioni

Tra le due tipologie principali di applicazioni, le applicazioni native e le applicazioni indipendenti, non è possibile determinare a priori quale sia una migliore scelta rispetto all'altra, perché ogni tipologia di applicazione porta con se sia vantaggi che svantaggi, perciò in base allo scopo per cui viene sviluppata un'applicazione, bisogna decidere accuratamente quale approccio utilizzare.

2.5.1 Vantaggi

Lo sviluppo di un'applicazione nativa offre diversi vantaggi:

- Essendo un'applicazione sviluppata appositamente per un determinato sistema operativo, risulta essere molto performante e può sfruttare tutte funzionalità che il sistema operativo mette a disposizione.
- Si ha una maggiore facilità nell'accedere ai dati e l'accesso è più veloce, poiché i dati risiedono sul dispositivo mobile.
- Si possono utilizzare tutte le risorse hardware presenti sul dispositivo, come GPS, oscillometro, giroscopio, ecc ...
- Non dipendono necessariamente dall'accesso alla rete, a meno che sia l'applicazione stessa ad avere funzionalità che ne richiede l'utilizzo.
- Sono facilmente reperibili, perché si possono consultare tutte le informazioni relative ed installare le applicazioni sui disposi mobile, direttamente utilizzando gli strumenti che il sistema operativo mette a disposizione dell'utente, gli App Store, che forniscono ulteriori garanzie e certificazioni per la non nocività dell'applicazione.
- Offre sistemi di sicurezza contro la pirateria, poiché è difficile estrarre i contenuti che l'applicazione utilizza per il proprio funzionamento.

Per quanto riguarda un'applicazione indipendente dalla piattaforma, il vantaggio più lampante è dato dal fatto che una volta sviluppata, può essere distribuita velocemente per la maggior parte, se non tutti, i dispositivi mobile smartphone, indipendentemente dal sistema operativo che montano. Per quanto riguarda lo sviluppo, un vantaggio molto significativo è rappresentato dal fatto che per la produzione dell'applicazione, da distribuire su più sistemi operativi, si necessita l'utilizzo di un solo linguaggio di programmazione, così da ridurre notevolmente il tempo necessario per il suo sviluppo.

2.5.2 Svantaggi

Il principale svantaggio di un'applicazione nativa è la portabilità, poiché sarà compatibile solo con il sistema operativo per la quale è stata sviluppata. C'è anche da dire che i continui aggiornamenti delle versioni di un sistema operativo possono portare a dover aggiornare e modificare un'applicazione nativa, che in alcuni casi perde la compatibilità con il sistema o

semplicemente si trova ad avere vari bug in determinate aree del proprio funzionamento.

Per quanto riguarda gli sviluppatori, se si vuole creare un'applicazione con varie versioni native per i diversi sistemi operativi presenti sul mercato, questo necessita una buona conoscenza di diversi linguaggi di programmazione e un investimento maggiore di tempo per lo sviluppo delle varie versioni.

Analizzando il caso delle applicazioni indipendenti, gli svantaggi più evidenti possiamo elencarli nel seguente modo.

- Le applicazioni indipendenti sono solitamente meno performanti, poiché durante la loro esecuzione parte del codice utilizzato deve essere tradotto in linguaggio nativo.
- Gli sviluppatori sono costretti ad utilizzare solo gli aspetti e le funzionalità che i diversi sistemi operativi hanno in comune.
- I sistemi hanno diverse convenzioni per quanto riguarda l'interfaccia grafica, quindi lo sviluppatore deve stare attento a rispettare queste caratteristiche, per evitare che l'applicazione risulti estranea, non intuitiva e di difficile utilizzo all'utente.
- Essendo applicazioni non progettate per uno specifico sistema soffrono maggiormente di problemi relativi alla sicurezza.
- Testare queste applicazioni risulta più difficoltoso.

Nel caso specifico delle applicazioni Web, ci sono svantaggi ulteriori, infatti un'applicazione Web richiede un costante accesso alle rete internet, questo porta in alcuni casi a non dare la possibilità all'utente di utilizzare l'applicazione in ogni momento. Un altro problema per l'utilizzo di applicazioni Web è legato ai consumi della connessione, infatti gli operatori telefonici mettono a disposizione dell'utente, un certo numero di GB utilizzabili al mese, in base al piano tarifario. Questa è un'ulteriore limitazione delle applicazioni Web, che se non ottimizzate o per loro natura hanno consumi elevati di connessione risultano poco vantaggiose per l'utente. È da considerare anche il fatto che le prestazioni di un'applicazione Web dipendono in modo sensibile dalla rapidità della connessione, perciò è difficile a priori garantire un costante livello prestazionale.

Per quanto riguarda le applicazioni platform indipendet, la portabilità di queste applicazioni è legata al framework utilizzato, perciò non tutti i framework grarantiscono lo stesso livello di portabilità. In più queste applicazioni hanno svantaggi maggiori a livello prestazionale, che sono dovuti al fatto dell'automatismo di traduzione del codice scritto in codice nativo e questo porta ad avere applicazioni con codici nativi non ottimizzati. Questo deficit prestazione, varia in base a com'è stata progettata l'applicazione e alle funzionalità che la caratterizzano.

2.5.3 Valutazione applicazioni basate su tecnologie Web

Tra le varie applicazioni per dispositivi mobile, negli ultimi tempi si sta concentrando l'attenzione sullo sviluppo di applicazioni ibride, che si servono di tecnologie Web. Queste applicazioni offrono interessanti casi di studio, che riguardano i vantaggi che si possono ottenere dal loro utilizzo e dalla loro ottimizzazione, quindi rappresentano un importante ambito di ricerca, nel quale molti dei più importanti produttori di dispositivi mobile stanno concentrando energie e risorse. I primi e più famosi sistemi operativi per dispositivi mobile, come Android, iOS e Window Phone sono stati concepiti inizialmente per l'utilizzo di applicazioni native, per motivi legati all'uso di applicazioni il più possibile performanti, al fine di venir in contro alle esigenze legate alle limitate disponibilità hardware. Negli ultimi tempi anche su questi sistemi operativi, l'aspetto legato alle applicazioni basate su tecnologie Web è stato caso di notevole interesse, infatti il recente sviluppo di questa tipologia di applicazioni ha portato a iniziare a concepire sistemi operativi improntati sull'utilizzo di applicazioni basate su tecnologie Web. Tra i sistemi operativi mobile di questa nuova generazione, che sono nati con l'idea di utilizzare le tecnologie Web come supporto allo sviluppo e all'esecuzione delle applicazioni sul proprio sistema, per il caso di studio relativo a questa tesi si è scelto il sistema operativo Tizen, che verrà analizzato e valutato nel capitolo successivo.

Capitolo 3

Caso di studio Tizen

Per la creazione di questo capitolo sono state raccolte ed analizzate le informazioni presenti sul sito ufficiale della piattaforma Tizen [18].

3.1 Introduzione

Tizen è una piattaforma open source ideata per sviluppare applicazioni per dispositivi mobile, tramite l'utilizzo di tecnologie Web e API native. Questa piattaforma fà parte di un progetto creato dalla Linux Foundation e può essere installato su diversi dispositivi mobile come smartphone , tablet , netbook e smart Tv. Questa piattaforma permette di creare applicazioni con un ampio range di funzionalità, che sono basate sull'architettura di Tizen.

3.2 Caratteristiche

La piattaforma Tizen è attualmente alla versione 2.2 ed è un sistema in continua evoluzione. Una caratteristica importante di Tizen riguarda la possibilità di sfruttare due diversi tipi di framework per lo sviluppo delle proprie applicazioni. Tizen è dotato di un Web Framework per le applicazioni Web e un Native Framework per lo sviluppo di applicazioni native. Questi due framework possono essere utilizzati anche per creare applicazioni ibride per aumentare le potenzialità delle applicazioni sviluppate. Qualisiasi

framework si utilizzi per sviluppare un'applicazione, Tizen assicura che le applicazioni saranno compatibili tra loro avendo un aspetto coerente.

Tizen offre un protocollo di sicurezza basato su privilegi e le applicazioni devo essere certificate. Tizen permette di creare per le applicazioni sviluppate, una lista di requisiti minimi per filtrare componenti hardware e software che non garantirebbero un buon funzionamento dell'applicazione. Tizen è una piattaforma che offre agli utenti un'esperienza completa al pari degli altri sistemi operativi più in voga.

3.2.1 Interfaccia grafica

L'interfaccia grafica di Tizen offre all'utente vari aspetti e funzionalità. Analizzando questa interfaccia grafica che viene adottata da Tizen, si può denotare una serie di elementi riguardanti l'interazione, il design e stili di visualizzazione scelti per questa piattaforma. Buona parte di questi può essere ritrovata in altri sistemi operativi più utilizzati, ma comunque si denotano anche interessanti novità e dal punto di vista architetturale, quindi risulta essere una piattaforma molto ben progettata ed intuitiva.

3.2.1.1 L'interazione

Ogni sistema operativo per dispositivi mobile deve dedicare particolare attenzione alla progettazione di questo caratteristica, poiché gli utenti basano le proprie valutazioni sull'impatto che questo aspetto fornisce alla loro user experience. Tizen implementa questa caratteristica attraverso l'utilizzo di architetture, componenti e funzionalità che sono pensate per dare all'utente una user experience di alto livello.

Struttura delle applicazioni

Le applicazioni possono suddividere le proprie interfaccie grafiche in 2 livelli, Top e Lower Level. Il Top Level racchiude le interfaccie principali, che possiamo chiamare main view, mentre nel Lower Level troviamo tutte le interfaccie secondarie che possiamo chiamare detail view.

Queste interfaccie possono essere utilizzate in due modalità, Normal Mode ed Edit Mode. In Normal Mode, le main view di ogni applicazione

permettono all'utente di visualizzare, di creare e di gestire gli elementi contenuti nel'applicazione, mentre le detail view visualizzano i contenuti e gli elementi dell'applicazione come testo o immagini, al fine di esegure rapidamente le funzioni offerte e navigare attraverso le view dell'applicazione. In Edit Mode, le main view di ogni applicazione permettono all'utente di gestire più elementi ed esegure più funzioni contemporaneamente, invece le detail view permetteno di creare, di eliminare e di modificare contenuti presenti nell'applicazione e le impostazioni della stessa.

Visualizzazione

Tizen permette di gestire la visualizzazione del display seguendo tre diverse modalità.

- **Portrait and landscape view**, classico cambio da una visualizzazione verticale ad una orrizzontale.
- **Portrait with a split view**, quando il dispositivo si dispone in orrizzontale, si andranno a formare due schermate verticali affiancate, come pagine di un libro.
- **Full landscape view**, la schermata rimane sempre orrizzontale.

Per quanto riguarda la split view, l'utente può utilizzare due tipi di split view, Fixed pane e Adjustable pane. Il Fixed pane impone la larghezza delle due aree prefissata, di default quella di sinistra mantine la stessa larghezza che si aveva nella portrait view. L'Adjustable pane permette all'utente di modificare la dimensione di queste aree a proprio piacimento.

Dynamic Box

Questo componente è un elemento presente nelle applicazioni, che mostra dinamicamente dati di applicazioni, informazioni prese dalla rete, nome e dettagli dell'applicazione, immagini, icone. La Dynamic Box risulta essere un componente utilizzabile dalle applicazioni, con un certo livello di indipendenza, infatti possiamo vedere questo componente come un'applet dell'applicazione, l'unico legame è rappresentato dalla chiusura dell'applicazione che porta alla chiusura stessa della Dynamic Box. La funzione

interessante di questo componente è che permette durante l'interazione di visualizzare una drop view, che prima era nascosta, dove possiamo trovare sia ulteriori informazioni, che funzioni legate alla Dynamic Box.

Multi-window

Tizen permette di visualizzare le interfaccie delle applicazioni, sia in modalità full window che in modalità mini window. La prima è la classica visualizzazione full screen che viene adottata di solito, la seconda permette di visualizzare l'interfaccia delle applicazioni in una finestra di dimensioni ridotte rispetto allo schermo, con la possibilità di utilizzare varie funzionalità. Queste finestre devono essere dotate di bottoni che permettono di chiudere l'applicazione o di passare alla modalità full window, inoltre la finestra può avere un nome, ma quando non è presente può essere inserito un bottone mostra/nascondi. Altre funzionalità che possiamo includere riguardano lo spostamento, il ridimensionamento della finestra.

In altre parole queste funzioni sono le tipiche modalità di gestione delle finestre nei sistemi operativi per computer, sicuramente un aspetto molto innovativo per i sistemi operativi smartphone.

Accessibilità

Tizen viene incontro anche ad utenti disabili offrendo funzioni per facilitare l'utilizzo. Le funzioni in questione sono rappresentate da uno Screen reader e un Font size adjustment. Il primo permette di aiutare gli utenti non vedenti traducendo tutte le scritte e i componenti grafici presenti in un'applicazione, in un contributo vocale servendosi di un innovativo componente, il TTS (Text-to-Speech). Il secondo permette di ingrandire o ridurre le dimensioni delle scritte rispetto a quelle impostate di default.

3.2.1.2 Componenti di interfaccia

L'interfaccia si divide principalmente in tre aree, un Header, un Body e un Footer. L'Header risiede nella parte superiore dell'interfaccia e contiene i titoli relativi all'applicazione, si può dotare anche di bottoni per eseguire funzioni associate all'applicazione, in più può essere suddiviso in sezioni attraverso dei tab. Il Body è la sezione riguardante la visualizzazione di tutti

i contenuti dell'applicazione, inoltre in questa sezione si possono utilizzare diversi tipi di liste e menù.

- List.
- Grouped List.
- Index List.
- Expandable List.
- Grid List.

Il Footer risiede nella parte più bassa dell'interfaccia e permette di inserire fino a 3 bottoni per l'utilizzo di funzioni base per l'applicazione. A questi bottoni possono essere associati anche dei menù Pop-Up per aumentare le funzionalità offerte.

Esistono poi una serie di componenti che possono essere integrati nelle interfaccie grafiche, tra i quali troviamo:

- **Scroll**, come scroll bar, jump to top button, screen handler, fast scroll.
- **Controls**, come bottoni, bottoni radio, check box e bottoni on/off.
- **Slider**, barre per regolare impostazioni come volume, luminosità, ecc ...
- **Progress**, barre che visualizzano lo stato di un processo.
- **Pop-Up**, che sono centrati sullo schermo o relativi gli elementi generanti.
- **Search**, campi di ricerca inseribili sia nell'Header che in cima alle liste nel Body.

3.2.2 Architettura della piattaforma

L'architettura della piattaforma Tizen è basata su 4 livelli, come mostrato in figura 3.1.

Ogni livello è incaricato di occuparsi di aree di interesse specifiche e mette a disposizione del livello superiore le proprie capacità. Andiamo quindi ad analizzare i vari livelli che compongono questa piattaforma.

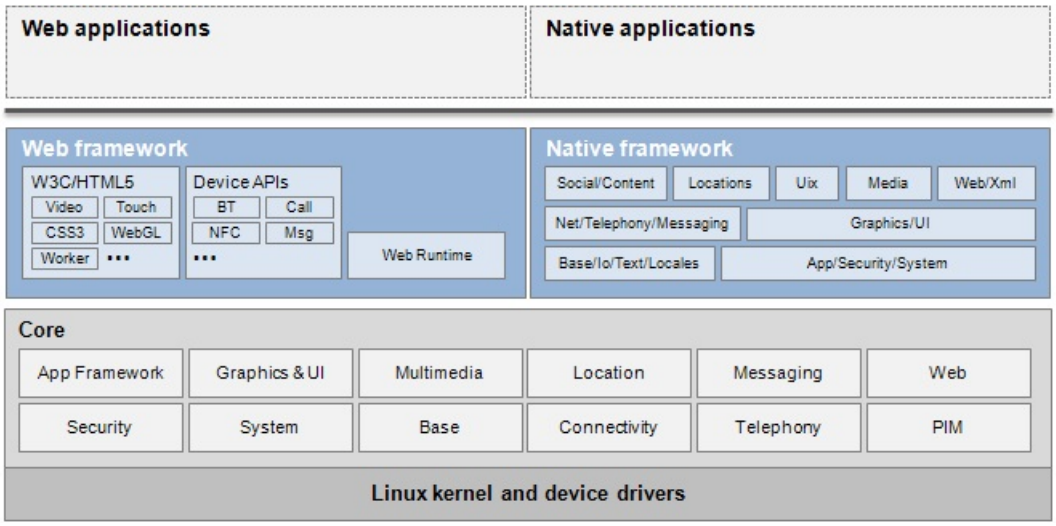

Figura 3.1: Architettura della piattaforma Tizen [10].

3.2.2.1 Applicazioni

Nella parte più alta di questa architettura risiedono le applicazioni che sono supportate dal livello sottostante. In questo livello risiedo tutte le applicazioni presenti sul dispositivo, infatti si hanno tutte le applicazioni offerte di default dal sistema, ulteriori applicazioni native e applicazioni basate su tecnologie Web installate successivamente dall'utente sul dispositivo.

3.2.2.2 Framework

Nel secondo livello troviamo i framework sui quali si appoggiano le applicazioni sviluppate in Tizen, il WebFramework e il Native framework. Il primo framework sfrutta le tecnologie web all'avanguarda come l'HTML5, che offre tante funzionalità come WebGL e CSS3, geolocalizzazione, Web Soket e Web Worker. Oltre alle tecnologie già citate vengono messe a disposizioni numerose API, per permettere di utilizzare funzionalità del dispositivo mobile come Bluetooth, NFC, di allarme e di messaggistica. Questo framework adotta un sistema di sicurezza che impone rigide regole per evitare lo scorretto utilizzo delle API offerte. Il secondo framework offre tutti i componenti, i servizi di sistema e le API per lo sviluppo di un efficiente ap-

plicazione nativa. Questo framework è composto da numerose librerie come glibc, libstdc++, libxml2 e librerie audio e video come OpenGL ES, OpenAL e OpenMP per consentire lo sviluppo di applicazioni molto performanti e permettere il porting di applicazioni che utilizzano queste librerie.

3.2.2.3 Core

Il terzo livello è rappresentato dal Core che offre tutte le funzionalità richieste dal Web Framework e dal Native Framework, in questo livello sono presenti tutte le API e le librerie open source che vengono utilizzate dal livello superiore.

3.2.2.4 Linux kernel and device driver

Questo livello è quello più vicino all'hardware ed è composto da un Linux Kernel e da tutti i driver necessari per l'astrazione delle risorse hardware del dispositivo mobile sul quale è installato Tizen.

3.2.3 Sicurezza della piattaforma

La sicurezza di Tizen è stata organizzata seguendo una certa architettura, per proteggere la piattaforma e i dati dell'utente che risiedono sul dispositivo, che si basa su un sistema di privilegi e di certificazione delle applicazioni. Questa architettura si basa sul modello base di sicurezza di Linux, che include meccanismi di isolamento dei processi e un controllo obbligatorio degli accessi.

Andiamo di seguito ad analizzare le funzionalità adibite alla sicurezza utilizzate da Tizen.

3.2.3.1 Privilegi

Le applicazioni che vogliono utilizzare determinate API, che sono ritenute più critiche, per utilizzarle devono dichiarare di essere in possesso dei necessari privilegi. In genere le API che richiedono un sistema di privelegi sono quelle che possono andare a violare e danneggiare la stabilità della piattaforma e la privacy dell'utente, se utilizzate in maniera scorretta. Le applicazioni che intendono utilizzare tali API devono dichiarare che sono in possesso dei privilegi necessari, questa dichiarazione deve essere fatta nel

file config.xml per le applicazioni Web o nel file manifest.xml per le applicazioni native. I privilegi che si possono adottare sono di tre tipologie, in base all'impatto che hanno a livello di sicurezza.

- **Public** - questi privilegi sono in possesso di tutti gli sviluppatori Tizen.

- **Partner** - questi privilegi possono essere utilizzati sono dagli sviluppatori registrati come partner del Tizen Store. Per utilizzare questi privilegi basterà applicare una patch chiamata partner certificate patch, all'IDE utilizzato per lo sviluppo di applicazioni.

- **Platform** - questi privilegi sono utilizzati dalle API di sistema più sensibili, per la gestione della stessa piattaforma Tizen. Questi privilegi possono essere utilizzati attraverso l'aplicazione di una patch, ma solo sull'emulatore o utilizzati dai file binari, ma non potranno mai essere installate sui dispositivi fisici applicazioni in possesso di questi privilegi.

3.2.3.2 Certificazione delle applicazioni

Le applicazioni Tizen una volta ultimate devono essere certificate sia dall'autore che dal distributore, in base alle linee guida descritte da W3C nel documento XML Digital Signatures for Widgets. Perciò il packpage della nostra applicazione dovrà essere in possesso di alcuni certificati.

Certificazione autore

Una firma d'autore che certifica l'integrità di un packpage, così com'è stato fornito dallo sviluppatore e ne garantisce l'affidabilità. Questo certificato è ottenuto tramite tool presenti nell'SDK di Tizen, che sono creati dall'unione della firma d'autore e del certificato Tizen Developer Certificate Authority.

Certificazione distributore

Questo certificato è generato dall'applicazione incaricata alla distribuzione, come per esempio il Tizen Store, che certifica che l'editore dell'applicazione ha distribuito la propria applicazione garantendo l'integrità del package distribuito. Questo certificato determina i livelli di privilegio utilizzati dall'applicazione.

3.2.3.3 Gestione delle eccezioni

Quando un'applicazione tenta di utilizzare API per accedere ad informazioni private, ma viene bloccata dalle impostazioni di sicurezza riguardanti la privacy dell'utente, si verifica questo blocco tramite il lancio delle eccezioni. Se l'applicazione è Web verrà lanciato un securityError, se nativa verrà lanciato un E_USER_NOT_CONSENTED. Sta allo sviluppatore gestire questi eventi per evitare malfunzionameti dell'applicazione.

La piattaforma Tizen al primo lancio in esecuzione di un'applicazione mostra un pop-up riguardante i privilegi associati all'applicazione e permette all'utente anche di revocare quei privelegi che non ritiene necessari, che sono stati conferiti all'applicazione. Ad esempio per un'applicazione che utilizza il Wi-Fi, l'utente durante la prima esecuzione può decidere di negargli il permesso di utilizzare quella risorsa, semplicemente disabilitando il privilegio associato.

Per quanto rigarda i privelegi legati alla privacy, Tizen mette a disposizione i seguenti.

- Accounts.
- Bluetooth.
- Calendar.
- Call log.
- Contacts.
- Location.
- Messaging.

- NFC.
- Wi-Fi Direct.

3.2.4 Application Packpage Manager

Questo componente è uno dei principali moduli dei framework per le applicazioni Tizen, si occupa infatti dell'installazione, dell'aggiornamento, della disinstallazione dei packpage delle applicazioni e della memorizzazione dei relativi dati. Utilizzando questo componente è possibile anche recuperare informazioni relative alle applicazioni installate sul dispositivo. Questo componente si può espandere per supportare diversi tipi di applicazioni, tramite l'installazione dei relativi moduli. Possiamo dividere l'Application Packpage Manager in componenti più specifici.

Il Tizen Native Application Package Manager per gestire le applicazioni native delle seguenti tipologie.

- Applicazioni UI.
- Applicazioni Service.

Tizen è provvisto dei seguenti componenti per gestire le applicazioni che sfruttano tecnologie Web.

- Web Application Package.
- Hybrid Application Package.

3.2.5 Filtri

Tizen mette a disposizione anche la possibilità di impostare opportuni filtri per creare una sorta di requisiti minimi, che l'applicazione richiede per essere eseguita su un dispositivo mobile. Questi filtri possono essere dichiarati nel file config.xml o nel file manifest.xml e servono per evitare malfunzionamenti legati ad aspetti hardware o software del dispositivo. I filtri riguardanti gli aspetti hardware del dispositivo possono essere legati da aspetti come la dimensione dello schermo, oppure la presenza di tecnologie come il GPS. I filtri riguardanti caratteristiche software sono legati ad aspetti quali funzioni richieste dall'applicazione per la sua esecuzione.

Quando l'applicazione viene mandata in esecuzione può succedere che i requisiti specificati non siano soddisfatti, per prevenire questo problema c'è la possibilità di eseguire una verifica sulle caratteristiche del dispositivo, tramite l'utilizzo dell'interfaccia SystemInfoDeviceCapability per le applicazioni Web e del metodo GetValue() della classe Tizen::System::SystemInfo per le applicazioni native. Questo permette di impostare una serie di possibili requisiti richiesti, come ad esempio se l'applicazione richiede un servizio di localizzazione, allora verrà richiesto ed utilizzato il GPS se il dispositivo possiede il GPS, invece verrà richiesto ed utilizzato il WPS se il dispositivo possiede il WPS, ma nel caso non sia soddisfatto almeno uno dei requisiti presenti fra le varie opzioni, l'esecuzione dell'applicazione verrà interrotta.

3.3 Tools di sviluppo

Tizen mette a disposizione degli sviluppatori una serie di tools da utilizzare per sviluppare le applicazione tramite l'installazione di un opportuno SDK, fornito nel sito ufficiale di Tizen, che mette a disposizione un IDE relativo alla piattaforma di sviluppo Eclipse. Questo SDK contiene librerie, platform binaries e header di Tizen e un IDE per lo sviluppo di applicazioni contenente tools di sviluppo Javascript e tools di sviluppo di Eclipse.

3.3.1 Tools generici

L'SDK di Tizen mette a disposizione i seguenti tools.

- Certificate Generator, che viene utilizzato per la creazione di certificati e chiavi private per lo sviluppatore.
- Emulator, che permette di virtualizzare dispositivi mobile su cui poter testare le applicazioni.
- Smart Development Bridge, che permette di virtualizzare più dispositivi mobile ed instaurare connessioni tra i vari dispositivi.
- Tool-chain, che permette di compilare ed effettuare operazioni di debugging.
- UI Customizer, che permette di personalizzare i temi dell'interfaccia di controllo, che verrano utilizzati sui dispositivi con piattaforma Tizen.

All'interno dell'SDK ci sono tools specifici per la creazione di applicazioni native o per la creazione di applicazioni che utilizzano tecnologie Web.

3.3.2 Tools per applicazioni native

Per le applicazioni native si possono utilizzare i seguenti tools.

- Native IDE, un ambiente di sviluppo per applicazioni native Tizen che mette a disposizione un set di strumenti per aiutare chi sviluppa queste applicazioni.
- Command Line Interface, che permette di creare applicazioni senza il supporto di alcun software di sviluppo, tramite l'inserimento di istruzione via linea di comando.
- Dynamic Analyzer, che permette di monitorare le performance di un'applicazione mandata in esecuzione sul simulatore.
- UI Builder, che aiuta lo sviluppatore nella creazione delle interfacce utilizzando un approccio basato su un ambiente WYSIWYG.
- UI Effect Builder, che permette di creare effetti grafici come le animazioni.

Tra questi tools, il Native IDE è quello più articolato, che incorpora una lunga serie di tools pensati per lo sviluppo di applicazioni native.

3.3.2.1 Native IDE

Questo tool incorpora una serie di importanti tools.

- **API and Privilege Checker** - questo strumento controlla durante il processo di building, se ci sono problemi nel codice sorgente legati all'utilizzo di API ed i relativi privilegi.
- **Application Settings Editor** - questo strumento si occupa di memorizzare le impostazioni legate all'applicazione che si sta sviluppando. Le impostazioni vengono memorizzate su un file XML,

che contiene dati riguardanti la struttura e lo stile dell'applicazione. Questi dati verranno utilizzati per la creazione delle UI control dell'applicazione.

- **Call Stack View** - questo strumento è molto utile durante l'esecuzione dell'applicazione, perché permette di tener traccia di informazioni derivanti da un crash dell'applicazione. Queste informazioni sono riassunte in tre categorie, l'Header contenente le informazioni base sul crash, il Call Stack Information che tiene traccia di tutti i passaggi che hanno portato al crash e il Debug Message che contiene l'ultimo messaggio di debug che è stato lanciato.

- **Content Assist** - questo componente fornisce informazioni sulle API per supportare lo sviluppatore nel loro utilizzo. Queste informazioni che vengono offerte come supporto riguardano metodi, macro e parole chiave relative all'API che si intende usare.

- **Manifest Text Editor** - questo strumento aggevola la gestione del file manifest.xml.

- **OProfile** - strumento utilizzato per testare le performance dell'applicazione, che produce importanti report che aiutano lo sviluppatore nel processo di miglioramento dell'applicazione.

- **Project Wizard** - strumento che aiuta lo sviluppatore nel setting di tutti i paramentri che devono essere impostati durante la creazione di un nuovo progetto. Fornisce anche a fornire una serie di template, utili allo sviluppo di applicazioni con determinati stili architetturali.

- **Smart Launch** - questo strumento permette di mandare velocemente in esecuzione un progetto.

- **Snippets** - questo strumento crea una gerarchia tra i vari file dell'applicazione per supportare lo sviluppatore nella gestione dei vari contenuti e velocizzare il reperimento e l'accesso ai file dell'applicazione.

- **Unit Test Tool** - questo strumento supporta la creazione, la modifica e l'esecuzione dei progetti utilizzati dallo sviluppatore.
- **Valgrind** - questo strumento permette di controllare che vi sia una buona gestione della memoria, così da permettere di individuare errori e memory leaks presenti nell'applicazione. Gli strumenti utilizzati per eseguire questi controlli sono il Memcheck e il Massif.

3.3.3 Tools per applicazioni basate su tecnologie Web

Per le applicazioni basate su tecnologie Web vengono messi a nostra disposizioni i seguenti tools.

- Web IDE, che è un ambiente di sviluppo per applicazioni basate su tecnologie Web di Tizen, che mette a disposizione un set di strumenti per aiutare chi sviluppa queste applicazioni.
- Command Line Interface, che permette di creare applicazioni senza il supporto di software di sviluppo, tramite l'inserimento di istruzione via linea di comando.
- UI Builder, che aiuta lo sviluppatore nella creazione delle interfacce utilizzando un approccio basato su un ambiente WYSIWYG.
- Web Simulator, un simulatore che permette di testare le applicazioni create.

Tra questi tools il Web IDE è quello più articolato, che incorpora una lunga serie di tools pensati per lo sviluppo di applicazioni Web.

3.3.3.1 Web IDE

Questo tool incorpora una serie di importanti tools.

- **Advanced Declaration View** - questo strumento permette di visualizzare il codice sorgente Javascprit relativo ad una funzione, che viene chiamata in un altro file, perettendo allo sviluppatore un veloce reperimento di tutte le informazioni, senza dover cambiare il file visualizzato.

- **Code Beautifier** - strumento che permette di abbellire il codice, senza modificarne la struttura.
- **Code Minifier** - questo strumento permette di minimizzare il codice dell'applicazione andando a rimuovere quelle parti non necessarie come i commenti e comprimendo le linee di codice.
- **Configuration Editor** - questo strumento permette di gestire la struttara gerarchica degli elementi XML presenti nel file config.xml.
- **Content Assist** - questo strumento aiuta lo sviluppatore mostrando la documentazione relativa alle API utilizzate nell'applicazione e velocizza la loro scrittura tramite la possibilità di completare automaticamente l'istruzione.
- **CSS Editor** - questo strumento aiuta lo sviluppatore nella creazione dei file CSS.
- **CSS Preview** - questo strumento permette di avere un'anteprima dello stile CSS che verrà applicato.
- **HTML Editor** - questo strumento aiuta lo sviluppatore nella creazione del codice HTML.
- **HTML Preview** - questo strumento permette di visualizzare i file HTML prodotti basata su un browser Google Chrome.
- **JavaScript Editor** - questo strumento aiuta lo sviluppatore nella creazione di file Javascript controllando la correttezza del codice Javascript inserito e visualizzando eventuali errori.
- **JavaScript Log Console View** - questo strumento permette di visualizzare i log dei file Javascript quando viene mandato in esecuzione un widget.
- **Privilege Checker** - questo strumento controlla se siano presenti dei problemi nel codice legati all'utilizzo errato dei privilegi.
- **Remote Inspector** - questo strumento permette di operare dei debug su applicazioni Web remote utilizzando il JavaScript Debugger, che è basato sul Webkit Web Inspector.

3.4 Applicazioni native

Tizen mette a disposizione il Native framework per sviluppare applicazioni native al fine di aumentare le funzionalità della piattaforma Tizen. Nel Native framework sono presenti tutti i componenti necessari per sfruttare le capacità del dispositivo mobile, tramite l'utilizzo delle API messe a disposizione dal sistema. Di seguito andremo ad analizzare i vari aspetti che caratterizzano le applicazioni native di Tizen.

3.4.1 Tipologie di applicazioni native

Le applicazioni native di Tizen si suddividono in due tipologie, che permettono allo sviluppatore di creare applicazioni che ricoprono ruoli più specializzati all'interno del sistema. Si possono creare applicazioni UI o applicazioni Service.

3.4.1.1 Applicazioni UI

Queste applicazioni sono caratterizzate da un'interfaccia grafica adibita all'iterazione con l'utente. Sul dispositivo possono trovarsi in esecuzione più applicazioni di questo tipo, ma solo l'interfaccia di una di queste applicazioni verrà mostrata all'utente, che si troverà ad essere in uno stato di foreground, mentre le atre saranno nascoste continuando ad essere in esecuzione in uno stato di background. Il cambio di stato di queste applicazioni avviene ogni qual volta venga richiesto dell'utente o al verificarsi di determinati eventi. Per esempio l'utente può cambiare l'applicazione visualizzata in primo piano mandando in esecuzione l'applicazione o selezionandola dal task switcher. Il task switcher è un componente che permette all'utente di visualizzare tutte le applicazioni UI in esecuzione e di terminarne l'esecuzione.

3.4.1.2 Applicazioni Service

Queste applicazioni native non utilizzano alcuna interfaccia grafica e vengono eseguite solo in modalità background. Il loro compito principale è quello di svolgere in maniera performante tutte quelle attività che non richiedono l'iterazione con l'utente e che devono essere eseguite periodicamente oppure in continuazione. Queste applicazioni possono essere mandate in esecuzione anche da applicazioni UI, che richiedo il loro intervento in maniera esplicita.

Un esempio di applicazioni Service sono quelle che permettono di ottenere informazioni dai sensori presenti sul dispositivo. Come per le applicazioni UI, anche per queste applicazioni si può utilizzare il task switcher per controllare quante e quali applicazioni Service siano in esecuzione sul dispositivo mobile.

3.4.2 Ciclo di vita di un'applicazione

Il ciclo di vita di un'applicazione Tizen è mostrato nella figura 3.2.

La differenza tra ciclo di vita di un'applicazione UI e quello di un'applicazione Service risiede nel fatto che, ogni volta che viene mandata in esecuzione un'applicazione UI, viene creato un frame invocando il metodo AddFrame(). Successivamente l'applicazione UI seguirà il ciclo di vita specifico dei frame, come mostrato in figura 3.3.

Come si può notare in figura, dopo che un'applicazione viene inizializzata, si trova in uno stato di esecuzione. Nel caso di applicazioni Service, l'applicazione permane nello stato di esecuzione fino alla sua terminazione. Nel caso di applicazioni UI, subito dopo l'esecuzione il frame passa allo stato Actived e prima di terminare l'applicazione, il frame potrà transitare tra gi stati Actived, Deactived e Minimized.

3.4.3 Il Multitasking

Tizen supporta il multitasking per mandare in esecuzione più applicazioni UI e più applicazioni Service contemporaneamente. Questa capacità comporta un utilizzo maggiore della memoria del dispositivo, che come sappiamo, essendo una risorsa limitata, può portare a delle situazioni critiche dovute all'esaurimento di questa risorsa. Per gestire questa problematica, Tizen adotta una politica di gestione della memoria che prevede l'interruzione di alcune applicazioni in esecuzione, al fine di liberare lo spazio richiesto per il corretto funzionamento della piattaforma. La scelta delle applicazioni da terminare avviene tramite la valutazione del relativo livello di priorità associato all'applicazione e nel caso in cui più applicazioni hanno livelli di priorità uguali, il sistema andrà a terminare prima le applicazioni che sono rimaste in uno stato di background per maggior tempo.

I livelli di priorità associati alle applicazioni vengono utilizzati per determinare in quale ordine verranno terminate.

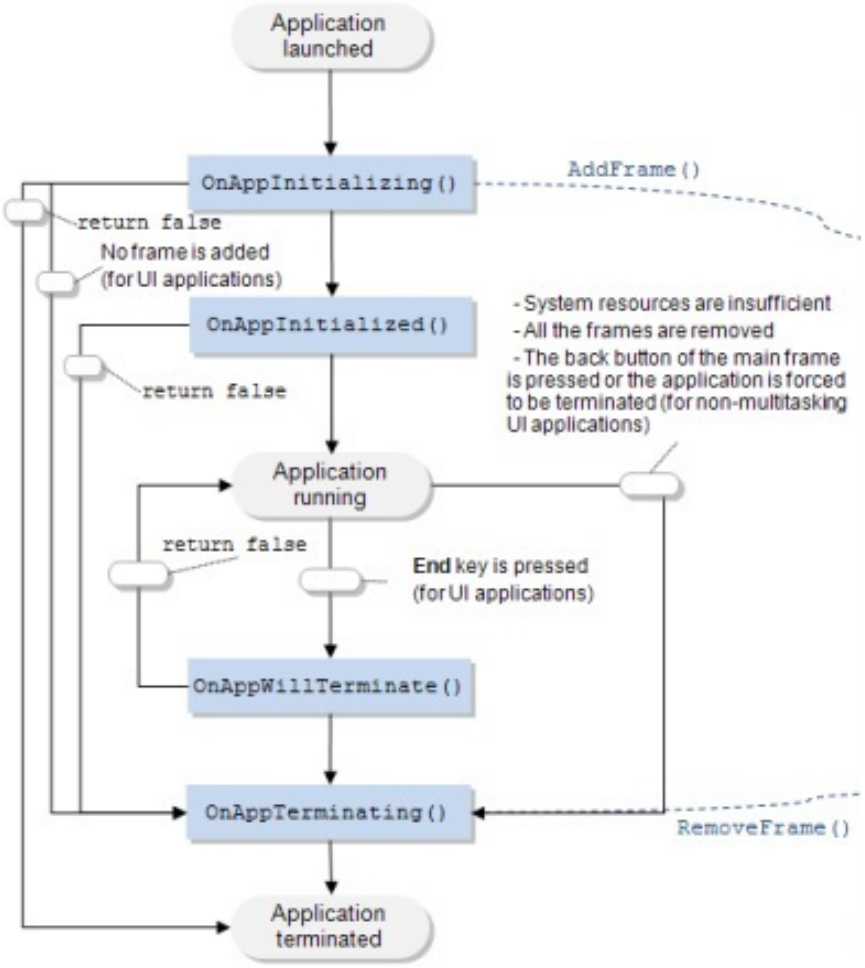

Figura 3.2: Ciclo di vita di un'applicazione Tizen, prima parte [10].

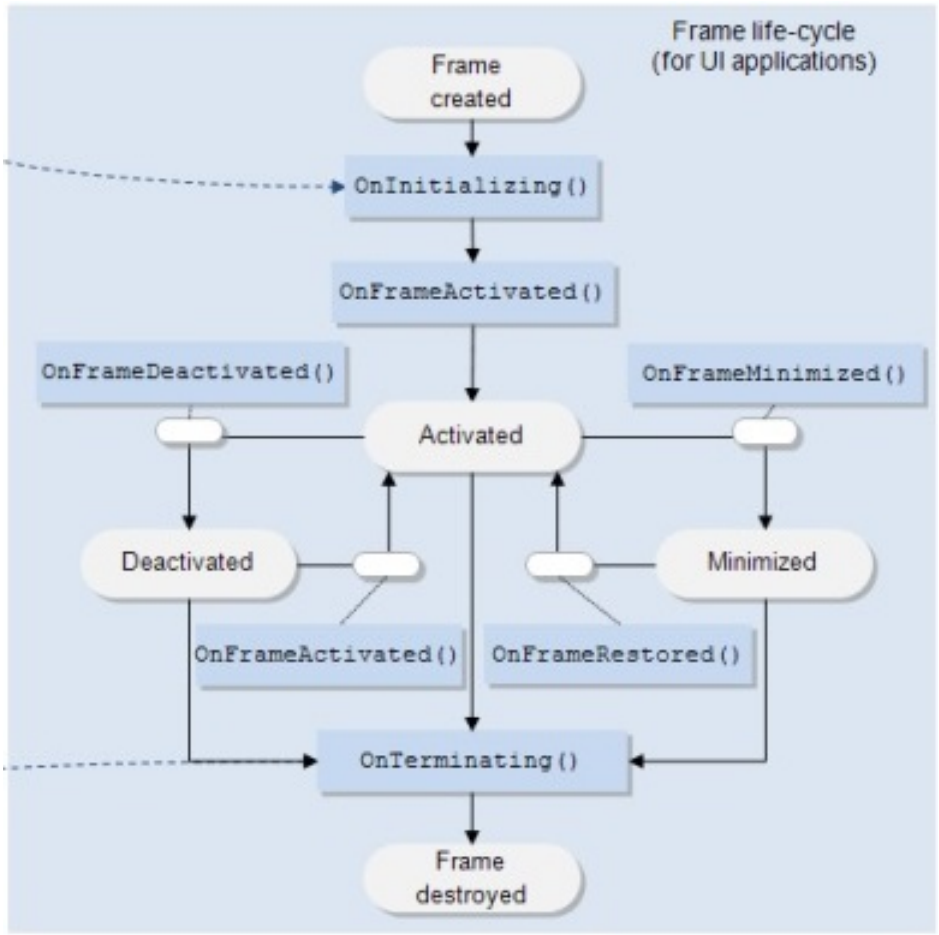

Figura 3.3: Ciclo di vita di un'applicazione Tizen, seconda parte [10].

- Applicazioni UI in background.
- Applicazioni Service.
- Media players in background.
- Applicazioni UI in foreground.
- Media players in foreground.
- Launcher delle applicazioni.
- Applicazioni di sistema, chiamate o videochiamate.

3.4.4 Multithreading

Le applicazioni Tizen possono utilizzare più thread durante la loro esecuzione. Quando un'applicazione viene mandata in esecuzione viene creato un thread principale chiamato main o UI thread, che ha il compito di indirizzare gli eventi all'interfaccia dell'applicazione e gestiste gli eventi provenienti da chiamate di sistema. Se si intendono utilizzare più thread per un'applicazione, tutti i thread a parte quello principale dovranno essere implementati nel codice dell'applicazione e possono essere di due tipologie event-driven o worker come mostrato in figura 3.4. Questo aspetto è molto importate durante lo sviluppo di un'applicazione, perché per ottimizzare un'applicazione occorre che tutte le operazioni, che richiedono una complessa elaborazione vengano eseguite su thread alternativi a quello principale, per evitare che il thread principale si ritrovi bloccato ad eseguire le operazioni richieste. Tutto ciò è dovuto al fatto che il thread principale si occupa dell'iterazione con l'utente e non ci si può permettere che resti bloccato.

I thread possono ritrovarsi a dover accedere alle stesse risorse condivise, per questo ci si può servire di meccanismi di sincronizzazione come mutex, semafori o monitor.

È possibile instaurare comunicazioni tra i diversi thread di un processo secondo diverse modalità. Si possono creare istanze della classe Event, si possono utilizzare gli event-driven threads o si può associare ai thread, che voglio comunicare fra loro, un oggetto della classe Monitor.

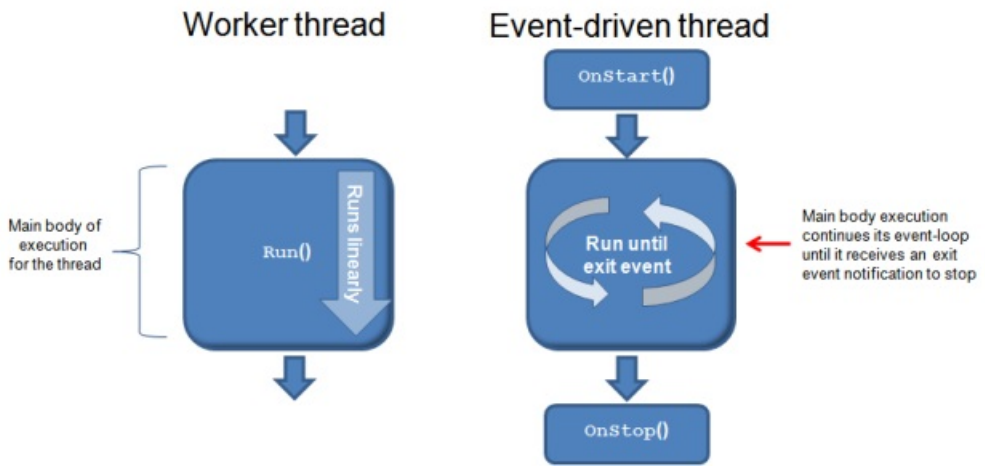

Figura 3.4: Threads di un'applicazione Tizen [10].

3.4.5 Two-phase Construction

Per istanziare oggeti nelle applicazioni native di Tizen, non viene utilizzato il classico costruttore del C++, ma viene utilizzato un costruttore che lavora in due fasi chiamato Two-phase Construction. Questo costruttore è stato così implementato per gestire le eccezioni che possono verificarsi durante la creazione di un oggetto. Ogni volta che si istanzia un oggetto di una determinata classe bisogna invocare il metodo Construct(), che permetterà di verificare se sono state generate eccezioni tramite un parametro di ritorno, cosicché si potranno gestire.

L'utilizzo di questo costruttore è mostrato qui in seguito.

```
String fileName("test.txt");
File file;

// Second-phase of construction must be called once
// right after instantiation
result r = file.Construct(App::GetInstance()->GetAppDataPath()
                          + fileName, "w+");

if (IsFailed(r))
{
```

```
   // Error handling
}
```

3.4.6 Politica di Object Ownership

Tutti gli oggetti creati da un'applicazione Tizen vengono memorizzati in memoria, perciò è importante gestire la rimozione degli oggetti quando non sono più necessari, al fine di evitare la presenza di memory leaks. Questa politica prevede che il proprietario di un determinato oggetto liberi la porzione di memoria occupata dall'oggetto, quando questo non sarà più utilizzato. Questa politica di Object Ownership di Tizen impone che la proprietà di un oggetto sia esclusiva, quindi non permette di avere più operatori in possesso di un singolo oggetto, ma un operatore potrà comunque cedere la proprietà di un oggetto ad un altro operatore. È importante che lo sviluppatore segua questa politica, durante lo sviluppo di un'applicazione, come mostrato nel seguente esempio di codice.

```
void MyUiApp::OnUserEventReceivedN(RequestId requestId, IList* pArgs)
{
  // pArgs ha assunto la proprieta' dell'oggetto
  // passato come argomento.

  // ... Altro codice della funzione ...

  // Liberare la memoria allocata dall'oggetto.
  pArgs->removeAll();
  delete pArgs;
}
```

3.4.7 Gestione delle eccezioni

In Tizen gli errori vengono gestiti diversamente dal C++, infatti non verrano lanciate delle eccezioni, ma verranno generati degli errori che saranno ritornati dalla funzione e inseriti in una varibile r di tipo result. Se non avvengono errori le durante l'esecuzione di una funzione viene ritornato un result contenente il valore E_SUCCESS, in caso contrario si possono avere diversi valori di ritorno. Per la gestione di questi errori vengono messi a disposizione diversi metodi.

- GetLastResult(), per recuperare l'ultimo valore presente nel result ritornato da un determinato thread.
- SetLastResult(), per impostare il valore di result.
- GetErrorMessage(), che ritorna un char* contenente il messaggio di result.

Un esempio dell'utilizzo di questi metodi è il seguente.

```
pObj = list.GetAt(...);

// Si puo' utilizzare anche if (pObj == null)
if (GetLastResult() != E_SUCCESS)
{
   // Error handling
}
```

3.4.8 Limitazioni della libreria C

Alcune delle funzioni provviste dalla libreria C utilizzate nelle API native di Tizen hanno problemi di compatibilità con il sistema. Tra queste funzioni troviamo system(), pipe() e tutta la famiglia delle funzioni exec(), perciò la possibilità di un loro utilizzo dipende dalla versione della piattaforma e dal dispositivo mobile che utilizza una determinata versione di Tizen. Queste operazioni possono essere eseguite sia da comandi creati dall'utente che da comandi di sistema, ma per le già citate problematiche si consiglia di utilizzare i metodi system(), pipe() e gli eventuali exec() solo nei comandi forniti dall'utente.

3.5 Applicazioni basate su tecnologie Web

Per creare applicazioni basate su tecnologie Web possiamo sfruttare il Web-Framework offerto da Tizen, che permette di creare applicazioni che utilizzano le più aggiornate tecnologie HTML, JavaScript e CSS. Tramite questo framework possiamo creare, oltre le classiche applicazioni Web, applicazioni basate su tecnologie Web chiamate Widget. I widget sono applicazioni Web standalone, che utilizzano queste tecnologie e possono essere mandate in

esecuzione sul dispositivo mobile, senza necessitare del supporto di risorse esterne o connettività di rete. I Widget hanno un modello che supporta molte funzionalità standard W3C/HTML5 integrando API JavaScript, oltre ad ulteriori markup HTML e funzioni di CSS.

Per lo sviluppo di queste applicazioni, Tizen offre importanti componenti che andremo ad analizzare in seguito, al fine di sviscerare tutti gli aspetti legati alla gestione e alle potenzialità di queste particolari applicazioni.

3.5.1 Web Runtime

Tizen utilizza un motore chiamato Web Runtime, per gestire molteplici aspetti legati alle applicazioni che utilizzano tecnologie Web. Il Web Runtime ricopre inoltre un ruolo molto importante, perché permette alle applicazioni Web di essere eseguite al di fuori del browser installandole su dispositivi mobile, per mandarle in esecuzione come fossero applicazioni standalone. Il Web Runtime supporta due tipi di applicazioni web.

- Applicazioni Web Packpaged.
- Applicazioni Web Hosted.

Tutte le applicazioni che utilizzano tecnologie Web devono creare il proprio packpage seguendo le linee guida dell'XML Configuration e del W3C Widget Packaging. A differenza delle applicazioni Web Packpaged, quelle Hosted hanno un documento esterno come pagina principale.

Il Web Runtime e il componente principale nella gestione di tutte le applicazioni basate su tecnologie Web, che possono essere sviluppate attraverso Tizen e permette di effettuare diverse operazioni per la loro gestione.

3.5.1.1 Operazioni

Il Web Runtime, come ogni gestore, utilizza le tipiche operazioni di gestione come l'installazione, l'aggiornamento e la disinstallazione.

Installazione

Il Web Runtime installa l'applicazione e una volta completata manda

una notifica, successivamente aggiunge sullo scermo del dispositivo un'icona con il nome dell'applicazione Web, sempre seguendo le specifiche del Web widget packpaging. In caso di mancato completamento dell'installazione, come ad esempio se si spegne il dispositivo per esaurimento della batteria, il Web Runtime reinstallerà l'applicazione al prossimo avvio del dispositivo.

L'installazione viene interrotta dal Web Runtime nei seguenti casi.

- La versione delle Web API di Tizen utilizzate dal Web Runtime è antecedente a quella richiesta dall'applicazione Web.
- L'applicazione Web ha un livello di privilegi Public e una o più API utilizzate hanno privilegi Platform o Partner descritti nel file di configurazione.
- Le applicazioni Web possono essere installate sia dal Tizen Store, che dal Browser o via Bluetooth.

Aggiornamento

Il Web Runtime permette l'aggiornamento della applicazioni quando viene rilasciata una nuova versione, sempre che la nuova versione segua le seguenti regole.

- Se l'applicazione ha un Tizen AppID, la nuova versione deve avere lo stesso Tizen AppID della precedente.
- Se l'applicazione ha una firma digitale dell'autore, la nuova versione deve avere lo stessa firma digitale dell'autore.
- Se l'applicazione non ha la firma digitale dell'autore, la nuova versione non deve averne.

Questo processo di aggiornamento è simile a quello dell'installazione, tutti i vecchi dati prodotti dalla vecchia versione dell'applicazione vengono conservati e bisogna assicurarsi che siano riutilizzabili dopo l'aggiornamento.

Disinstallazione

Il Web Runtime quando cancella l'applicazione rimuove tutti i dati associati ad essa.

3.5.1.2 API di supporto

Il Web runtime ha principalmente il compito di fornire alle applicazioni Web le seguenti API.

- W3C/HTML5 API.
- Tizen Device API, solo per le applicazioni packpaged.

Le API utilizzate dal Web Runtime permettono ad un'applicazione di creare più pagine e di utilizzare tecniche di hyperlink per la navigazione. Tutte la Web API di Tizen possono essere utilizzate sia dalla pagina corrente che da quelle innestate come gli iFrame.

3.5.1.3 Sicurezza e Privacy

Il Web Runtime offre meccanismi di protezione tramite:

- Widget Signature, le applicazioni vengo certificate tramite firma digitale d'autore e dal distributore, la firma del distributiore ne determina livelli di privilegi tra Public, Partner o Platform.
- Viene offerta una protezione ulteriore criptando i file HTML, CSS e Javascript presenti sul dispositivo, che solo in fase di esecuzione verranno decriptati dal Web Runtime, così da garantire all'applicazione la trasparenza di questa operazione.
- Ogni applicazione ha uno spazio privato in memoria per contenere i propri dati.

Il sistema del livelo di privelegi associato alle applicazioni garantisce una protezione nell'utilizzo delle HTML5 API e delle Device API più critiche, che potrebbero causare malfunzionamenti nel sistema.

Molto importante è la politica di sicurezza dei contenuti, perché questa politica permette di informare l'utente da dove l'applicazione intende caricare le risorse utilizzate, al fine di dare la possibilità di individuare e bloccare le risorse sospette. In più viene data la possibilità agli sviluppatori di impostare dove l'applicazione deve andare a cercare le proprie risorse. Questa politica di sicurezza è necessaria al fine di limitare i tentativi di cross-site scripting, che rigurdano quegli utenti malintenzionati che vogliono iniettare script dannosi all'interno dell'applicazione.

3.5.2 Caratteristiche di sviluppo

Tizen attraverso i tools di sviluppo offerti permette di creare applicazioni Web e Widget che sfruttano i framework e le relative API, per dare la possibilità agli sviluppatori di creare applicazioni con una vasta gamma di funzionalità. In questa sezione ci dedicheremo maggiormente ai Widget, che riguardano il nostro caso di studio, quindi andremo ad analizzare gli aspetti da prendere in considerazione per lo sviluppo di un'applicazione Widget.

3.5.2.1 Impostazioni di configurazione

I Widget hanno un file contenente le impostazioni di configurazione chiamato config.xml, che al suo interno contiene svariate impostazioni, che possiamo catalogare qui in seguito. Per impostare queste configurazioni, in fase di sviluppo ci vengono incontro i tools offerti da Tizen, che suddividono queste categorie di aspetti in vari tab contenenti tutte le impostazioni da editare.

Overview

In questa sezione verranno contenute tutte le informazioni di carattere generale come:

- Identificatore.
- Versione.
- Nome del Widget.
- Content, i file di esecuzione del Widget.
- Icona del Widget.

Widget

Contiene le informazioni riguardanti la licenza e le impostazioni dell'interfaccia grafica. Per quanto riguarda la licenza abbiamo la possibilità di impostare:

- Autore.

- E-mail.
- Sito Web.
- Licenza, che contiene informazioni riguardanti contratto, copyright, ecc ...
- Descrizione.

Per quanto riguarda l'interfaccia grafica:

- Larghezza.
- Altezza.
- Modalità di visualizzazione.

Features

Qui possono essere aggiunti tutti i filtri software e hardware, che si vogliono impostare per l'applicazione. Un esempio di come verrrano inseriti questi filtri nel file config.xml è il seguente.

```
<tizen:feature name="http://tizen.org/feature/network.nfc/">
```

Privileges

Qui si possono inserire API o i gruppi di API, che verranno utilizzate nell'applicazione e perciò occorre avere privilegi adeguati per il loro utilizzo. Si possono aggiungere privilegi in diverse modalità:

- Internal, che renderà possibile sceglire da una lista, i privilegi di quali API aggiungere.
- Privilege name, i privilegi verranno letti da un indirizzo Web, che contiene le definizioni.
- File, i privilegi sono contenuti su di un file .xml o .widlprocxml.

Policy

In accordo con il W3C Access Requests Policy, non è possibile utilizzare di default le risorse di rete, perciò se l'applicazione necessita di queste risorse, bisogna richiedere le autorizzazioni per l'utilizzo di queste risorse. In questa sezione è possibile impostare le seguenti proprietà:

- content-security-policy, per impostare ulteriori politiche di sicurezza sia per applicazioni hosted che per applicazioni packpaged.
- content-security-policy-report-only, per impostare ulteriori politiche di sicurezza sia per applicazioni hosted che per applicazioni packpaged, con la possibilità di produrre report.
- allow-navigation, che definisce una lista di domini accessibili per applicazioni Web.
- access, per permettere all'applicazione di accedere a specifici siti internet.

Localization

In questa sezione lo svliuppatore piò offrire supporto all'applicazione, per localizzare più agevolmente nel file config.xml informazioni come:

- Nome, informazioni riguardanti il nome dello sviluppatore.
- Descrizione, descrizione del Widget.
- Licenza, informazioni riguardanti nome ed indirizzo Web della licenza.

Preferences

In questa sezione è possibile impostare una serie di coppie nome-valore, che possono essere richimate tramite l'uso di Widget API, per essere utilizzate nell'applicazione. È possibile anche impostare se queste coppie sono utilizzabili solo in lettura oppure se è possibile modificarle. Di fatto queste coppie possono essere viste come variabili globali.

Tizen

In questa sezione è possibile impostare tutte le informazioni generali riguardanti vari aspetti dell'applicazione sviluppata tramite Tizen, alcune di queste informazioni è obbligatorio che siano impostate, mentre altre sono opzionali e possono avere un valore di default.

- Applicazione, con informazioni riguardanti ID, Packpage, Versione Richiesta.
- Contenuti, dove viene indicata la cartella o il server dove si trova l'applicazione.
- Impostazioni, dove si impostano caratteristiche quali screen-orientation, context-menu, background-support, encryption, install-location, hwkey-event.
- App-control, che descrive le funzionalità dell'applicazione.
- App-widget, con attributi quali id, primary, auto-launch, update-period, box-label, box-icon, box-content, box-size, pd.
- Account, per informazioni riguardanti Display name, Multiple account, Icon, Small icon, Capabilities.
- Metadata, che contiene metadati chiamati key, che possono essere accedudi solo in lettura dalle API usate dalle applicazioni Tizen.

Source

In questa sezione viene mostrato il contenuto attuale del file config.xml, che può essere anche direttamente modificato. Le modifiche che si attuano nel file config.xml otterranno un riscontro nelle sezioni relative.

3.5.2.2 Supporto ai display

Tizen offre vari strumenti per supportare la scalabilità delle applicazioni su dispositivi mobile aventi display di dimensioni e risoluzioni differenti. Questo aspetto è fondamentale per rendere le interfaccie grafiche delle applicazioni, compatibili con la maggior parte dei dispositivi mobile. Per garantire questo supporto vengono introdotti i seguenti aspetti.

Standard W3C

Gli standard offerti da W3C per supportare la scalabilità delle applicazioni sono rappresentati da due componenti adibiti a soddisfare questa necessità.

- **Viewport Meta Tag** - definisce la porzione di area relativa all'applicazione dove verranno mostrati i contenuti della stessa. Tramite questo componente possiamo impostare le dimensioni dell'area in altezza e larghezza, in più ci permette di utilizzare fattori di scala. Bisogna tener presente che gli attributi altezza e larghezza, non corrispondono esattamente a quelli reali del dispositivo, ma utilizzando il fattore di scala si può ovviare a questo problema.

  In ogni dispositivo il fattore di scala è calcolato utilizzando il DPR, Device Pixel Ratio.

- **CSS Media Query** - tramite questo componente è possibile impostare condizioni per l'utilizzo di diversi caratteristiche multimediali di varie tipologie ed applicare specifici file CSS in base alle necessità. Questo componente permette di utilizzare importanti funzionalità per definire quale immagine utilizzare in base al DPI dello schermo, oppure definire quale file CSS utilizzare in base alle dimensioni dello schermo. Questo componente è utilizzabile sia nei file CSS che nei file HTML.

Ottimizzazione delle risorse

L'utilizzo di applicazioni basate su tecnologie Web risulta efficiente sui diversi sistemi operativi per dispositivi mobile, poiché queste applicazioni utilizzando componenti come il W3C Media Queries e il Flexible Box Layout, che sono in grado di creare layout che si adattano bene alle diverse condizioni in cui si trovano. Questo facilita la produzione di packpage che possono essere distribuiti ed utilizzati su dispositivi mobile aventi differenti caratteristiche fisiche.

Risorse come immagini animate, foto ed icone possono essere utilizzate sia in file HTML o come proprietà di file CSS. Queste risorse che spesso verranno utilizzate nello sviluppo di applicazioni vengono ottimizzate graficamente attraverso SVG API, Canvas API e WebGL API.

3.5.3 Elementi di programmazione

Durante lo sviluppo di applicazioni Web o applicazioni basate su tecnologie Web, Tizen offre una vasta gamma di API che possono essere utilizzate per sviluppare l'applicazione. Come vedremo in seguito non tutte le API utilizzabili provengono dal WebFramework di Tizen, infatti Tizen permette di incorporare nelle applicazioni che utilizzano tecnologie Web, anche altri componenti compatibili con queste tecnologie, infatti come si vedrà in seguito alcune API utilizzate nelle applicazioni native di Tizen possono anche essere utilizzate nelle applicazion basate su tecnologie Web.

3.5.3.1 Device API

Queste API integrano importanti funzionalità presenti nella programmazione nativa per applicazioni basate su tecnologie Web, così da aumentarne le potenzialità. In questa collezione di API troviamo API di svariate tipologie, che vengono utilizzate dalla stessa piattaforma Tizen, elencate qui di seguito.

Tizen API

Le Tizen API si occupano di filtri, modalità di ordinamento, metodi per la gestione di eventi, interfacce per la gestione degli errori e per la visualizzazione delle informazioni relative alla posizione. Per quanto riguarda i filtri che possono essere utilizzati, ne esistono di varie tipologie tra cui filtri che si occupano di determinati attributi, filtri che controllano attributi entro un certo range e filtri formati da un set di altri filtri. Le modalità di ordinamento offerte sono le classiche, decrescente e crescente. Le interfacce per la gestione degli errori possono essere sincronizzate con l'interfaccia WebAPIException, oppure in modo asincrono con l'interfaccia WebAPIError.

Application API

Le Application API contengono API riguradanti la gestione delle applicazioni, dei packpage e dei timer, chiamati Alarm. Gli alarm hanno interessanti funzionalità, poiché permettono di mandare in esecuzione applicazioni e ricevere informazioni riguardanti la generazione di eventi futuri. Le API

riguardanti la gestione delle applicazioni permetto di ricavare informazioni relative ad applicazioni installate o in esecuzione, così da permettere anche di ricevere notifiche in caso di aggiornamento o rimozione. A livello più operativo queste API permettono di mandare in esecuzione le applicazioni o terminarne la loro esecuzione.

In questo set di API troviamo un meccanismo importante chiamato Application Control, che permette di utilizzare specifiche operazioni esportate da altre applicazioni come per esempio chimare, mandare SMS, riprodurre componenti multimediali, ecc ... Questo meccanismo permette anche di lanciare in esecuzione altre applicazioni che offrono funzionalità necessarie all'applicazione stessa, anche se non si conoscono gli identificativi o le specifiche, tramite la richiesta per l'avvio di queste applicazioni al sistema. Infatti l'Application Control piò mandare in esecuzione le applicazioni in modo esplicito fornendogli l'ID dell'applicazione da lanciare o in modo implicito descrivendo l'operazione che deve essere eseguita. L'Application Control cercherà di trovare l'applicazione che piò portare a termine il compito richiesto. Questo meccanismo richiede due informazioni, l'applicazione che necessita l'intervento dell'Application Control e l'operazione richiesta.

Comunication API

Le Comunication API sono API che consentono di avvalersi delle funzionalità relative all'utilizzo di servizi di Bluetooth, Messaging, NFC, Push e Secure Element. Per quanto riguarda il Bluetooth, le API permettono di gestire l'accensione e lo spegnimento di questa risorsa, di utilizzare funzionalità come trovare altri dispositivi con questa risorsa attiva, di stabilire una connessione tra i due dispositivi e scambiare dati. Il Messaging consente di gestire, editare, inviare e ricevere SMS, MMS e EMail. L'NFC è una tecnologia simile al Bluetooth ma con diversi vantaggi come un più veloce set-up, minori consumi di energia, non viene richiesto l'accoppiamento dei dispositivi e riduce le interruzioni indesiderate. Il Push è un meccanismo per ricevere notifiche da un determinato server, che può essere visto come un processo client deamon che mantiene una connessione permanente con il server. Se l'applicazione è connessa, il servizio Push passa i dati di notifica attraverso la connessione, altrimenti manda i dati di notifica all'interfaccia. Le Secure Element API permettono di accedere a risorse protette di un dispositivo come SIM Card e schede SD. Queste API permettono la gestione

degli elementi in sicurezza dando la possibilità di instaurare e chiudere canali di comunicazione, dove sarà possibile ricevere le informazioni e notifiche relative a queste risorse.

Content API

Le Content API forniscono funzionalità per ricercare e gestire i diversi contenuti presenti sul dispositivo. Quando si ricerca un contenuto si possono utilizzare anche filtri e modalità di ordinamento per i risultati di ricerca, come descritto nelle Tizen API.

Queste API permetto di ricevere notifiche di uno specifico contenuto, tramite l'utilizzo di un'interfaccia ContentChangeCallback, che implementa un ascoltatore adibito agli eventi generati.

In questo set di API sono presenti anche delle Download API, che permetto di gestire l'istallazione sul dispositivo di contenuti presenti sulla rete internet, tramite l'utilizzo dell'iterfaccia DownloadManager, che permette anche di avere informazioni riguardanti lo stato corrente dell'operazione di download.

I/O API

Le I/O API sono API che permetto di gestire i file e le directory nel filesystem del dispositivo, sempre che non siano protette e quindi inaccessibili. Queste parti del filesystem alle quali si può accedere vengono rappresentate come directory virtuali e si possono gestire sia i file che le directory, anche andando ad installare nuovi file al loro interno.

Inoltre in questo set di API risiedono le API per gestire il Message Port, per permettere di comunicare con le altre applicazioni native e Web, tramite l'utilizzo di messaggi in formato map data, che sono composti da una string key e una coppia di valori. Si possono utilizzare due tipologie di Message Port, le LocalMessagePort che consentono di registrare e preparare alla ricezione la MessagePort dell'applicazione, le RemoteMessagePort cosentono di trasmettere messaggi alle altre applicazioni.

Social API

Le Social API sono API che contengono le funzionalità base che i sistemi

operativi offrono agli utenti come segnalibri, calendario, ultime chiamate, contatti e la sincronizzazione dei dati. La sincronizzazione dei dati viene effettuta tramite il server OMA DS utilizzando il protocollo 1.2 .

System API

Le System API sono API di sistema che offrono funzionalità molto importanti per accedere alle informazioni relative alla piattaforma, lo stato della batteria e permettono di gestire tutti gli aspetti riguardanti l'orologio, gli sfondi, le suonerie, la cronologia internet, ecc ... Tra queste API sono molto interessanti le Power API, che permettono di gestire quegli aspetti del dispositivo che incidono sul consumo di batteria come la luminosità, l'accensione e lo spegnimento dello schermo. Tramite queste API si può richiedere e rilascire uno stato di risparmio energetico riguardante lo schermo o la CPU. Se viene richiesto di entrare in un nuovo power state, senza prima rilasciare il precedente power state verrà impostato come attuale stato energetico, quello a più basso consumo.

UI API

Queste API permettono di creare e gestire notifiche di eventi generati dall'interfaccia dell'applicazione. La creazione delle notifiche viene implementata secondo il seguente codice.

```
/* Application control */
var appControl = new tizen.ApplicationControl("
http://tizen.org/appcontrol/operation/create_content", null,
   "image/jpg", null, null);

var notificationAct =
{
   /* Notification content */
   content: "This is a simple notification.",
   /* Path to the notification icon */
   iconPath: "images/image1.jpg",
   /* Path to the sound file to be played when the notification
      is displayed */
```

```
   soundPath: "music/Over the horizon.mp3",
   /* Device vibrates when the notification is displayed */
   vibration: true,
   /* Application control to be launched when the user selects
      the notification */
   appControl: appControl
};
```

Per quanto riguarda la gestione delle notifiche, si può utilizzare il seguente codice.

```
// Recuperare l'ultima notifica
var myId = notification.id;
var myNotification = tizen.notification.get(myId);

// Recuperare tutte le notifiche
var notifications = tizen.notification.getAll();

// Aggiornare una precedente notifica
myNotification.content = "Aggiornata";
tizen.notification.update(myNotification);

// Rimuovere l'ultima notifica
tizen.notification.remove(myNotification.id);

// Rimuovere tutte le notifiche
tizen.notification.removeAll();
```

3.5.3.2 Widget User Interface

Le Widget UI sono dei controllori di interfacce come per esempio i bottoni, che possono interagire con l'utente ed essere gestiti secondo le necessità. Questi componenti utilizzano il Web framework per essere creati sulla base di template e gli si possono associare delle funzionalità generiche.

UI Service Template

Le Widget UI sono create utilizzando i template presenti nel Web UI che

sono supportati della tecnologia WebKit. Le funzionalità offerte dal Web UI possono essere utilizzate poiché viene importata la libreria del Tizen Web UI framework tramite l'inserimento di opportuni script nell'header della pagina HTML. Questi template implementano anche proprietà di scalabilità per adattarli meglio al display, che avrà il compito di visualizzarli. Quando viene importata la libreria vengono caricati dinamicamente anche i file CSS. Un esempio dell'utilizzo di questi UI Service Template è il seguente.

```
<head>
  <meta name="viewport" content="width=device-width, initial-
    scale=1">
  <title>Page Title</title>
  <script src="tizen-web-ui-fw/latest/js/jquery.min.js"></script>
  <script src="tizen-web-ui-fw/latest/js/tizen-web-ui-fw-libs.min
    .js"></script>
  <script src="tizen-web-ui-fw/latest/js/tizen-web-ui-fw.min.js"
    data-framework-theme="tizen-white"></script>
  <script src="myapp.js"></script>
  <link rel="stylesheet" href="my.css"/>
</head>
```

Funzionalità generiche

Il Tizen Web UI mette a disposizione importanti funzionalità da poter utilizzare all'interno delle Widget UI.

- **Accessibilità** - definita dallo standard WAI-ARIA che supporta il facile accesso ai contenuti Web delle applicazioni che utilizzano tecnologie Web. Il Tizen screen reader analizza i contenuti mostrati sullo schermo e li trasmette all'utente sotto forma di dati audio, tramite l'utilizzo della tecnologia text-to-speech e i dati WAI-ARIA. Il Tizen screen reader analizza i contenuti seguendo un determinato ordine, prima legge gli elementi HTML presenti nel file, successivamente vengono letti gli attributi lasciando alla fine l'analisi del'attributo role e dell'attributo aria-label.

- **Eventi** - tutti gli eventi supportati sono descritti nella documentazione jQueryMobile, che si occupa degli eventi riguardanti l'iterazione dell'utente con il dispositivo. Gli eventi supportati riguardano

diverse categorie come il tocco dello schermo, il mouse virtuale, il posizionamento del dispositivo, lo scrolling, la visualizzazione delle pagine e le animazioni CSS.

- **Localizzazione** - vengono supportati tutti i servizi legati alla localizzazione messi a disposizione dalla libreria Globalize, che sono utilizzabili tramite le API fornite dalla libreria.

- **Scrolling** - questa funionalità è la medesima utilizzata dai browser, che può essere abilitata inizializzando la proprietà overflow, che permette l'inserimento e la visualizzazione di una scroll bar associata all'interfaccia. Questa scroll bar può essere utilizzata per scorrere i contenuti visualizzati sul display. Gli attributi della scroll bar possono essere modificati nel file configure.js.

- **Utilità** - vengono supportati diversi metodi utilizzabili nelle applicazioni, come ad esempio abilitare e disabilitare la possibilità di selezionare i contenuti visualizzati, oltre ad abilitare e disabilitare il contex menù.

3.5.3.3 W3C/HTML5 API

Per lo sviluppo di applicazioni basate su tecnologie Web vengono messe a disposizione le più aggiornate API di queste nuove tecnologie, che forniscono un'ampia gamma di funzionalità utilizzabili nelle applicazioni. Queste funzionalità ricoprono svariati ambiti, che verranno analizzati qui in seguito.

DOM, Forms e stili

Queste API permettono di gestire gli aspetti relativi alla struttura DOM del file HTML, ai Forms per l'inserimento di dati da parte dell'utente e agli stili legati al documento. Per la gestione di questi aspetti vengono offerte le seguenti funzionalità.

- **CSS Animations Module Level 3** - per la creazione di animazioni utilizzando la proprietà animation definita dal CSS3. Queste animazioni vengono generate andando a cambiare nel tempo i parametri degli stili CSS. Le animazioni utilizzeranno dei keyframe

specifici per definire il valore degli attributi durante questo processo grafico. Quando si utilizzano le proprietà delle animazioni CSS occorre specificare per ogni keyframe il prefisso -webkit-, che è un requisito richiesto dal browser di Tizen, come mostrato nel seguente codice.

```
@-webkit-keyframes boxani
{
  from {left: 0; top: 0;}
  to {left: 100px; top: 100px;}
}
```

Le proprietà caratteristiche delle animazioni sono le seguenti.

- animation-name.
- animation-duration.
- animation-iteration-count.
- animation-timing-function.
- animation-direction.
- animation-play-state.
- animation-delay.
- animation-fill-mode.

- **CSS Backgrounds and Borders Module Level 3** - per specificare caratteristiche quali bordi e stili di backgroud utilizzate dagli elementi HTML definiti nelle proprietà del CSS. Le proprietà che si possono utilizzare sono le seguenti.
  - background-color.
  - background-image.
  - background-position.
  - background-repeat.
  - background-origin.
  - background-size.
  - background-clip.

- **CSS Basic User Interface Module Level 3** - permette di applicare le specifiche di un file CSS al documento HTML. Tra queste

specifiche troviamo le proprietà del Box model come Content, Padding, Border e Margin. Queste proprietà di stile vengono assegnate agli elementi del DOM.

- **CSS Color Module Level 3** - permette di inizializzare le caratteristiche di colore ed opacità degli elementi HTML utilizzando le proprietà definite dal CSS. La caratteristica del colore può essere definita attraverso diversi formati quali Keyword, RGB, RGBA, HSL e HSLA.

- **CSS Flexible Box Layout Module** - per gestire la flessibilità dei layout delle applicazioni. I componenti principali di un layout flessibile di una Flexible Box sono i Flex container e i Flex item.

- **CSS Multi-column Layout Module** - per la creazione di layout multi colonna. Possono essere gestite caratteristiche delle colonne quali numero, dimensioni, gap, rule, span e break.

- **CSS Transforms** - permette di modificare, di ruotare e di spostare gli elementi presenti nel documento utilizzando la proprietà transform definita dal CSS. Gli elementi possono essere gestiti sia in spazi 2D che in spazi 3D e le coordinate vengono espresse in pixel. Per migliorare le performance e il rendering delle immagini è possibile sfruttare l'accellerazione hardware. Si possono creare anche effetti di dissolvenza. Quando si utilizza questa proprietà occorre aggiungere un prefisso, -moz- se si utilizza un browser Firefox, -webkit- per browser Chrome e Safari e -o- per usarle su browser Opera.

- **CSS Transitions Module Level 3** - permette di aggiungere effetti grafici al cambiare delle proprietà di stile associate ad un elemento presente nel documento utilizzando la proprietà transition definita dal CSS. Per utilizzare questa proprietà sul browser di Tizen occorre aggiungere il prefisso -webkit-. Le transizioni sono caratterizzate da diverse proprietà.
  - transition-property.
  - transition-duration.
  - transition-timing-function.

  - transition-delay.

- **CSS3 Text Effects** - permette di cambiare il font dei testi e aggiungere diversi effetti. Il CSS3 introduce nuove caratteristiche riguardanti le proprietà dei Font, file in formato WOFF 1.0 e le proprietà Text.

- **HTML5 Forms** - permette di implementare agevolmente form Web utilizzando elementi dell'HTML5, per permettere l'utente di inserire dati. L'HTML5 introduce nuovi elementi quali il datalist, il keygen, il meter, l'output e il progress. Inoltre introduce una serie di nuove tipologie di elementi di input e nuovi attributi per gli elementi di input.

- **Media Queries** - permette di sfruttare funzionalità multimediali utilizzando nuove caratteristiche introdotte dal CSS3.

- **Selectors API Level 1 and Level 2** - permettono di selezionare uno o più specifici elementi presenti in un documento e confrontarli con stringhe o parametri, per velocizzare la ricerca e la selezione di elementi complessi. Queste ricerche andranno infatti ad operare lungo il DOM tree. Per ricercare gli elementi si possono utilizzare i metodi querySelector() e querySelectorAll(), che accettano come paramentro un string selector. Nel caso si volesse utilizzare un gruppo di parametri di ricerca, si potrà utilizzare come string selector il valore di ritorno del metodo querySelector(). Nel caso si ricerchino singoli elementi, si possono utilizzare gli string selector del CSS.

Dispositivo

Queste API permettono di analizzare aspetti riguardanti il dispositivo mobile offrendo importanti funzionalità per recuperare informazioni tecniche.

- **Battery Status API** - permettono di ottenere informazioni sullo stato della batteria e di scoprire se sono avvenuti dei cambiamenti. Queste API sono molto utili perché tramite il loro utilizzo si

può adattare il comportamento dell'applicazione in base al livello energetico del dispositivo. Funzionalità interessanti che possiamo implementare per ottimizzare le applicazioni sono differenti, infatti le applicazioni possono utilizzare queste API anche per fare un salvataggio dei dati ogni volta che il livello della batteria diventa critico, così da evitare la perdita dei dati da parte dell'utente nel caso il dispositivo si spenga. Oppure se si pensa ad un'applicazione che si occupa della gestione delle email, si può impostare la frequenza con cui controlla se sono state ricevute nuove mail, in base al livello energetico della batteria attuando così una politica di risparmio energetico. Un utilizzo corretto di queste API permette allo sviluppatore di creare applicazioni ottimizzate.

- **DeviceOrientation Event Specification** - permette di rilevare rotazioni ed accellerazioni effettuate dal dispositivo. Rilevando i dati relativi alle rotazioni effettuate dal dispositivo, si possono creare applicazioni che ruotano elementi e personaggi nel caso dei videogiochi, in base alla rotazione del dispositivo mobile. Tramite la rilevazione delle accellerazioni del dispositivo è possibile reperire informazioni sulla velocità di rotazione, sulla gravità tramite l'evento devicemotion definito dalla tecnologia HTML5. Anche questi dati possono essere utilizzati per implementare le animazioni legate ad elementi e personaggi di videogiochi.

- **HTML5 Browser State** - permette di ottenere informazioni sullo stato di connessione del browser. Queste informazioni possono essere reperite utilizzando l'attributo booleano navigator.onLine. Tramite l'utilizzo di oggetti di classe Window o WorkerGlobalScope possiamo monitorare i cambiamenti dello stato della connessione, attraverso la generazione di eventi quali online e offline.

- **Network Information API** - permettono di ottenere informazioni riguardanti la velocità di connessione del dispositivo. Per ottenere queste informazioni basta consultare l'interfaccia Connection contenente due importanti attributi, bandwidth contenente lo stato e le informazioni sulla velocità di connessione e metered contenente le informazioni sulle limitazioni relative all'internet

service provider. Si possono utilizzare i seguenti attributi come mostrato nel seguente codice.

```
// Attributo bandwidth
connection = navigator.webkitConnection;
myNet = ""+connection.bandwidth;

// Attributo metered
limit = navigator.webkitConnection.metered;
```

- **Screen Orientation API** - permettono di ottenere informazioni sullo stato di orientazione del dispositivo e di bloccare lo stato dello schermo in una speciafica orientazione. Si possono sfruttare varie funzionalità utilizzando queste API, come mostrato nel seguente codice.

```
// Informazioni sull'attuale orientamento del dispositivo.
var currentScreenOrientation = screen.orientation;

// Impostare forzatamente una certa orientazione e
// bloccarla.
screen.lockOrientation("portrait-secondary");

// Agganciare un ascoltatore per percepire i cambiamenti
// di orientazione.
screen.addEventListener("orientationchange",
handleScreenOrientationFun, false);

// Utilizzare l'evento handleScreenOrientationFun.
// Generato quando l'orientazione dello schermo cambia.
screen.onorientationchange = handleScreenOrientationFun;
```

- **Touch Events Version 1** - permette di definire le varie tipologie di input touch che generano eventi, di ricevere informazioni riguardanti la locazione dove è stato generato un evento touch e di gestire gli eventi multi touch. Ci sono vari tipi di eventi touch per i quali si può associare un gestore di eventi.
  - touchstart, che viene generato quando si tocca lo schermo.

  - touchmove, che viene generato quando si fa scorrere il dito sullo schermo.
  - touchend, che viene generato quando si rimuove il dito dallo schermo.
  - touchcancel, che viene generato quando si cancella un tocco.

  Ogni sistema operativo definsce a modo suo questi eventi andando ad impostare le tempistiche tra un evento e l'altro, anche per la gestione della differenza di significato tra un tap e un double-tap e il loro riconoscimento.

- **Vibration API** - permettono di implementare differenti tipologie di vibrazione su dispositivi che utilizzano Tizen. Ogni vibrazione è caratterizzata da un tempo di durata e da un pattern di vibrazione. L'interfaccia legata alla vibrazione è implementata da tutte le istanze Navigator. Tramite questa interfaccia è possibile impostare gli attributi di vibrazione quali time e pattern, tramite l'invocazione del metodo vibrate().

Grafica

Le W3C/HTML5 API permettono di gestire elementi grafici tramite l'utilizzo di due componenti.

- **HTML5 Canvas** - permette di creare immagini, figure e testi sfruttando gli elementi presenti nell'HTML5 canvas e nell'HTML canvas 2D. Queste API sono caratterizzate principalmente dalle seguenti funzionalità.
  - Inserire immagini, tramite l'invocazione del metodo drawImage() andando a specificare informazioni come url e posizione.
  - Disegnare figure, tramite l'invocazione di metodi come rect() per rettangoli, arc() per cerchi, ecc ... Si possono specificare informazioni relative a dimensione e posizione.
  - Usare stili e trasformazioni, per quanto riguarda le trasformazioni si possono invocare metodi quali scale(), translate(), transform() e rotate().

- **HTML5 SVG** - permette di creare e gestire gli elementi SGV. Inserendo il tag svg nel body del file HTML si possono creare immagini, testi e figure come mostrato nel seguente codice.

```
// Immagini.
<svg xmlns="http://www.w3.org/2000/svg">
   <image xlink:href="http://developer.tizen.org/sites/
   all/themes/tizen_theme/logo.png" x="10" y="10"
   width="224" height="74" />
</svg>

// Testi.
<svg xmlns="http://www.w3.org/2000/svg">
   <text x="60" y="150">Hello World</text>
</svg>

// Figure.
<svg xmlns="http://www.w3.org/2000/svg">
   <circle cx="150" cy="100" r="60" fill="blue"
   stroke="red" stroke-width="3" />
</svg>
```

Multimedia

Queste API offrono funzionalità specifiche per gestire gli aspetti multimediali.

- **getUserMedia** - permette di accedere al dispositivo per generare flussi multimediali. Le principali caratteristiche sono l'utilizzo dei contenuti multimediali del dispositivo e la loro gestione. Per utilizzare i contenuti si invoca il metodo webkitGetUserMedia() della classe navigator, che ritorna il contenuto richiesto come un oggetto JSON. Per gestire questi contenuti si possono utilizzare metodi quali getCapture(), webkitGetUserMedia(), drawImage().

- **HTML Media Capture** - permettono di reperire file generati da componenti hardware del dispositivo. I principali formati di file

multimediali che possono essere reperiti sono filesystem, camera, camcorder e microphone.

- **HTML5 Video and Audio Element** - permette di gestire ed ottenere informazioni riguardanti file audio e video, utilizzando gli opportuni tag dell'HTML5. Le principali funzioni di queste API permettono di creare un player, che può utilizzare i metodi Play() e Pause(), recuperare informazioni sul file, riprodurre da un punto random del file, recuperare informazioni sullo stato di caricamento del file tramite l'evento Progress e controllare se il formato del file è supportato dal sistema.

- **Web Audio API** - permette di riprodurre un file audio utilizzando il tag appropriato dell'HTML5. Queste API mettono a disposizioni funzionalità per il caricamento dei file audio, la riproduzione del file, l'interruzione della riproduzione e l'instradamento del segnale audio verso risorse di output come altoparlanti o attraverso la connessione ad AudioNodes.

Comunicazione

Queste API forniscono le funzionalità per gestire gli aspetti riguardanti la comunicazione del dispositivo verso il server.

- **HTML5 Session History of Browsing Contexts** - permette di salvare i contenuti della pagina visualizzata dall'utente. Per salvare i contenuti possono essere utilizzati oggetti di stato riguardanti aspetti come la posizione, il livello di scorrimento della pagina o specifici elementi del DOM. Le principali funzioni di queste API permettono di aggiungere oggetti di stato tramite il metodo pushState() contenuto nell'interfaccia Hystory o aggiornare le informazioni tramite il metodo replaceState(). Si possono recuperare informazioni riguardanti i cambi di stato tramite l'evento popstate che incorpora le informazioni contenute nei metodi sopracitati.

- **HTML5 Web Messaging** - permette di istaurare un canale di comunicazione tra il sito Web e il dispositivo per l'invio e la rice-

zione di dati. Queste API utilizzano un meccanismo di Cross-document Messaging tramite la chiamata al metodo postMessage(), che contiene parametri quali message, targetOrigin e opzionalmente il parametro transfer. Il canale di comunicazione instaurato è chiamato MessageChannel, che utilizza due proprietà port1 e port2, per comunicare mediante l'invio di messaggi utilizzando il metodo postMessage().

- **Server-Sent Events** - permette di scambiare dati con un server attraverso il meccanismo di Push. Il meccanismo di Push è stato implementato tramite un server-sent event, un'evoluzione dalla precedente tecnica utilizzante iFrame nascosti attraverso un Ajax polling e Comet. Queste API definiscono una struttura, un'interfaccia e un meccanismo di comunicazione per realizzare il server Push. Le principali caratteristiche di queste API consistono nella crazione di istanze dell'interfaccia EventSource e nella ricezione dei dati dal server tramite Push, questi dati verrano rappresentati da oggetti di classe MessageEvent.

- **WebSocket API** - permette di connettere il dispositivo ad un soket server, per inviare e ricevere dati. Le principali caratteristiche di queste API permettono di instaurare una connessione al server tramite la creazione di un'interfaccia WebSoket, una volta che la connessione non è più richiesta può essere chiusa invocando il metodo close(). Inviare dati utilizzando il metodo send() presente sull'interfaccia WebSoket, che potrebbe lanciare un'eccezione InvalidStateError se l'attributo readyState contiene il valore CONNECTING. Ricevere dati attraverso l'evento message.

- **XMLHttpRequest Level 2** - permette di inviare e ricevere dati di diversa natura monitorando tutto il progresso della cominicazione, tramite l'utilizzo del CORS (Cross-Origin Resource Sharing).

Salvataggio

Queste API forniscono le funzionalità per il salvataggio dei dati dell'applicazione. Troviamo diverse categorie di API dedite a questo compito.

- **File API** - permettono di acquisire informazioni su contenuti di file, data object e di gestire i file system in modalità sandbox. Una delle caratteristiche di queste API è la gestione dei file locali, che permette di leggere le informazioni riguardanti ai file, di utilizzare l'interfaccia FileList, che crea e ritorna un oggetto di classe File contenente le informazioni richieste. I risultati derivanti dalla lettura delle informazioni dei file possono essere visualizzati come testo, file binari o in formato dataURL utilizzando i metodi dell'interfaccia FileReader. La gestione dei file system attraverso la sandbox permette di accedere ai file system usando l'interfaccia LocalFileSystem, di visualizzare i file system nella sandbox utilizzando il metodo readEntries() dell'interfaccia DirectoryReader, di creare directory o file senza l'ausilio della sandbox invocando il metodo getDirectory() o getFile() dell'interfaccia DirectoryEntry e rimuovere una directory o un file tramite i metodi removeRecursively() dell'interfaccia DirectoryEntry e il metodo remove() dell'interfaccia Entry.

- **HTML5 Application Caches** - permette di creare una cache dove memorizzare e gestire dati. L'utilizzo delle cache porta vari benefici quale l'utilizzo delle applicazioni quando si è scollegati dalla rete, un caricameto più veloce delle risorse e una riduzione del carico di lavoro del server. Queste API permetto di attivare, di aggiornare e di gestire la memoria cache allocata sul dispositivo.

- **Indexed Database API** - permetto di creare database per il salvataggio di dati e la loro gestione. Le API provvedono alle funzionalità base per la creazione di un Database, la creazione di un Object Store, la gestione dei dati memorizzati e la creazione di un indice.

- **Web SQL Database** - permette di creare un database che utilizzi le funzionalità dell'SQL. Queste API permetto di segure approcci sincroni e asincroni al Database fornendo funzioni per la creazione del database, l'esecuzione di istruzioni SQL, l'accesso al database e la gestione degli errori.

- **Web Storage** - permette di salvare i dati sia in modo permanente che temporaneo legato alla sessione di connessione. I dati salvati

possono essere condivisi tra tutte le finestre del browser, nel caso di dati temporanei quando vengono aperte nuove finestre i dati vengo automaticamente copiati e messi a disposizione della nuova finestra. Quando una finestra del browser viene chiusa, tutti i dati temporanei associati a quella finestra vengono cancellati.

Sicurezza

Queste API implementano meccanismi di sicurezza per proteggere le risorse del dispositivo e il dispositivo stesso. Questi meccanismi di sicurezza sono caratterizzati dalle seguenti funzionalità che posssono essere utilizzate da una applicazione Web di Tizen.

- Cross-Origin Resource Sharing, che permette ad un'applicazione di richiedere dati da una risorsa esterna, attraverso un meccanismo di CORS (cross-origin resource sharing), che richiede i dati tramite l'invio di una richiesta COR (cross-origin request) all'applicazione.

- HTML5 Iframe Element, che permette di abilitare o bloccare specifici contenuti di una pagina Web.

Interfacce

Queste API introducono funzionalità per arricchire le interfacce grafiche.

- Clipboard API and Events, che permettono di copiare e incollare i contenuti del documento in apposite aree editabili dell'interfaccia.

- HTML5 Drag and Drop, che permettono di creare e gestire elementi che possono essere spostati dall'utente all'interno dell'interfaccia. Queste API permettono inoltre di implementare gli eventi di trascinamento degli elementi. Gli eventi di trascinamento possono essere di varie tipologie come dragstart, drag, dragleave, dragenter, dragover, drop e dragend e possono inviare dati attraverso l'interfaccia DataTransfer.

Performance e ottimizzazione

Queste API introducono funzionalitá per gestire ed ottimizzare le applicazioni.

- Page Visibility, queste API permetto di analizzare ed individuare i cambiamenti dello stato di visibilità dei documenti.
- Timing Control for Script-based Animations, che permettono di controllare la velocità delle animazioni gestendo le richieste di aggiornamento del frame.
- Web Workers, che permettono di creare dei Web worker che servono per mandare in esecuzione thread JavaScript indipendenti in modalità background per non gravare sulle performance del dispositivo.

Localizzazione

Le W3C Geolocation API vengono utilizzate per definire interfacce contenenti le informazioni relative alla localizzazione del dispositivo. Queste informazioni possono essere raccolte da risorse quali GPS, indirizzo IP, Wi-Fi, RFID, il MAC address del Bluetooth e celle identificative GSM/CDMA. Le informazioni riguardanti la localizzazione del dispositivo possono essere ottenute tramite il metodo getCurrentPosition(), oppure si può avere un aggiornamento continuo della posizione tramite il metodo watchPosition(). Questi metodi possono essere invocati dall'interfaccia Geolocation.

3.5.3.4 API supplementari

Queste API non sono contenute nelle funzionalità introdotte da W3C, ma mettono a disposizione importanti strumenti per lo sviluppo di applicazioni che implementano aspetti grafici.

FullScreen

Sono API adibite a supportare la visualizzazione degli elementi appartenenti all'applicazione, quando l'applicazione richiede di essere visualizzata

in modalità fullscreen. Queste API supportano sia l'attivazione che la sospensione di questa modalità e permettono anche di associare fogli di stile CSS alla modalità fullscreen. Tizen supporta questa modalità attraverso API basate sul WebKit, perciò è richiesto l'inserimento del prefisso webkit quando si utilizzano queste API.

Ci sono due metodi che permetto di gestire la modalità fullscreen.

- webkitRequestFullScreen(), per visualizzare uno specifico elemento in modalità fullscreen.
- webkitCancelFullScreen(), per disabilitare la visualizzazione fullscreen.

Typed Array

I Typed Array sono elementi definiti dalle Khronos API, progettati per contenere dati binari complessi prodotti dal WebGL. Questi elementi possono contenere dati di diverso tipo come protocolli Network, file binari, buffer grafici. In aggiunta questo elemento può essere utilizzato per gestire i dati complessi provenienti da applicazioni Web come funzioni video e audio, Web soket e API. Le principali funzionalità di queste API permetto di creare un array buffer, di utilizzare i typed array, di usare le typed array views nei vari formati a disposizione.

WebGL

Appartenente alle Khronos API, il WebGL è una libreria grafica che supporta la creazione di elementi 3D utilizzabili nelle applicazioni, senza alcun supporto di plug-in. Questa libreria è stata progettata come rendering degli elementi canvas dell'HTML5, in più utilizza le API per il rendering 3D provenienti dalla tecnologia OpenGL ES 2.0 . Il WebGL permette di importare contenuti, di utilizzare programmi e shader, di utilizzare buffer, di creare animazioni ed effetti 3D, di visualizzare sullo schermo i contenuti.

Capitolo 4

Sviluppo applicazione con Tizen

4.1 Introduzione

Quando ci si trova a dover sviluppare un'applicazione basata su tecnologie Web con Tizen è importante assicurasi di essere in possesso dei tools offerti dalla piattaforma, che sono raccolti nell'SDK di Tizen scaricabile gratuitamente dal sito ufficiale. Perciò come prima cosa ho reperito l'SDK e l'ho installato sul mio computer per poter cominciare lo sviluppo dell'applicazione. Prima di iniziare a scrivere righe di codice è molto importante fare una buona pianificazione del processo di sviluppo dell'applicazione, quindi sono partito dalla fase di progettazione andando a definire concettualmente l'organizzazione e le funzionalità di cui l'applicazione necessita.

4.2 Progettazione

Nello sviluppo di ogni software o applicazione questa fase è molto importante, perché ci da la possibilità di mettere in chiaro gli scopi e la struttura che l'applicazione deve andare a ricoprire.

4.2.1 Struttura

Il primo passo è stato valutare quale struttura dovesse avere il progetto dell'applicazione, quindi ho deciso di adottare una struttura dove all'interno della cartella principale del progetto fossero presenti solo i file riguardanti le configurazioni e la pagina iniziale di apertura dell'applicazione. Tutte le altre risorse dell'applicazione ho pensato di racchiuderle in sottocartelle dividendole per tipologia di risorsa. Adottare una stuttura organizzata in questo modo porta a mio avviso diversi vantaggi, uno dei quali ci permette di separare concettualmente in gruppi le diverse risorse utilizzate dall'applicazione aggevolando la fase di sviluppo, dove ci si ritroverà spesso a fare modifiche mirate a parti specifiche dell'applicazione consentendo così di velocizzare il processo di sviluppo. La suddivisione non si ferma solo alle cartelle, ma anche all'interno dell'applicazione, infatti ho pensato di suddividere le diverse parti dell'applicazione in vari file, questo per creare un sistema composto da più componenti specializzati. La decisione di separare le varie parti dell'applcazione e racchiuderle in diversi file è dovuta sempre a motivi di sviluppo, perché in caso di modifiche o correzione di errori presenti nel codice ci permette di muoverci velocemente fra le varie parti dell'applicazione. Questa suddivisione molto specifica delle varie parti ha un ulteriore vantaggio, nel caso ci si ritroverà a dover sviluppare una seconda applicazione con alcune funzionalità uguali a questa, si potranno riutilizzare le parti già sviluppate risparmiando così tempo. Utilizzando questo approccio chiunque si trovi a leggere i codici sorgenti potrà capire più facilmente il funzionamento dell'applicazione e quale logica o strategia c'è stata dietro all'implementazione, così da essere un ulteriore vantaggio per chi vorrà in un futuro estendere o migliorare l'applicazione. Nel caso specifico dell'applicazione che ho sviluppato, il risultato di tale progettazione strutturale si è rispecchiato nel seguente modo, come mostrato in figura 4.1.

Per tutte queste ragioni ritengo che seguire questo stile architetturale sia alla base di una buona programmazione, sia per i vantaggi che porta nel singolo progetto sviluppato sia in un ottica di futuri sviluppi.

4.2.2 Obbiettivi

Quando si progetta un'applicazione bisogna aver chiaro quali saranno gli obbiettivi che l'applicazione dovrà andare a completare. Per il mio caso di interesse ho deciso di sviluppare un'applicazione che mi permettesse di

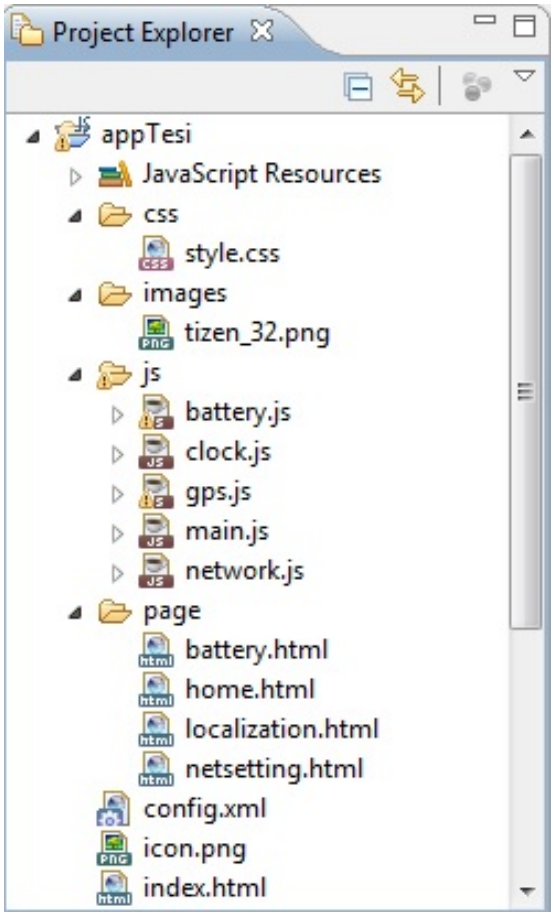

Figura 4.1: Struttura progetto App Tesi.

approfondire le seguenti potenzialità offerte dal Web framework di Tizen, per la gestione delle risorse del dispositivo mobile.

- Batteria.
- Connettività.
- GPS.
- NFC.

Una volta stabilito quali dovessero essere le funzionalità dell'applicazione ho iniziato a progettare l'organizzazione delle varie parti. Essendo chiaro che questa applicazione ha un semplice compito di analisi e gestione di vari aspetti legati al dispositivo, non ho ritenuto necessario che ogni funzionalità avesse una propria interfaccia, poiché questa applicazione è stata pensata per fornire servizi di informazione e gestione che non sono molto articolati. Quindi ho deciso di adottare un'unica interfaccia dove al suo interno le varie funzionalità fossero suddivise in tab. Dal mio punto di vista, date le caratteristiche dell'applicazione, questo approccio è molto performante, perché permette di creare un'applicazione molto leggera, che riduce al minimo il carico di lavoro del processore durante il passaggio da una sezione all'altra dell'applicazione.

Analizzando la tecnologia a mia disposizione ho deciso di adottare una strategia basata sull'utilizzo di iframe, perché questo mi permette di andare a sviluppare il contenuto relativo ad ogni funzione, in un singolo file html esterno contenente solo il codice relativo alla funzione, il quale avrà a sua volta associato il proprio file javascript che implementa le funzioni richieste. La scelta di questo approccio è stata condizionata dalla necessità di ottimizzare l'applicazione durante la navigazione dell'utente all'interno delle varie sezioni, per cercare durante ogni spostamento di minimizzare il caricamento di dati, così da aumentare le prestazioni.

Il risultato di questa organizzazione e suddivisione dei compiti, mi ha permesso di definire le linee giuda da seguire durante lo sviluppo dell'applicazione, senza aver scritto una linea di codice.

4.3 Gestione Interfaccia

Per lo sviluppo dell'applicazione ho deciso di implementare due interfacce grafiche index.html e home.html. L'interfaccia principale ha il compito di informare l'utente sulle funzionalità che l'applicazione svolge permettendogli di accedere alla seconda interfaccia, la quale si occupa di gestire l'iterazione dell'utente con le varie aree di interesse dell'applicazione, come mostrato in figura 4.2. Per lo sviluppo di queste interfaccie ho utilizzato i nuovi divisori specializzati offerti dall'HTML5. La prima interfaccia è standard, quindi non mi soffermo ulteriormente, mentre la seconda l'ho organizzata seguendo le linee guida che avevo stabilito durante la fase di progettazione. Infatti in questa interfaccia ho utilizzato solo tre tag specializzati quali l'header, il nav e l'aticle. Nell'header è stato contenuto il titolo dell'applicazione, come mostrato nel seguente codice.

```
<header id="header">
  <hgroup>
    <h1>Tesi App</h1>
      <p>Questa applicazione \‘e stata creata durante la tesi.</p>
    </hgroup>
</header>
```

Nel nav sono stati inseriti tutti i tab dell'applicazione, che permettono di navigare nelle varie sezioni. Per implementare questo meccanismo ho deciso che al verificarsi dell'evento di click su una delle sezioni venisse richiamato il metodo relativo a quella sezione, che permettesse così di cambiare il contenuto all'interno del divisore article. Se per esempio venisse cliccato il tab Batteria verrebbe richimata la funzione seguente presente nel file main.js.

```
function toBattery(){
  document.getElementById("article").innerHTML =
    "<iframe id=’frameArt’ src=’battery.html’ " +
    "scrolling=’no’ frameborder=’0’ width=’100%’ height=’72%’" +
    "></iframe>";
}
```

La sezione article è stata così pensata per contenere esclusivamente gli iframe relativi ai diversi file html dell'applicazione, che verranno di volta

in volta caricati in base all'iterazione con l'utente. Oltre i contenuti da visualizzare, dentro i file html specializzati viene incorporato anche il footer dell'interfaccia. Ho messo i footer nella pagine html relative a sezioni specifiche dell'applicazione, perché ho voluto usarli per contenere i bottoni relativi ai comandi. Questo è stato necessario poiché quando si richiamano funzioni javascript che agiscono su componenti del documento, i contenuti esterni al documento come quelli rappresentati dagli iframe, non possono essere acceduti, quindi se si vogliono manipolare i contenuti presenti in un iframe, occorre che le funzioni javascript siano lanciate all'interno dell'iframe. Questo mi ha permesso un'ulteriore specificazione delle varie parti, poiché ogni file html si serve solo dei file javascript dedicati alle funzioni che deve ricoprire, come mostrato nel seguente codice relativo al file battery.html .

```
<head>
  <link rel="stylesheet" type="text/css" href="../css/style.css"/>
  <script src="../js/clock.js"></script>
  <script src="../js/battery.js"></script>
</head>
```

La transazione dalla prima alla seconda interfaccia è stata implementata tramite l'utilizzo di un semplice bottone.

```
<div align="center">
  <a href="./page/home.html">
    <button style="width: 80px; height: 40px;">Accedi ai servizi
    </button>
  </a>
</div>
```

4.4 Funzionalità sviluppate

Durante lo sviluppo delle varire funzionalità dell'applicazione, oltre ad esaminare le caratteristiche di ogni specifico ambito, ho avuto la possibilità di verificare l'aspetto di Tizen legato alla gestione dei privilegi. Infatti la maggior parte delle API possono essere utilizzate liberamente, ma nel caso specifico di questa applicazione, per risorse quali NFC e Bluetooth, sono

Figura 4.2: Interfaccie grafiche App Tesi.

dovuto andare a modificare il file config.xml per aggiungere i privilegi di cui necessitavo. La gestione dei privilegi legati ad un'applicazione è un aspetto cardine nello sviluppo di applicazioni Tizen, che permette di instaurare tutti i meccanismi di sicurezza e protezione menzionati nel capitolo precedente. Sono stato felice di constare che la gestione di un aspetto così importante viene attuata secondo una metodologia molto semplice ed intuitiva, che secondo me rappresenta un buon vantaggio, perché ritengo conveniente facilitare l'implementazione di aspetti riguardanti la sicurezza e la protezione. Quindi mi sono servito del relativo tool per la gestione di questo file, che ha prodotto al suo interno le seguenti modifiche andando ad aggiungere al file config.xml i seguenti codici.

```
<tizen:privilege name="http://tizen.org/privilege/bluetooth.gap"/>
<tizen:privilege name="http://tizen.org/privilege/nfc.common"/>
```

4.4.1 Batteria

Durante l'implementazione di questa funzionalità ho constatato che le API offerte da Web framework di Tizen permettono di reperire importanti informazioni sullo stato di questa risorsa del dispositivo, come ad esempio:

- Livello energetico della batteria.
- Se il dispositivo è attaccato all'alimentazione.
- Il tempo di carica della batteria restante.
- Il tempo a disposizione prima della scarica del dispositivo.

L'iterazione con questo componente è di particolare importanza, perché ci permette di implementare delle politiche di risparmio energetico basate sullo stato della batteria, nel caso particolare dell'applicazione sviluppata ho implementato la seguente strategia. Per acccedere ad informazioni quali il livello energetico della batteria ho utilizzato il seguente codice.

```
// L'attributo dipende dalla piattaforma utilizzata.
// Il simulatore utilizza l'attributo battery.
// Tizen utilizza l'attributo webkitBattery.
// In questo modo l'istruzione e' indipendente da chi la esegue.
```

```
var batteria = navigator.battery || navigator.webkitBattery
               || navigator.mozBattery;

levelBat = batteria.level * 100;
```

Una volta che l'utente preme il bottone per controllare il livello energetico della batteria, se il livello dovesse risultare basso, viene richiamata la funzione stopResorce(), la quale spegne le risorse NFC e Bluetooth, per mandare il dispositivo in regime di risparmio energetico.

```
// Controllo il livello della batteria.
if(levelBat<20){
  stopResorce();
}

function stopResorce(){

  var b = tizen.bluetooth.getDefaultAdapter();
  var n = tizen.nfc.getDefaultAdapter();

  if(b.powered){
    // "ok" ed "errorB" sono call-back.
    b.setPowered(false, ok, errorB);
  }

  if(n.powered){
    // "ok" ed "errorN" sono call-back.
    n.setPowered(false, ok, errorN);
  }
}
```

4.4.2 GPS

Durante lo sviluppo di questa sezione dell'applicazione ho scoperto che il Web framework di Tizen permettono di reperire le informazioni dal GPS del dispositivo, in due diverse modalità a seconda delle necessità. Le API del W3C definiscono un'interfaccia navigator per gestire l'oggetto di classe

Geolocation. Questo oggetto è utilizzato negli script per reperire progressivamente informazioni sulla posizione associate al dispositivo, che sono acquisite mediante un algoritmo specifico user-agent. L'interfaccia dell'oggetto di classe Geolocation è implementata dal seguente codice.

```
interface Geolocation {
   void getCurrentPosition(PositionCallback successCallback,
                           optional PositionErrorCallback errorCallback,
                           optional PositionOptions options);

   long watchPosition(PositionCallback successCallback,
                      optional PositionErrorCallback errorCallback,
                      optional PositionOptions options);

   void clearWatch(long watchId);
};

callback PositionCallback = void (Position position);

callback PositionErrorCallback = void (PositionError positionError);
```

La prima modalità ritorna semplicemente dati relativi alla possizione attuale percepiti dal GPS del dispositivo mobile. La seconda modalità permette di monitorare costantemente i dati relativi alla posizione forniti dal GPS, fino a quando non verrà richiesta la sua interruzione, come mostrato nelle seguenti porzioni di codice.

```
// Richista dati di localizzazione
// "successo" ed "errore" sono call-back
navigator.geolocation.getCurrentPosition(successo, errore,
   {maximumAge: 60000});

// Richiesta monitoraggio GPS.
// "successo" ed "errore" sono call-back
watchId = navigator.geolocation.watchPosition(successo, errore);

// Stop monitoraggio GPS.
navigator.geolocation.clearWatch(watchId);
```

In caso di errore si andrà a gestire le varie eccezioni visualizzando a video i relativi messaggi d'errore, ma per quanto riguarda la corretta ricezione dei dati relativi alla posizione del dispositivo, si invocherà la funzione di call-back descritta nel seguente codice.

```
function successo(position){
  document.getElementById("localizzazione").innerHTML =
     "<pre>Latitudine: &#9;<label>" + position.coords.latitude +
     "</label><br>Longitudine:&#9;<label>" +
     position.coords.longitude + "</label></pre>";
}
```

4.4.3 Connessioni

In questa sezione dell'applicazione sono andato ad analizzare l'acquisizione di informazioni riguardanti la connessione internet del dispositivo, la connessione Wi-Fi e la gestione di risorse quali Bluetooth e NFC. Per quanto riguarda la connessione internet e la connessione WiFi, che sono aspetti non trattati dalle API derivanti dall'HTML5, perché la loro gestione è compito delle API di Tizen, purtroppo ho constatato che queste API mettono a disposizione meccanismi per reperire solamente informazioni relative a tutte le risorse presenti sul dispositivo, senza permettere alcuna gestione. Per quanto riguarda risorse quali Bluetooth e NFC, l'HTML5 permette con il suo set di API una buona gestione permettendo di utilizzare appieno queste risorse per la comunicazione con gli altri dispositivi mobile. Nell'applicazione mi sono limitato a controllare e gestire li loro stato di operatività, infatti ho implementato meccanismi che permettono l'utente di accendere e spegnere queste risorse, oltre ad ottenere informazioni sul loro stato di attività. Un esempio è il seguente stralcio di codice relativo al cambiamento di stato della risorsa NFC.

```
function changeNFC(){

  //Occorre aggiungere il privilegio:
  //http://tizen.org/privilege/nfc.common

  var nfc = tizen.nfc.getDefaultAdapter();
```

```
  if(nfc.powered){
    nfc.setPowered(false, okNFC, errorNFC);
  } else {
    nfc.setPowered(true, okNFC, errorNFC);
  }
}

function okNFC(){
  alert("Cambio di stato effettuato");
}

function errorNFC(error){
  alert("" + error.message);
}
```

4.5 Valutazioni finali

Lo sviluppo di applicazioni tramite tecnologie Web risulta molto interessante, perché ho riscontrato diversi vantaggi. In primo luogo l'utilizzo di questi linguaggi di programazione permette uno sviluppo rapido e molto semplificativo, infatti è possibile utilizzare variabili senza doverne specificare il tipo, che rende più immediato l'utilizzo e l'implementazione delle funzioni, anche a livello di costruzione dell'interfaccia grafica, queste tecnologie permettono un rapido sviluppo grazie alle sinergie che si creano fra l'utilizzo dell'HTML e del CSS. Per quanto riguarda il caso dell'utilizzo delle API presenti nel Web framework di Tizen ho apprezzato notevolmente la strutturazione delle chiamate, infatti le API quando vengono invocate possono avere al loro interno dei parametri come le normali chiamate alle funzioni, ma la loro particolarità è rappresentata dal fatto che permettono di definere quali metodi devono essere invocati nel caso in cui si presentino degli errori o la loro esecuzione vada a buon fine, come rappresentato nel seguente frammento di codice. Questa struttura semplifica notevolmente la gestione e l'utilizzo delle API.

```
var blue = tizen.bluetooth.getDefaultAdapter();

if(blue.powered){
```

```
  //Chiamata API setPowered(boolean, function(), function()).
  blue.setPowered(false, okBlue, errorBlue);
}
```

Capitolo 5

Conclusioni

Questa tesi è stata sviluppata per approfondire le conoscenze relative alle applicazioni per dispositivi mobile. Quando mi sono addentrato in questo argomento ho potuto constatare che esistono diversi tipi di applicazioni mobile e quelle che hanno catturato maggiormente il mio interesse sono state le applicazioni basate su tecnologie Web. Questa tipologia di applicazioni ha catturato maggiormente il mio interesse, perché sono più recenti e rappresentano l'anello di congiunzione fra le applicazioni native e le applicazioni indipendenti. Infatti questo mi ha portato a pensare che costituiscono un importante ed interessante caso di studio, perché offrono la possibilità di aumentare le potenzialità delle applicazioni native andando ad integrare tutti i vantaggi derivanti dall'utilizzo delle tecnologie Web, perciò ho ritenuto necessario approfondire le mie conoscenze in questo ambito. Una volta inquadrato il percorso da seguere mi sono posto le seguenti domande:

- Quali sono le loro caratteristiche?
- Quali sono i nuovi sistemi orientati al loro utilizzo?
- Come vengono supportate dal sistema?
- Quali sono le reali potenzialità?
- Come si implementano?
- Si possono avere dei vantaggi, sia a livello implementativo che a livello esecutivo?

Osservando le varie possibilità che oggi ci si presentano, per approfondire questo argomento ho scelto come caso di studio Tizen, che mi ha permesso di aumentare la mia conoscenza sulle applicazioni basate su tecnologie Web e di rispondere a quelle domande che mi ero posto. Sono stato contento di questa mia scelta, perché questa piattaforma è stata a mio avviso organizzata e progettata molto bene, infatti riesce a fornire una buona visione di questo nuovo modo di sviluppare ed utilizzare le applicazioni per dispositivi mobile.

Entrando nel merito dello sviluppo di queste applicazioni ho riscontrato che l'utilizzo di tecnologie Web porta con se diversi vantaggi. Le tecnologie Web mettono a disposizione diversi meccanismi, che se sviluppati in linguaggio nativo richiederebbero un'implementazione più complessa. A livello di sviluppo del codice risulta più semplice l'implementazione di tutta la parte riguardante l'interfaccia dell'applicazione. Ho potuto constatare che queste tecnologie di nuova generazione come l'HTML5 e il CSS3 hanno definito un orientamento verso lo sviluppo di applicazioni per dispositivi mobile, infatti sono presenti API specifiche per l'iterazione con le risorse del dispositivo. Ho scoperto che queste applicazioni utilizzano le tecnologie Web per sfruttare anche le funzionalità native della piattaforma, questo vantaggio è rappresentato dal fatto che non verrà richiesto lo sviluppo di codice nativo, ma ci si potrà avvalere di queste funzioni mediante i linguaggi propri delle tecnologie Web, che a mio parere caratterizza ulteriormente il concetto di utilizzo di tecnologie Web.

Putroppo queste applicazioni sono ancora deboli dal punto di vista della gestione delle risorse, infatti ho potuto constatare che sulla maggior parte delle risorse presenti nel dispositivo, si possono solamente ottenere informazioni riguardanti lo stato e gli attributi che le caratterizzano, a mio avviso un'importante limitazione per chi vuole sviluppare applicazioni per gestire le impostazioni del dispositivo. Ciò nonostate queste applicazioni permettono l'utilizzo delle varie risorse del dispositivo, infatti rispetto alle applicazioni native non ho riscontrato un grosso divario nel loro utilizzo.

5.1 Futuri sviluppi

Quello delle applicazioni basate su tecnologie Web è un argomento che merita di essere approfondito ulteriormente, soprattutto dal lato pratico andando ad analizzare tutti i vari aspetti che possono essere utilizzati nello

sviluppo di un'applicazione. Infatti sarebbe interessante approfondire il loro studio andando a confrontare il divario prestazionale che si ha con le applicazioni native e confrontare anche le varie differenze a livello di sviluppo del codice. Approfondendo questo studio potremmo capire meglio gli effettivi vantaggi derivanti dall'utilizzo di queste tecnologie e soprattutto saremo in grado di valutare meglio per quali applicazioni conviene utilizzare queste tecnologie.

È mio parere che queste applicazioni potranno essere migliorate ulteriormente in futuro e credo che aumenterà il loro utilizzo. Il mio consiglio è quello di andare ad arricchire le loro funzionalità nella gestione delle risorse, che da come ho notato è un aspetto ancora carente, infatti estendendo ulteriormente le loro potenzialità potranno trovare utilizzo anche in quegli ambiti che al momento sono di dominio delle applicazioni native.

Bibliografia

[1] Immagine tratta da :. http://elinux.org/.

[2] Immagine tratta da :. http://www.subfurther.com/.

[3] Immagine tratta da :. http://blog.radvision.com/.

[4] Immagine tratta da :. http://www.qnx.com/.

[5] Immagine tratta da :. http://developer.nokia.com/.

[6] Immagine tratta da :. http://developer.android.com/.

[7] Immagine tratta da :. https://developer.apple.com/.

[8] Immagine tratta da :. http://docs.oracle.com/.

[9] Immagine tratta da :. http://www.webmasterpoint.org/.

[10] Immagine tratta da :. https://developer.tizen.org/.

[11] Sito di w3c dedicato allo stato dell'arte della tecnologia css3. http://www.w3.org/Style/CSS/current-work.

[12] Sito html.it con guide per gli sviluppatori. http://www.html.it/development/guide/.

[13] Sito mr.webmastert, con guide per dispositivi mobile. http://www.mrwebmaster.it/mobile/.

[14] Sito per gli sviluppatori symbian, di proprietà nokia. http://developer.nokia.com/Devices/Symbian/.

[15] Sito ufficiale android per gli sviluppatori.
`http://developer.android.com/index.html`.

[16] Sito ufficiale apple per sviluppatori, sezione iphone.
`https://developer.apple.com/devcenter/ios/`.

[17] Sito ufficiale blackberry per gli sviluppatori.
`http://developer.blackberry.com/`.

[18] Sito ufficiale di tizen, giude agli sviluppatori.
`https://developer.tizen.org/`.

[19] Sito ufficiale inglese di wikipedia.
`http://en.wikipedia.org/wiki/`.

[20] Sito ufficiale italiano di wikipedia.
`http://it.wikipedia.org/wiki/`.

[21] Sito ufficiale windows phone per gli sviluppatori.
`http://developer.windowsphone.com/en-us`.

[22] M. Carli. *Android 3. Guida per lo sviluppatore*. Apogeo, 2011.

[23] M. Firtman. *Programmare per il web mobile*. Tecniche Nuove, 2013.

[24] K.-J. K. E.-H. H. Yeong-Jun Kim, Seong-Jin Cho. Benchmarking java application using jni and native c application on android. *12th International Conference on Control, Automation and Systems*, 2010. KINTEX, Gyeonggi-do, Korea.

Elenco delle figure

1.1 Dispositivo Mobile: Cellulare 7
1.2 Dispositivo Mobile: Tablet 10
1.3 Dispositivo Mobile: PDA 12
1.4 Dispositivo Mobile: Smartphone 14
1.5 Architettura sistema operativo Android [1]. 18
1.6 Architettura sistema operativo iOS [2]. 22
1.7 Architettura sistema operativo Windows Phone [3]. 30
1.8 Sistema operativo QNX [4]. 32
1.9 Architettura sistema operativo Symbian [5]. 34

2.1 Ciclo di vita di un Activity [6]. 48
2.2 Ciclo di vita di un Service [6]. 49
2.3 iOS Model-View-Controller [7]. 52
2.4 Ciclo di vita di un'applicazione iOS [7]. 54
2.5 Tipica struttura per applicazione Web [8]. 64
2.6 Ciclo di vita di un'applicazione Web [8]. 65
2.7 Struttura HTML5 [9]. 70

3.1 Architettura della piattaforma Tizen [10]. 82
3.2 Ciclo di vita di un'applicazione Tizen, prima parte [10]. . . . 94
3.3 Ciclo di vita di un'applicazione Tizen, seconda parte [10]. . . 95
3.4 Threads di un'applicazione Tizen [10]. 97

4.1 Struttura progetto App Tesi. 131
4.2 Interfaccie grafiche App Tesi. 135

Glossary

API Application Programming Interface, in informatica è un termine utilizzato per indicare un insieme di procedure utilizzate dai programmatori.. 15, 17, 21, 29, 33, 45, 66, 69, 77, 82, 83, 85, 92, 99, 100, 102–105, 107–111, 114, 117, 120, 122, 123, 125, 126, 134, 136, 137, 139, 140, 144

Bluetooth Tecnologia per la trasmissione di dati tra due dispositivi, attraverso una frequenza radio a corto raggio. 2, 7, 10, 13, 27, 82, 101, 109, 126, 134, 137, 139

CPU È l'unità di elaborazione centrale, un processore digitale che si occupa gestire tutta l'elaborazione del sistema.. 15, 27, 28, 33, 39, 111

driver In informatica rappresenta quell'insieme di comandi e procedure, che il sistema operativo utilizza per comandare le risorse hardware del dispositivo.. 20, 27, 28, 32, 36, 83

dual-core Indica uno specifico processore avente due nuclei per l'elaborazione dei dati.. 14

FireWire Interfaccia standard sviluppata da Apple, per la trasmissione seriale dei dati attraverso un cavo in due diverse modalità, asincrona e isocrona.. 13

framework È una struttura logica utilizzata in informatica, per dare supporto allo sviluppo di software, tramite l'utilizzo di librerie contenenti insiemi di codice già implementati.. 15, 21, 23–25, 27, 29, 35, 36, 43, 65, 75, 77, 82, 86, 92, 99, 103, 112, 132, 136, 137, 140

full window Modalità di visualizzazione, che permette al software di avere l'uso esclusivo del display.. 80

Garbage Collector È un componente che si occupa della gestione automatica della memoria, che viene occupata dai processi durante la loro esecuzione.. 16, 30, 31

GB Gigabyte, in informatica è un'unità di misura della memoria convertibile in 1024 Megabyte.. 11, 14, 74

GPS Global Positioning System, un sistema di posizionamento che utilizza reti di satelliti in orbita, per determinare coordinate geografiche e orari.. 4, 8, 11, 19, 40, 43, 86, 126, 137, 138

hardware In informatica questo termine indica l'insieme dei componenti elettronici, che consentono il funzionamento di un dispositivo.. 3, 4, 9, 11, 13, 15, 17, 20, 21, 28, 34, 36, 39, 40, 75, 78, 83, 86, 104, 116, 121

HDMI High-Definition Multimedia Interface, uno standard digitale per l'interfaccia di segnali audio e video.. 13

hub È un dispositivo di rete che gestisce lo smistamento dei dati tra i vari dispositivi che vi sono collegati, generalmente utilizzato nelle reti con una topologia a stella. 29

JNI Java Native Interface, un framework del linguaggio Java, che consente di richiamare il codice nativo di un determinato sistema operativo.. 41

layout In informatica rappresenta l'impaginazione e la struttura grafica di un documento.. 46, 107, 116

License Key In italiano si traduce come chiave di licenza, che rappresenta quel codice alfanumerico richiesto da un software per poter essere utilizzato dall'utente.. 60

link In informatica rappresenta un collegamento fra un trasmettitore e un ricevitore, in pratica un canale di comunicazione.. 70, 102

memory leaks In informatica rappresenta una perdita o una fuoriuscita di memoria, che è generalmente causata dalla cattiva gestione.. 33, 90, 98

multi-core È un termine per indicare quei processori dotati di più nuclei per l'elaborazione dei dati.. 14, 28

multitasking Questo termine in informatica rappresenta la capacità di un sistema di mandare in esecuzione più processi contemporaneamente, attraverso diverse politiche di assegnazione della CPU.. 2, 20, 28, 31, 33, 93

multithreading Questo termine in informatica rappresenta la capacità di un processore di eseguire più thread allo stesso tempo, che a differenza dei processi condividono la stesso spazio d'indirizzamento, dato che fanno parte dello stesso processo. 33

MVC Model-View-Controller, un pattern architetturale molto difuso per lo sviluppo di software, nell'ambito della programmazione ad oggetti.. 33, 52

NFC Near Field Communication, una tecnologia per instaurare connessioni wireless a corto raggio.. 82, 109, 134, 137, 139

pattern In informatica indica un modello, una struttura o uno schema, che viene definito per lo sviluppo di software o per risolvere un determinato problema ricorrente.. 16, 52, 120

ROM Read Only Memory, ovvero la memoria che può solo essere letta, una memoria non volatile, cioé che non viene eliminata allo spegnimento, dove i dati vengono memorizzati.. 33

SDK Software Development Kit, un insieme di strumenti e di documentazioni pensati per lo sviluppo di software.. 42–44, 50, 56, 84, 87, 88, 129

software In informatica rappresentano quei programmi per la gestione di informazioni.. 9, 11–13, 15, 19, 21, 29, 31, 36, 39, 40, 42, 78, 86, 88, 90, 104, 129

tag È un metadato, che rappresenta una parola chiave utilizzata per descrivere un oggetto, al fine di rendere possibile una facile gestione dei contenuti, attraverso ricerche basate sulle parole chiave.. 69, 70, 121, 133

tethering Tecnologia che consente di utilizzare il colegamento alla rete internet di un dispositivo mobile, da parte di altri dispositivi che vi si collegano tramite le tecnologie a loro disposizione, che possono essere Bluetooth, Wi-Fi o tramite USB. 9, 13

thread In informatica rappresenta il flusso di controllo di un processo, che detto più semplicemente è la serie di linee di codice che il processore dovrà computare.. 27, 96

USB Universal Serial Bus, standard di comunicazione seriale utilizzato per creare collegamenti fra diversi dispositivi, al fine di scambiare dati.. 2, 8, 10, 13, 29

Virtual Machine Un software che simula il comportamento di una macchina fisica, attraverso un processo di virtualizzazione.. 16, 17, 19, 30, 31, 35, 41

Wi-Fi Tecnologia che permette a due dispositivi di instaurare un collegamento, senza il bisogno di supporti fisici quali cavi, ma tramite l'utilizzo di antenne.. 7–9, 13, 85, 126, 139

WPS Wi-Fi-based positioning system, un sistema di posizionamento che utilizza le reti Wi-Fi e la potenza del segnale, per determinare la posizione del dispositivo.. 87