ALMA MATER STUDIORUM - UNIVERSITA' DI BOLOGNA

SEDE DI CESENA

FACOLTA' DI SCIENZE MATEMATICHE, FISICHE E NATURALI

CORSO DI LAUREA IN SCIENZE E TECNOLOGIE INFORMATICHE

Strumenti mobile interattivi al servizio del cittadino

TESI DI LAUREA IN MOBILE WEB DESIGN

Relatore:

Dott. Mirko Ravaioli

Presentata da:

Giacomo Rossi

Sessione III

Anno Accademico 2011-2012

INDICE

INTRODUZIONE

L'obiettivo di questa tesi di Laurea è la creazione di un'applicazione su piattaforma iOS che consente di visualizzare le news e gli eventi del Comune di Cesena e di fruire di alcuni servizi erogati dagli uffici comunali messi a disposizione dei cittadini.

L'idea è nata dalla collaborazione tra l'azienda BSD@Software di Cesena, presso la quale ho realizzato l'applicazione, con il reparto di sistemi informativi del Comune di Cesena, dalla quale è stata possibile la stesura di un documento d'analisi delle principali funzionalità implementate dall'applicazione quali: visualizzazione dello stato delle pratiche anagrafiche, visualizzazione della sede elettorale, prenotazione presso gli uffici dell'edilizia e visualizzazione delle news.

Sono state analizzate le tecnologie utilizzate e le motivazioni che portano alla loro scelta. In particolare, sono descritti la tecnologia web service REST utilizzata per l'implementazione delle chiamate di lettura dati su ambiente Java e l'ambiente di sviluppo iOS per la realizzazione dell'app vera e propria.

La struttura del documento è organizzata secondo il seguente schema: nel Capitolo 1 viene fatto uno studio sugli strumenti mobile, quali smartphone e tablet, l'influenza che hanno avuto sulla vita di tutti i giorni, e sui principali sistemi operativi mobile più utilizzati. Nel Capitolo 2 viene presentata la fase progettuale dell'applicazione nella quale sono introdotte le principali funzioni dell'applicazione e descritte le specifiche di progettazione, quali interrogazioni e struttura del database. Nel Capitolo 3 è descritta la fase implementativa che introduce rispettivamente le scelte fatte per quanto riguarda l'ambiente di sviluppo utilizzato e una descrizione per la parti server e client dell'app. Infine vengono presentati possibili sviluppi futuri per le nuove versioni che verranno sviluppate.

1.1 DISPOSITIVI MOBILE

I dispositivi mobili di ultima generazione stanno diventando i nuovi personal computer e il loro mercato è in continua e rapida crescita.

Si sono sviluppati notevolmente negli ultimi dieci anni. Non solo gli schermi sono cresciuti in dimensioni e qualità, ma anche l'hardware interno è cresciuto fino a raggiungere livelli di performance viste solo nei computer portatili alcuni anni fa.

Gli utenti grazie a questi dispositivi possono trovare informazioni su vari livelli, che vanno dai social network ai dati aziendali ed e-mail, essendo multifunzionali. Non vi è più la necessità di portare con sé un telefono, un lettore musicale e una fotocamera digitale. Tutti i dispositivi mobili odierni possono compiere tutti questi compiti. La maggior parte di questi offre servizi di geolocalizzazione, ovvero utilizzano il sistema di posizionamento globale (GPS) per determinare dove sono nel mondo.

I dispositivi mobili si dividono in tre categorie principali:

- dispositivi palmari: spesso indicati come PDA (Personal Digital Assistant) sono computer di dimensioni ridotte dotati di uno schermo tattile, tali da essere comodamente utilizzati sul palmo di una mano (da cui il nome). Sono molto utilizzati per la scrittura di e-mail, note e promemoria.

- dispositivi smartphone: dispositivi portatili che abbinano funzionalità di telefono cellulare a quelle di gestione di dati personali.

- tablet: è un computer con un formato simile a quello di un notebook, con in più la possibilità di scriverci a mano grazie ad un dispositivo, detto “digitizer”, che è una via di mezzo tra una tavoletta grafica e un touch screen.

- Ultra-mobile PC: una via di mezzo tra un palmare e un portatile. Integrano gli stessi sitemi operativi dei tradizionali PC e possono utilizzare come input sia la tastiera che lo schermo touchscreen.

1.1.1 DISPOSITIVI SMARTPHONE

Uno smartphone, o in italiano telefonino intelligente, è un dispositivo portatile che abbina funzionalità di telefono cellulare a quelle di gestione di dati personali.

Derivano dall'evoluzione dei PDA, a cui si aggiungono le funzioni di telefono. Un aspetto molto importante da sottolineare è la possibilità di installare applicazioni (App) nel dispositivo aggiungendo nuove funzionalità. Questi programmi possono essere sviluppati dal produttore dello smartphone, dallo stesso utilizzatore o da terze parti, rilasciate poi in genere in forma gratuita o a pagamento sul mercato.

Un aspetto fondamentale della loro diffusione è da attribuire ai loro sistemi operativi. Oggi i sistemi operativi più utilizzati presenti sul mercato sono cinque: Android, Apple iOS, Windows Phone 7 e 8, BlackBerry OS, Symbian.

1.2 SISTEMI OPERATIVI PER DISPOSTIVI MOBILE

Negli smartphone, il sistema operativo è il software preinstallato che gestisce tutte le funzioni che il dispositivo è in grado di svolgere.

I primi sistemi operativi installati su telefoni cellulari erano dotati di funzioni molto limitate, ma nel corso degli anni si ebbe una notevole evoluzione.

Nel 1996 Palm, Inc. rilascia il primo sistema operativo mobile della storia. Sviluppato unicamente per palmari, offriva come funzioni aggiuntive il promemoria, le note e memo per i contatti.

In seguito, nel 1999, fu lanciato sul mercato BlackBerry OS, sviluppato da Research In Motion; il loro obiettivo era di realizzare telefoni in ambito business.

L'anno 2000 vide il rilascio da parte di Microsoft di Pocket PC 2000 con caratteristiche aggiuntive quali Windows Messenger, Media Player etc.

Nel 2005 Google acquisisce Android Inc e dopo due anni, nel 2007, Apple lancia il primo modello di iPhone integrando come sistema operativo iOS.

Nel 2008 gran parte degli sviluppatori lanciarono versioni sempre più nuove e aggiornate dei loro sistemi operativi, con particolare rilevanza per l'avvento del primo dispositivo Android.

Dal 2008 i mercati sono stati colpiti da un grande numero di aggiornamenti di versioni di mobile OS, sia di successo sia di fallimento. Nel 2010 Microsoft rilascia Windows Phone 7, destinato a diventare uno dei tre sistemi operativi mobili più utilizzati sul mercato insieme a Android e iOS.

Questi saranno descritti in modo più approfondito nei capitoli successivi.

1.2.1 iOS

iOS (originariamente noto come iPhone OS) è un sistema operativo basato su piattaforma UNIX costruito per iPhone, iPod touch e iPad. E' stato introdotto e rilasciato come sistema operativo per iPhone nel giugno 2007. Sono stati venduti più di un milione di unità iPhone nel primo mese e mezzo. Al momento si contano milioni di dispositivi IPhone in tutto il mondo. Negli anni sono state rilasciate diverse versioni di iOS, la versione attuale è iOS 6. L'App Store è il servizio realizzato da Apple che permette agli utenti di scaricare e acquistare applicazioni disponibili in iTunes Store. Le applicazioni possono essere sia gratuite che a pagamento, e possono essere scaricate direttamente dal dispositivo o su un computer.

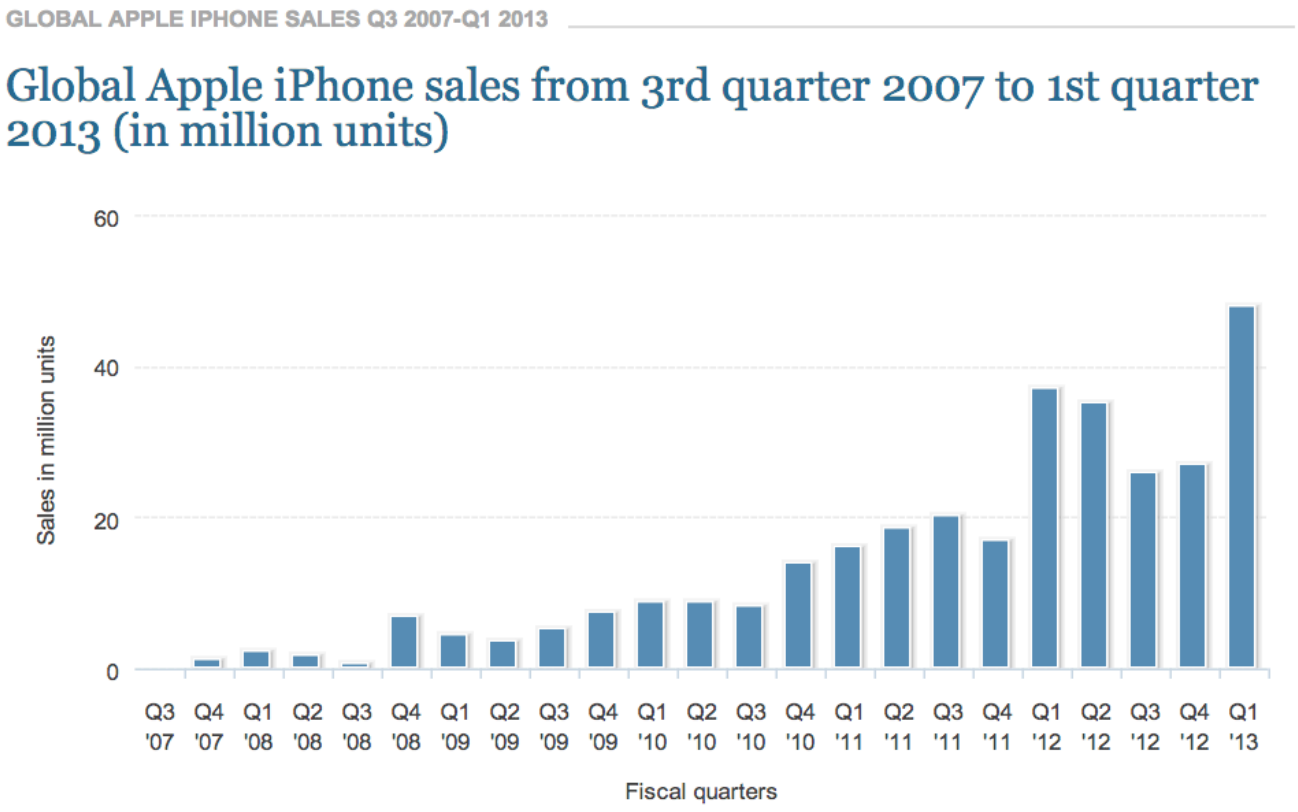


Figura 1 - Vendite iPhone negli ultimi cinque anni

Caratteristiche tecniche

iOS può essere presentato come un proxy tra l'hardware dei componenti hardware e delle applicazioni che appaiono sullo schermo del dispositivo. Le applicazioni interagiscono con l'hardware attraverso interfacce di sistema che le proteggono da modifiche hardware. Il loro sviluppo si basa su un linguaggio orientato agli oggetti chiamato Objective-C, che supporta la stessa sintassi di base del C.

Anche se iOS è stato progettato per i dispositivi mobili, condivide una variante dello stesso kernel Mach di base che viene utilizzato in Mac OS X, e molte tecnologie con esso. L'architettura iOS comprende quattro livelli: Core OS, Core Services, Media e Cocoa Touch.

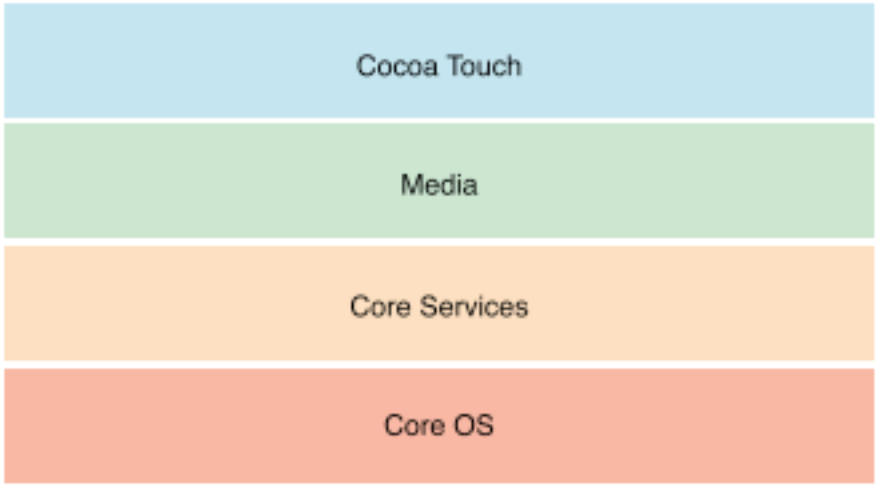


Figura 2 - Architettura iOS

Core OS e **Core Services** sono strati di livello inferiore. Questi strati includono interfacce per l'accesso ai file, tipi di dati di basso livello e network socket.

Le principali tecnologie del livello Core OS includono, ad esempio, framework di sistema per gestire l'accesso al file system, allocazione di memoria, e framework di sicurezza con lo scopo di proteggere i dati. Core Services include SQLite per integrare database SQL nelle applicazioni, e il supporto XML. Le interfacce in entrambi gli strati sono per lo più basati sul linguaggio di programmazione C.

Lo strato **Media** contiene funzioni per immagini, audio e video. AirPlay è costruito in questo strato. Le funzioni nello strato Media sono per lo più sviluppate in C, ma contengono anche un avanzato motore di animazione sviluppato in Objective-C chiamato Core Animation.

Ogni applicazione iOS utilizza il framework UIKit dello strato **Cocoa Touch** per implementare caratteristiche di base come taglia, copia e incolla, le informazioni sullo stato della batteria, il supporto per testo, contenuti web, interfaccia utente e la gestione delle applicazioni. La maggior parte delle funzioni di Cocoa Touch utilizzano il linguaggio Objective-C.

Ambiente di sviluppo

L'SDK iOS contiene tutti gli strumenti necessari per la progettazione, creazione, debug e ottimizzazione del software per iOS. Gli strumenti che lo costituiscono sono i seguenti: frameworks, librerie standard condivise, Xcode, Interface Builder, simulatore iOS e l'iOS Developer Library. Xcode è l'applicazione principale per lo sviluppo di applicazioni e viene utilizzato per la gestione di progetti applicativi, la modifica, la compilazione, esecuzione e debug del codice. Interface Builder è uno strumento per assemblare l'interfaccia utente.

iOS Simulator può essere utilizzato per testare le applicazioni iOS su Mac OS X, rendendo il test più veloce, dato che lo sviluppatore non ha bisogno di caricare l'applicazione su un dispositivo mobile. L'iOS Developer Library è la principale fonte per la documentazione relativa allo sviluppo di applicazioni.

1.2.2 ANDROID

Google, insieme alla Open Handset Alliance (OHA), ha lanciato il sistema operativo mobile Android, alla fine del 2007. Android è un sistema open source realizzata allo scopo di migliorare l'esperienza mobile per gli utenti finali. Appena Android è stato rilasciato, gli strumenti di sviluppo e i tutorial sono stati resi disponibili per aiutare gli sviluppatori a scoprire il nuovo sistema.

L 'ultima versione rilasciata di Android è la 4.2 denominata Jelly Bean. Google Play Store (precedentemente denominato Android Market) è lo store di applicazioni, brani musicali, pellicole cinematografiche, libri e riviste online sviluppato da Google per i dispositivi Android, per i sistemi desktop e notebook che abbiano disponibilità della connessione Internet, che ha sostituito l'Android Market, il quale trattava solo applicazioni per Android.

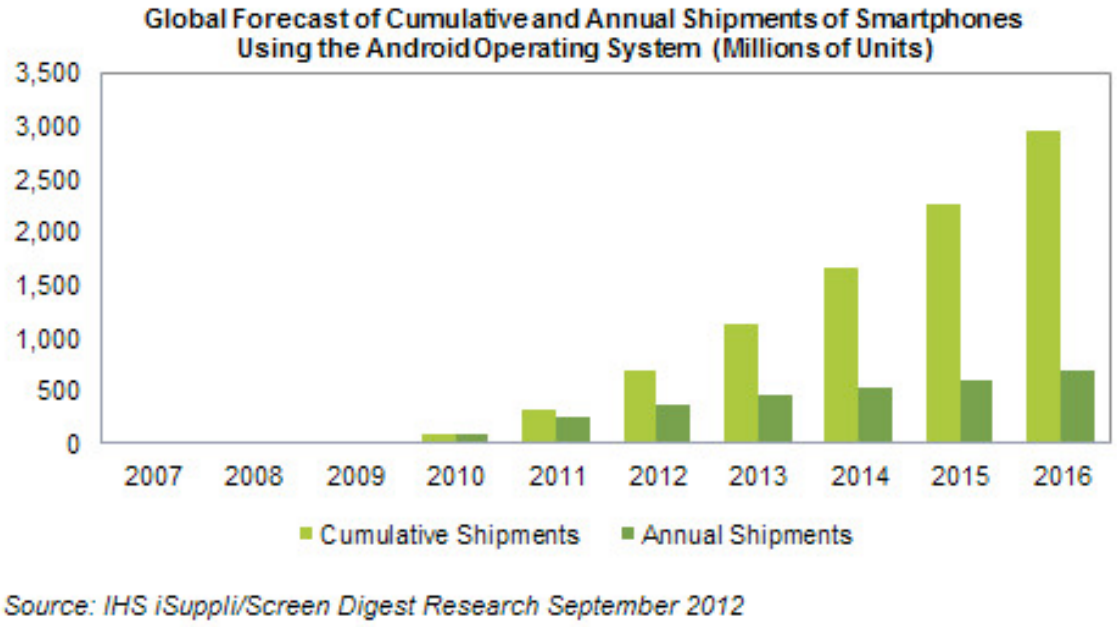


Figura 3 - PRevisione vendite smartphone che utilizzano Android

Caratteristiche tecniche

Android è uno stack di software costruito per i dispositivi mobili. Si tratta di un sistema operativo di base, middleware e alcune applicazioni di base. Le applicazioni Android possono essere programmate con il linguaggio Java.

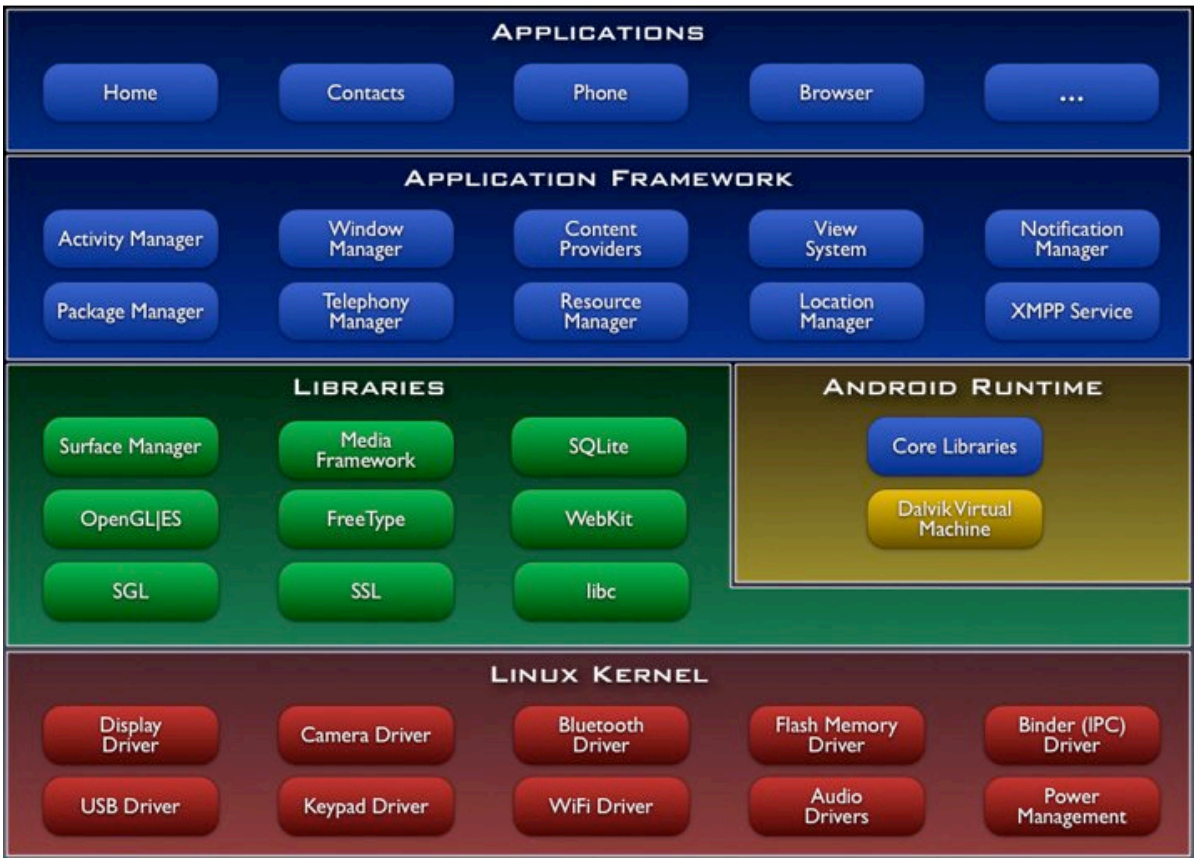


Figura 4 - Architettura Android

Per impostazione predefinita, tutti i dispositivi Android hanno un client di posta elettronica, un programma di messaggistica di testo, un calendario, un software per le mappe (di solito Google Maps), un browser Internet, un'applicazione per la gestione dei contatti, e altre applicazioni di base installate in loro.

A causa della piattaforma di sviluppo open-source di Android, gli sviluppatori hanno piena libertà per creare applicazioni molto innovative, consentendo loro di avere pieno accesso ai framework API che sono utilizzati dalle applicazioni core.

Android include un insieme di librerie C/C++ che sono utilizzate da vari componenti del sistema. Le librerie principali e le loro funzioni sono: System C Library, Media Libraries, Surface Manager, LibWebCore, SGL, 3D Libraries, FreeType, SQLite.

Android utilizza una determinata macchina virtuale conosciuto come Dalvik per eseguire ciascuna delle sue applicazioni. Queste applicazioni vengono eseguite all'interno dei propri processi ciascuno dei quali ha la loro propria instanza nella macchina virtuale Dalvik. Un dispositivo può eseguire in modo efficiente più macchine virtuali contemporaneamente. Dalvik si basa sul kernel Linux, che è l'intermediario tra l'hardware e il software del dispositivo. Il kernel si occupa anche di servizi core di Android di sistema,ovvero la sicurezza globale, la gestione della memoria, gestione dei processi, sistemi di rete collegati e il modello di driver.

Ambiente di sviluppo

L'Android SDK fornisce un insieme di strumenti necessari per sviluppare applicazioni per il sistema operativo Android. Si consiglia di utilizzare l'ambiente di sviluppo Eclipse per eseguire ADT, Android Development Tools, dai quali l'utente accede a tutti gli strumenti SDK. È possibile sviluppare applicazioni anche senza Eclipse attraverso un editor di testo e la capacità di utilizzare gli strumenti SDK sulla linea di comando o con script.

1.2.3 WINDOWS PHONE

Windows Phone è il sistema operativo per smartphone di Microsoft, completamente differente dalle precedenti versioni di Windows Mobile. E' basato sulle API Win32 sempre di Microsoft, supporta il multitouch, gli schermi capacitivi, e ha una nuova interfaccia grafica.Windows Phone 7 è la prima generazione del sistema operativo uscito nell'ottobre 2010. Windows Phone 8 è la seconda generazione del sistema operativo Windows Mobile rilasciato il 29 ottobre 2012. Esso ha introdotto importanti cambiamenti nell' architettura e molti nuove funzionalità. I segreti di Windows Phone 8 risiedono dell'interfaccia (figura) ben progettata e molto distintiva (rispetto ad Android e iOs) e presenta anche numerosi Widget. L'intuitività dell'interfaccia sono uno dei tanti punti di forza di questo nuovo sistema operativo. Windows Phone Store è il servizio, sviluppato dalla Microsoft, che permette agli utenti di cercare e scaricare applicazioni sviluppate per il sistema operativo.

Figura 5 - Interfaccia Windows Phone 8

Caratteristiche tecniche

Windows Phone 8 è basato sul kernel Windows NT. Windows Phone 8 condivide lo stesso file system (NTFS), stack di rete, elementi di sicurezza, motore grafico (DirectX), driver di periferica e livelli di astrazione dell'hardware (HAL) con Windows 8. In questo modo un'applicazione può essere tradferita tra queste due piattaforme con molto meno sforzo. Questo cambiamento porta anche il supporto per i processori multi-core.

Windows Phone 8 include il motore CoreCLR precedentemente gestito da. NET Compact Framework. Il CoreCLR integra una funzione di auto-tuning garbage collector, che consente una riduzione dei tempi di avvio e maggiore reattività nelle applicazioni.

Scrivere applicazioni per più piattaforme risulta molto semplice, poichè Windows Phone 8 è suportato dai linguaggi di programmazione C e C++.

L'hardware di Windows Phone è in grado di scalare l'interfaccia grafica di un'applicazione per soddisfare qualunque dimensione dello schermo dei dispositive, supportando tre diverse risoluzioni WVGA (800 x 480), WXVGA (1280 x 768), e 720p (1280 x 720).

Il multitasking è notevolmente migliorato, consentendo all'utente di riaprire l'applicazione dallo stesso stato in cui era stata precedentemente chiusa tramite tasto centrale.

Windows Phone 8 utilizza le Live Tiles (piccolo icone sullo schermo) per visualizzare i contenuti interattivi sulla home iniziale evitanto all'utente di spostarsi manualmente dentro e fuori le applicazioni.

In questo modo è possibile visualizzare più informazioni da quelle applicazioni che effettivamente usiamo più spesso, senza però rinunciare a vedere, con un colpo d'occhio, qualche dato anche dalle altre app. Microsoft in questo modo aumenta effettivamente il numero di informazioni visibili rispetto al passato ed allo stesso tempo ha anche svelato le Live Apps, cioè applicazioni capaci di avvantaggiarsi della Lock Screen e fornire contenuti live dalle stesse.

Tra le altre funzionalità ricordiamo che Windows Phone 8 permette di inviare file tramite Bluetooth e ha il supporto per il chip l'NFC, schede micro-SD e Wi-Fi Direct; è garantita una serie di API per l'integrazione di servizi aggiuntivi di photosharing, messagistica e video chiamate; tramite l'app Wallet è possibile memorizzare I dati delle carte di credito per effettuare pagamenti online: integra le mappe Nokia al posto delle mappe Bing.

Ambiente di sviluppo

L'SDK 8.0 fornische gli strumenti necessary per sviluppare giochi e applicazioni per Windows Phone 8 e Windows Phone 7. I più importanti sono:

- Visual Studio Express 2012 edizione per Windows Phone: versione ridotta dell'IDE Visaul Studio che fornisce le funzioni necessarie per lo sviluppo;

- Blend for Visual Studio 2012: strumento per progettare l'interfaccia grafica;

- Windows Phone Developer Registration: consente la registrazione per poter distribuire i propri progetti;

- Windows Phone Connect; utilizzato per collegare il telefono all'ambiente di sviluppo.

- Windows Phone Application Analysis: fornisce un profile per valutare e migliorare la qualità e le prestazioni delle app.

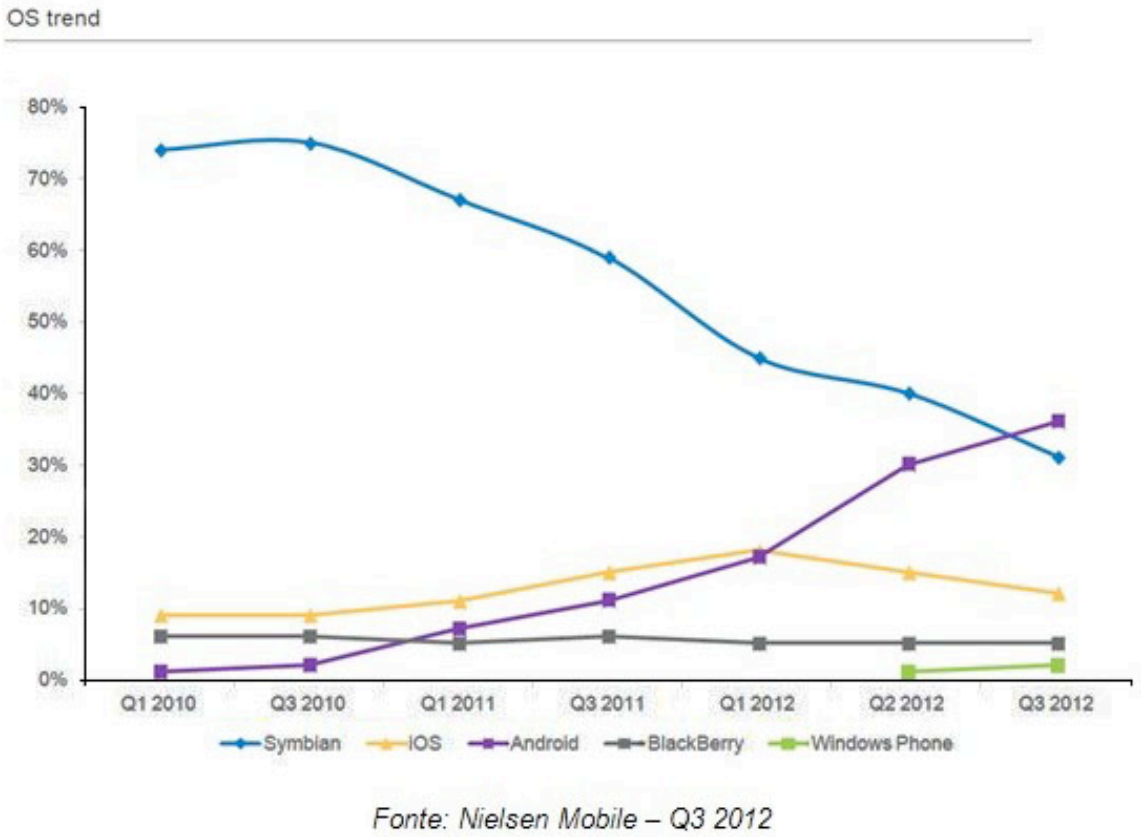
- Simulation Dashboard: offre opzioni di simulazione per garantire che l'applicazione si comporta bene sotto alcuni scenari imprevisti che possono verificarsi nella vita reale.

Micorostf sta lentamente deprecando Silverlight e XNA nella piattaforma Windows Phone. L'SDK consente ancora di realizzare applicazioni basate su Silverlight e XNA, ma solo per Windows Phone 7, mentre la versione 8 si basa su C # 5.0, più semplice e moderno.

1.3 SCENARIO MOBILE IN ITALIA

I dati Nielsen sull'utilizzo del Mobile in Italia nel terzo trimestre 2012 rivelano conferme e importanti cambiamenti. La crescita degli smartphone è un trend che continua ormai da più di due anni e che coinvolge oltre la metà dei possessori di cellulari, ovvero il 56% degli utenti italiani di telefonia mobile.

La vera novità però non è la crescita dei possessori di smartphone, meno netta rispetto agli anni passati, bensì l'andamento dei principali attori all'interno del segmento. Symbian, leader indiscusso con oltre il 70% di quota nel 2010, ha registrato trimestre dopo trimeste un graduale calo, imputabile anche al passaggio di Nokia a Windows, arrivando nel terzo trimestre del 2012 a dover cedere la posizione di leadership ad Android, che, con una crescita senza precedenti, arriva al 36% di quota di mercato, superando ogni altro sistema operativo.



Fonte: Nielsen Mobile – Q3 2012

Figura 6 - Scenario mobile in Italia

Per quanto riguarda gli altri player, rimane sostanzialmente stabile iOS, che nei dati riferiti al Q3 2012 ancora non trae benefici dall'uscita dell'iPhone 5. Degno di nota è invece Windows, per la prima volta con numeri statisticamente significativi che fanno presagire scenari futuri interessanti.

Si tratta di un cambio di leadership storico, che porta il mercato italiano ad una struttura più simile a quella americana e inglese, dove Android è già da tempo il principale sistema operativo. A guidare questo cambiamento sono soprattutto i giovani, considerando che ben il 39% dei possessori di device Android ha meno di 34 anni. Altro aspetto fondamentale è l'accessibilità al grande pubblico: gli utenti Android dichiarano di spendere la metà per il proprio device rispetto a chi possiede iOS, fattore non da poco in un momento di crisi come questo, che vede in continuo calo la spesa percepita per i servizi di telefonia.

Nel corso del terzo trimestre le crescite maggiori sono state registrate nelle applicazioni, nella messaggistica istantanea e, particolare davvero interessante, nel mobile commerce.

Figura 7 - I servizi evoluti con il maggior tasso di crescita

FASE PROGETTUALE

2.1 DESCRIZIONE GENERALE

Il progetto presentato in questa tesi prende il nome di Strumenti mobile interattivi al servizio del cittadino.

L'idea di questo progetto è quella di realizzare un'applicazione su piattaforma iOS tramite la quale l'utente può usufruire di alcuni dei servizi che il Comune di Cesena mette a disposizione del cittadino.
Il progetto è strutturato, sostanzialmente, in quattro parti o servizi:

- **visualizzazione dello stato delle pratiche anagrafiche;**
- **visualizzazione delle sedi elettorali;**
- **prenotazione appuntamenti con i tecnici dell'Ufficio Edilizia;**
- **visualizzazione delle news del Comune.**

2.2 PAGINE DELL'APPLICAZIONE

L'applicazione si compone di più interfacce, la cui struttura è la seguente:

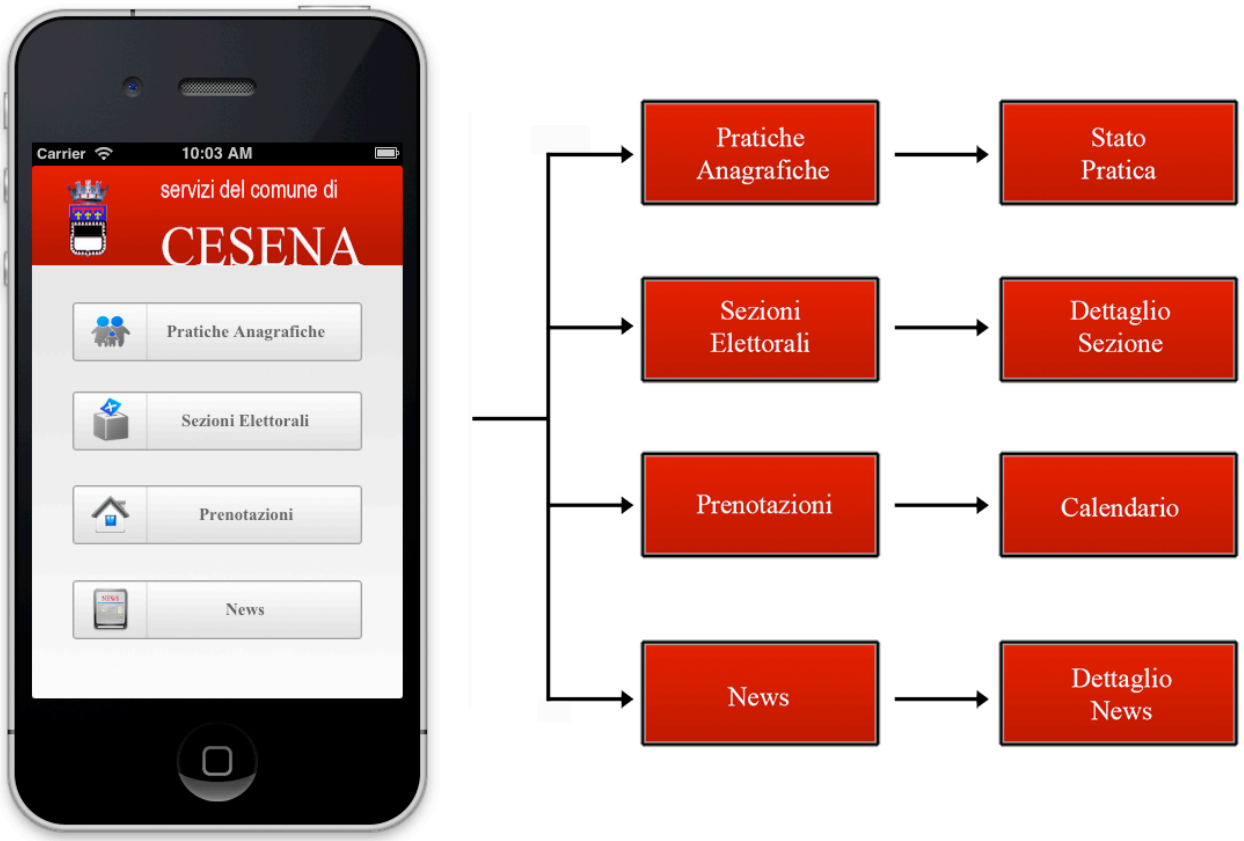


Figura 8 - Pagine dell'applicazione

La schermata Home, più a sinistra, è la prima pagina alla quale l'utente accede.

Attraverso essa è possibile raggiungere tramite gli appositi bottoni le altre quattro sezioni che compongono l'applicazione.

VISUALIZZAZIONE DELLO STATO DELLE PRATICHE ANAGRAFICHE

La prima funzione sviluppata nell'applicazione corrisponde alla possibilità di controllare lo stato attuale di una pratica, in altre parole tutti i passi del procedimento che attraversa prima di essere accertata positivamente. Gli utenti interessati devono aver prima effettuato presso l'Ufficio Anagrafe una dichiarazione relativa a un Cambio di Indirizzo o a una Immigrazione. All'atto della dichiarazione è rilasciata una ricevuta nella quale sono indicati il numero e la data associati alla pratica. Numero, data e tipo della pratica sono i parametri necessari per usufruire di questo servizio, e dovranno essere inseriti all'interno di un form. Dopo aver inserito i dati nella pagina "Pratiche Anagrafiche" cliccando su "cerca" verrà visualizzato l'avanzamento nella schermata "Stato Pratica".

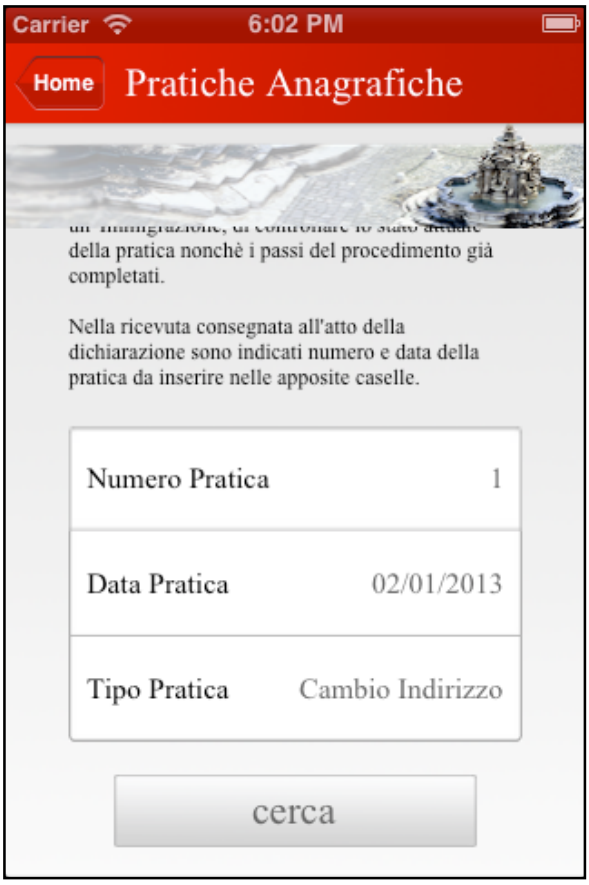


Figura 9 - Inserimento parametri pratiche anagrafiche

Figura 10 - Stato della pratica

VISUALIZZAZIONE DELLE SEDI ELETTORALI

Questo servizio consente agli iscritti alle liste elettorali del Comune di Cesena di verificare il seggio presso il quale si dovranno recare a votare.

Il parametro obbligatorio da immettere per vedere la propria sede elettorale è il proprio codice fiscale, e, dopo aver cliccato su "cerca", verrà visualizzato nella mappa della pagina ("sedi elettorali") un segnaposto posizionato sulla sede elettorale corrispondente.

E' possibile inoltre visualizzare cliccando sula freccia nel segnaposto il numero della sezione, il nome del plesso e l'indirizzo.

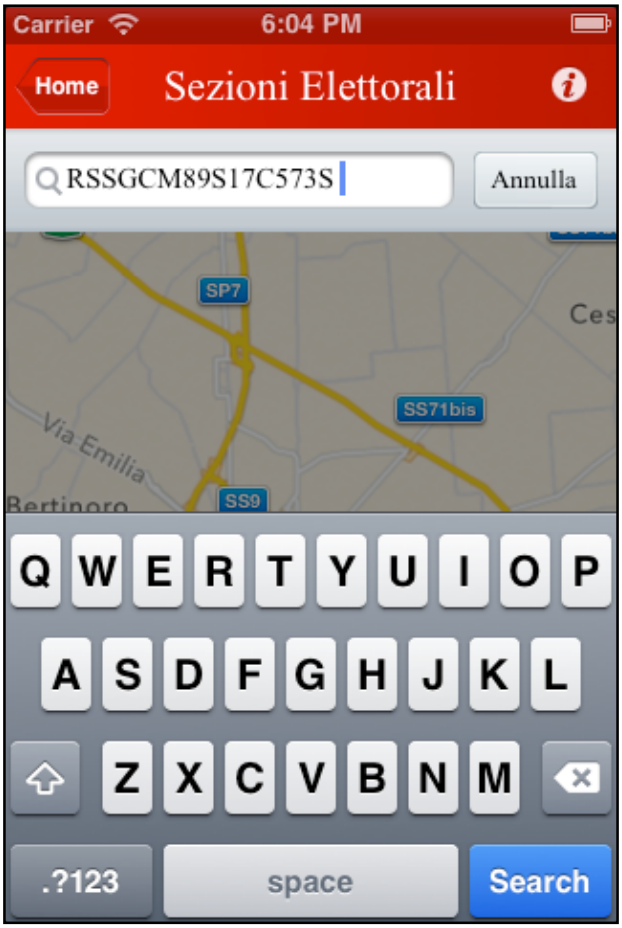


Figura 11 - Inserimento codice fiscale

Figura 12 - Visualizzazione sede elettorale

PRENOTAZIONE APPUNTAMENTI CON I TECNICI DELL'UFFICIO EDILIZIA

Grazie a questa funzione, l'utente può prendere appuntamenti con i tecnici dell'Ufficio Edilizia del Comune di Cesena. L'utente deve specificare in un form un insieme di dati, nell'ordine: nome, cognome, Codice Fiscale, numero di telefono, e-mail, e una volta inseriti, deve selezionare da un menù a tendina il nome del tecnico con il quale vuole prendere appuntamento. Il passo successivo consiste nella selezione del giorno e ora della prenotazione, scelti da un calendario con vista sul mese corrente dove sono presenti i giorni in cui il tecnico riceve gli appuntamenti ai quali sono associate le ore già prenotate e quelle ancora disponibili.

Infine è necessario specificare l'oggetto per il quale l'utente richiede l'appuntamento.

Inseriti tutti i parametri, è sufficiente cliccare su "conferma" per convalidare la prenotazione.

Figura 13 - Prenotazione inserimento parametri

Figura 14 - Conferma prenotazione

VISUALIZZAZIONE DELLE NEWS DEL COMUNE

Dalla schermata “Home” è possibile visualizzare l’elenco delle news del Comune di Cesena premendo sull’apposito pulsante “News”.

L’applicazione legge le ultime notizie ed eventi dal feed RSS del Comune. Un feed RSS è un file di testo che contiene le informazioni per reperire le ultime risorse inserite in un sito web (come un titolo, una breve descrizione ed un link per fruire della risorsa) o anche le risorse per esteso.

Selezionando una voce dall’elenco, l’applicazione mostra la schermata di dettaglio della news dove viene visualizzata la pagina web il cui link corrisponde a quello ricavato dal feed RSS per quella notizia.

Figura 15 - Elenco news

Figura 16 - Visualizzazione news

2.3 SPECIFICHE DI PROGETTAZIONE

L'applicazione è caratterizzata da due parti principali, il lato server e il lato client, sulle la cui progettazione è descritta nel seguito.

Il lato server si occupa di gestire ha il compito di memorizzare e fornire all'applicazione le informazioni relative alle pratiche anagrafiche, alle sedi elettorali e alle prenotazioni.

Il lato client ha il compito di interrogare il lato server per ricevere i contenuti e gestirli per il corretto funzionamento all'interno dell'applicazione, in modo tale da poterle visualizzare all'utente.

La logica lato server si basa su un servizio web, sviluppato appositamente per l'applicazione, collegato alla gestione di un database, attraverso il quale sono ottenute le informazioni aggiornate.

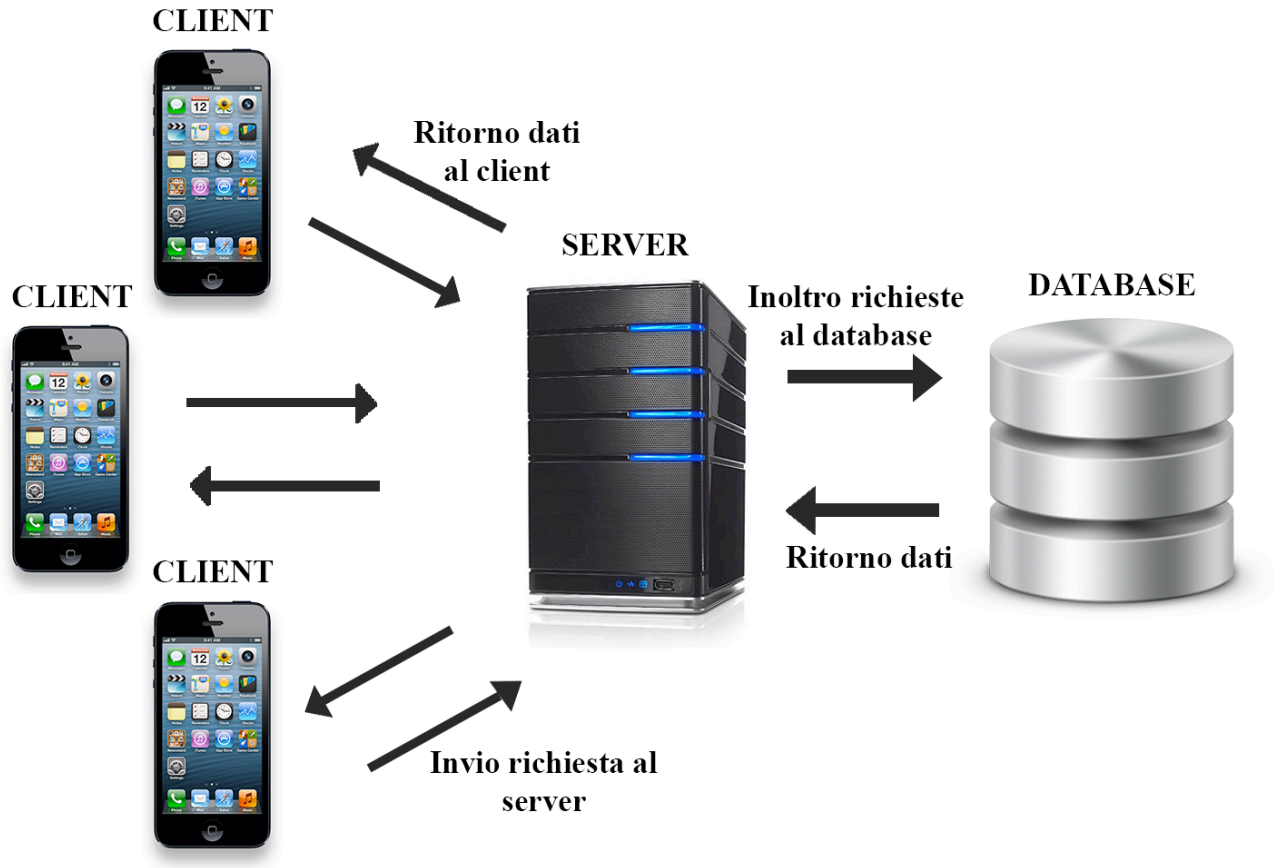


Figura 17 - Schema client-server

Nella figura è illustrato il procedimento di base:

il dispositivo effettua una richiesta al server remoto; il server apre una nuova connessione al database ed esegue su di esso una specifica query; il server riceve i dati dal database e li ritorna al client.

I dati sono memorizzati in file di tipo JSON (Javascript Object Notation) che è un semplice formato fra i più utilizzati per quanto riguarda lo scambio dati in applicazioni client-server.

La struttura dei file JSON inviati dal dispositivo al server prende come modello un oggetto caratterizzato da un insieme di coppie chiave/valore la cui sintassi, racchiusa tra parentesi graffe, è così denotata: “chiave proprietà” : ”valore proprietà”, con una virgola che separa tutte le coppie tranne l’ultima prima della chiusura della parentesi.

Le informazioni inviate dal server al dispositivo, anch’esse memorizzati in file JSON, hanno come struttura un oggetto composto da un messaggio di esito (“ok” o “errore”) e un contenuto che contiene le coppie chiave/valore relative ai dati se l’esito è positivo, oppure una stringa corrispondente alla descrizione dell’errore in caso di esito negativo.

2.3.1 INTERROGAZIONI

Le interrogazioni al database si distinguono in base ai servizi implementati dall'applicazione:

- Ottenimento dettagli sullo stato di una pratica anagrafica:

  con questa richiesta l'applicazione riceve i dati relativi alle fasi di avanzamento dello stato di una pratica, filtrata per numero, tipo e data.

- Ottenimento dati di una sezione elettorale:

  l'applicazione, tramite questa richiesta, riceve tutti le informazioni, filtrate in base al codice fiscale, associate a una sede elettorale, ovvero numero, plesso, indirizzo e le coordinate (latitudine e longitudine) per l'esatto posizionamento del segnaposto nella mappa.

- Ottenimento dei nomi dei tecnici:
  quando si accede alla sezione "Prenotazioni", l'applicazione deve permettere all'utente di selezionare un tecnico da una lista. Questa richiesta riceve i nomi dei tecnici che ricevono appuntamenti dalla data di accesso in avanti.

- Ottenimento impegni per un determinato tecnico:

  consente di ricevere le date relative ai giorni in cui il tecnico, passato come parametro, prende appuntamento.

- Ottenimento prenotazioni disponibili per un determinato giorno:

  necessaria per ottenere le ore che sono ancora disponibili per prenotare un appuntamento in un dato giorno.

- Inserimento di una prenotazione:

  questa richiesta viene effettuata dall'applicazione nel momento in cui una

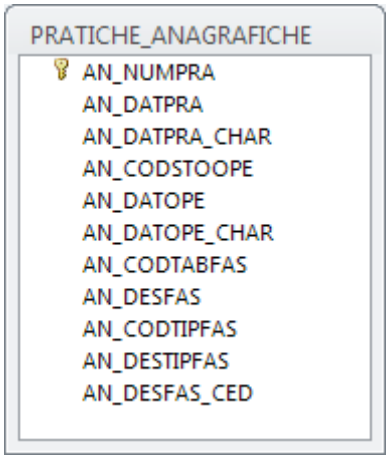
  nuova prenotazione deve essere inserita nel database.

2.3.2 STRUTTURA DEL DATABASE

Tutte le informazioni necessarie al funzionamento dell'applicazione sono mantenute all'interno del database remoto.

Sono prese in considerazione le seguenti tabelle:

- Tabella per la gestione delle pratiche anagrafiche:



I principali campi utilizzati sono:

AN_NUMPRA: numero intero identificativo univoco della pratica;

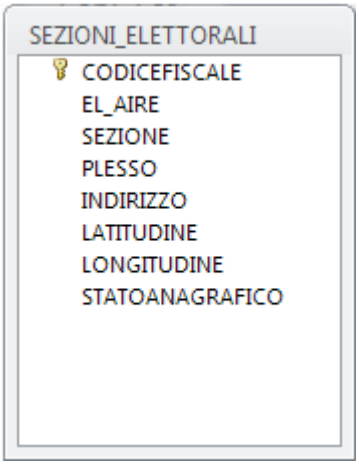
AN_DATPRA: data della pratica in formato gg/mm/aaaa;

AN_CODTIPFAS: numero intero corrispondente al tipo della pratica (1 = Immigrazione, 2 = Cambio Indirizzo);

AN_DESFAS: stringa che descrive e fasi dello stato associato alla pratica;

AN_DATOPE_CHAR: stringa che rappresenta la data associata allo stato della pratica in formato gg/mm/aaaa.

- Tabella per la gestione delle sezioni elettorali:



I principali campi utilizzati sono:

CODICEFISCALE: chiave primaria di tipo stringa di 16 caratteri corrispondente al codice fiscale del cittadino;

SEZIONE: numero intero che rappresenta il numero della sede elettorale;

PLESSO: stringa che memorizza il nome della struttura che ospita la sezione elettorale;

INDIRIZZO: stringa corrispondente agli indirizzi dei vari plessi;

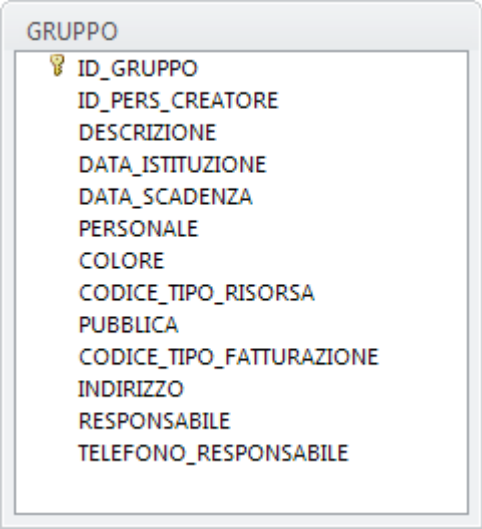
LATITUDINE: numero corrispondente alla latitudine nelle coordinate;

LONGITUDINE: numero corrispondente alla latitudine nelle coordinate.

- Tabelle per la gestione delle prenotazioni:

Tabella GRUPPO:

memorizza i dati di ciascun gruppo:



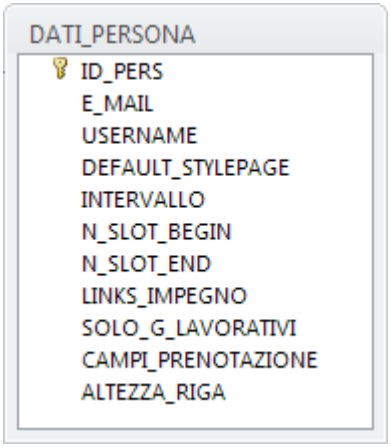
I principali campi utilizzati sono:

ID_GRUPPO: chiave primaria costituita da un numero identificativo univoco associato al gruppo (un gruppo corrisponde a un solo tecnico);

ID_PERS_CREATORE: chiave esterna, importata dalla tabella PERSONA_GRUPPO, corrispondente all'identificativo della persona, che in questo caso sarà un tecnico, associata a quel gruppo;

DESCRIZIONE: Stringa che memorizza il nome e cognome del tecnico associato al gruppo.

Tabelle DATI_PERSONA e PERSONA_GRUPPO:



DATI_PERSONA memorizza i dati di ciascun dipendente all'interno del comune, e di conseguenza di tutti i tecnici. PERSONA_GRUPPO è la reificazione della relazione tra le tabelle GRUPPO e DATI_PERSONA. Presenta una chiave composta che include le due chiavi primarie importate dalle due tabelle collegate.

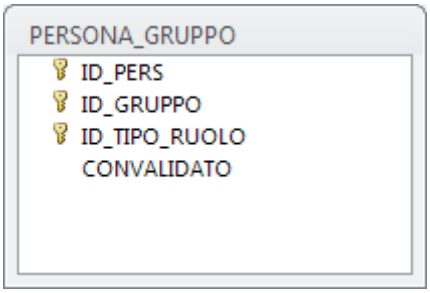
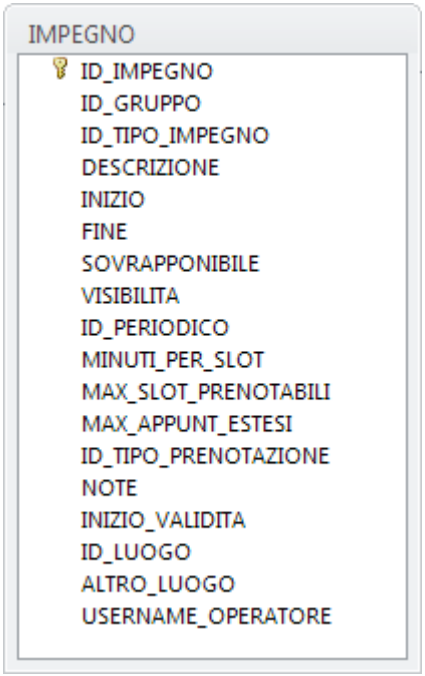


Tabella IMPEGNO:



I principali campi sono:

ID_IMPEGNO: chiave primaria costituita da un numero intero che identifica l'impegno, cioè l'insieme degli appuntamenti in una certa data;

ID_GRUPPO: chiave esterna importata dalla tabella GRUPPO che collega l'impegno a un gruppo;

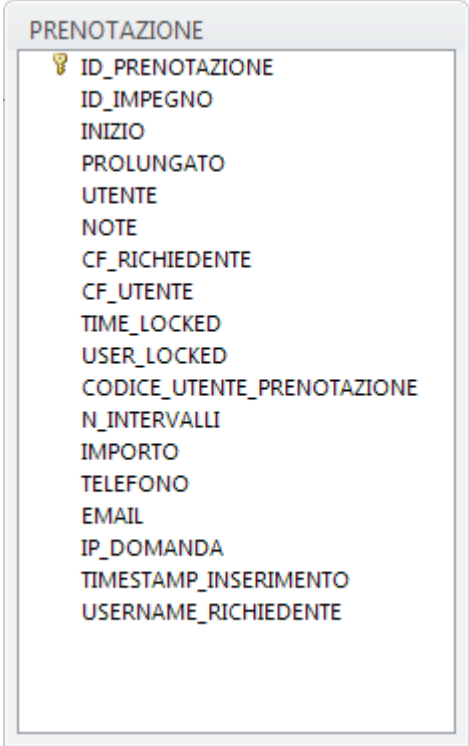
ID_TIPO_IMPEGNO: numero intero che identifica la tipologia d'impegno;

INIZIO: data e ora d'inizio dell'impegno;

FINE: data e ora di fine dell'impegno.

MINUTI_PER_SLOT: numero intero che rappresenta la durata in minuti di ogni appuntamento dell'impegno.

Tabella PRENOTAZIONE:



I principali campi sono:

ID_PRENOTAZIONE: chiave primaria che basata su un numero intero che identifica il singolo appuntamento all'interno di un impegno;

ID_IMPEGNO: chiave esterna importata dalla tabella IMPEGNO, necessaria per collegare l'appuntamento all'impegno associato;

INIZIO: data e ora d'inizio dell'appuntamento;

UTENTE: stringa che rappresenta il nome e cognome dell'utente che ha richiesto la prenotazione;

NOTE: stringa contenente l'oggetto della prenotazione;

CF_UTENTE: codice fiscale dell'utente memorizzato come stringa di 16 caratteri;

TELEFONO: numero di telefono dell'utente;

EMAIL: stringa corrispondente all'email dell'utente;

Le relazioni fra le tabelle che si riferiscono alle prenotazioni sono mostrate nel seguente schema logico:

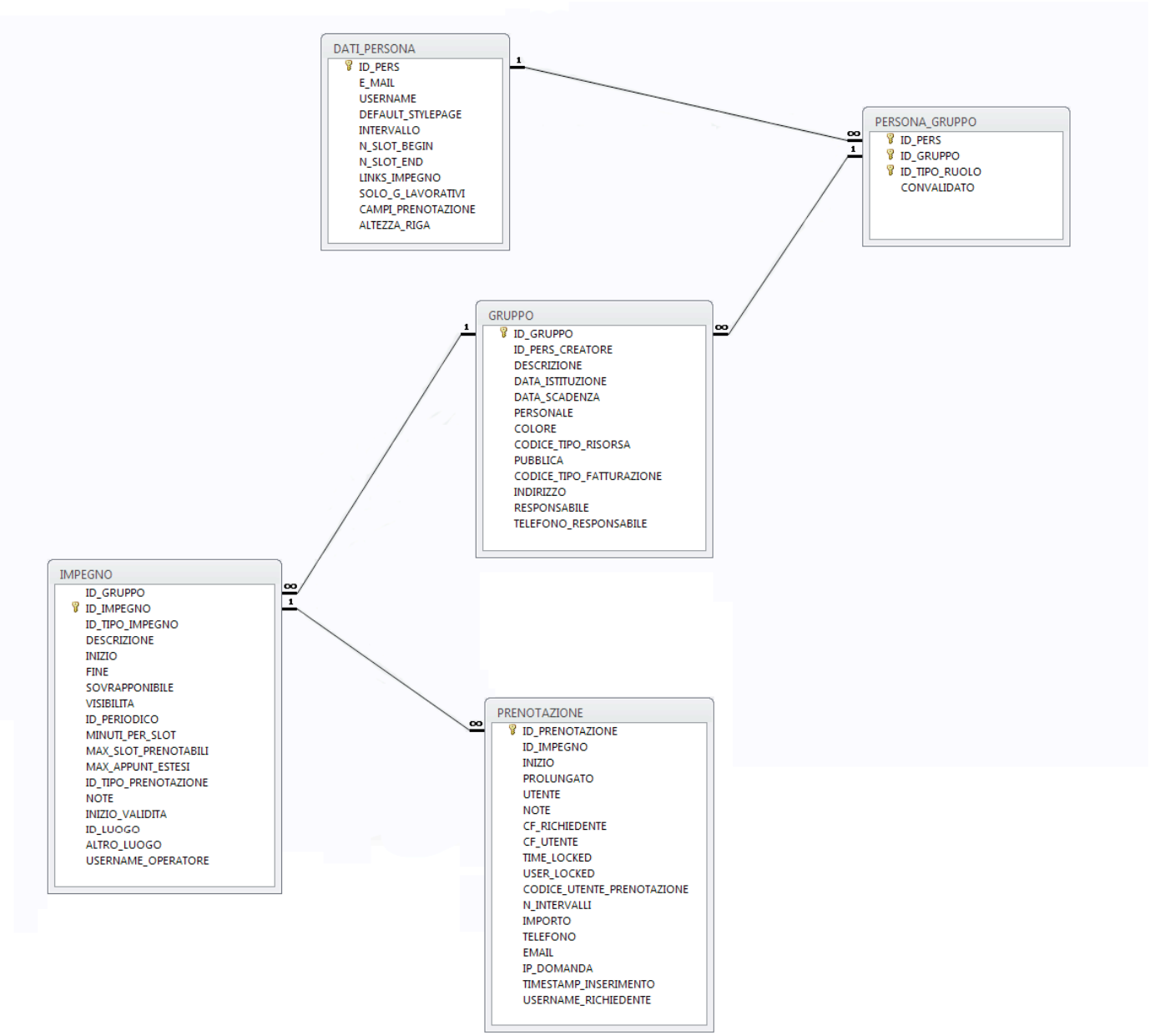


Figura 18 - Struttura relazioni tabelle per la prenotazione

FASE IMPLEMENTATIVA

3.1 AMBIENTI DI SVILUPPO

3.1.1 Scelta di iOS

Dal punto di vista dell'implementazione si è deciso di utilizzare la piattaforma iOS. Xcode è l'ambiente utilizzato per lo sviluppo dell'applicazione. Consiste in un pacchetto che include tutti gli strumenti necessari alla scrittura del codice sorgente, al debug e alla progettazione di un'interfaccia utente.

Il linguaggio di programmazione cui si è fatto riferimento è Objective-C. Come per il linguaggio C esistono principalmente due tipi di file: i file d'intestazione (.h), che includono tutte le dichiarazioni delle variabili d'istanza della classe, delle proprietà e delle firme dei metodi; e i file d'implementazione (.m), che contengono il codice di ogni metodo e il completamento delle proprietà dichiarate nell'interfaccia.

In Xcode ci sono molti modi per semplificare la stesura di un codice. Ad esempio integra la funzione di autocompletamento del codice, chiamato Code Sense, che permette di ottenere il nome completo di un metodo, una variabile o anche un intero costrutto battendo solamente alcuni caratteri. Lo scopo non è solo quello di risparmiare tempo ma anche quello di evitare errori di ortografia.

Xcode include Interface Builder, utilizzato per la realizzazione delle interfacce, e iOS Simulator per testare l'applicazione.

Gli aspetti fondamentali che hanno condizionato la scelta del sistema operativo iOS sono i seguenti:

- **compatibilità**: lo stile dell'applicazione è realizzato grazie ad un insieme di animazioni che la rendono molto facile da utilizzare. In ambiente iOS, questo è un compito relativamente semplice. Ciascun iPhone presenta le stesse dimensioni e risoluzione dello schermo e il rilascio di nuove versioni di OS non comparta problemi di compatibilità tra i dispostivi. Sviluppare per device Apple non è troppo difficile ed è molto semplice eseguire test. Android è disponibile su dispositivi diversi, i quali presentano distinte specifiche hardware, come ad esempio le dimensioni dello schermo, e versioni diverse del sistema operativo. L'ultima versione del software di Google, Jelly Bean, che ha cominciato a diffondersi molti mesi fa secondo i dati forniti da Google, è presente solo sul 10.2% dei dispositivi Android. Quel che è peggio è che Gingerbread, versione di Android rilasciata nel 2010, è ancora la più diffusa tra gli utenti, precisamente sul 47.4% dei device. Il grafico nella pagina seguente mostra in maniera esauriente la frammentazione del sistema operativo Android.

Version	Codename	API	Distribution
1.6	Donut	4	0.2%
2.1	Eclair	7	2.4%
2.2	Froyo	8	9.0%
2.3 - 2.3.2	Gingerbread	9	0.2%
2.3.3 - 2.3.7		10	47.4%
3.1	Honeycomb	12	0.4%
3.2		13	1.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	29.1%
4.1	Jelly Bean	16	9.0%
4.2		17	1.2%

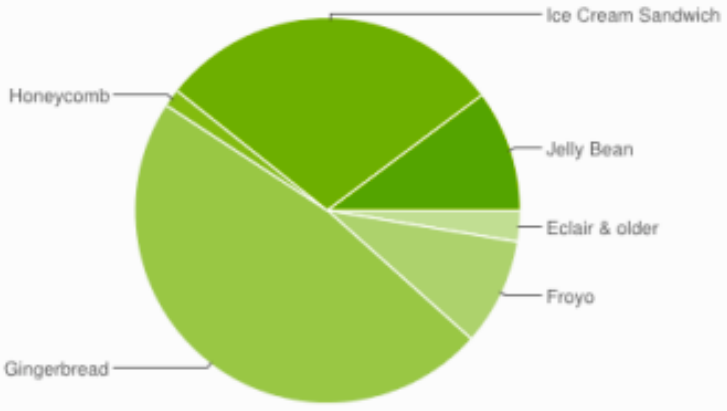


Figura 19 - Frammentazione Android

- **App Store**: per poter pubblicizzare l'applicazione e quindi ottenere visibilità è necessario il rilascio nell'Apple App Store. In uno dei recenti comunicati stampa Apple ha annunciato un dato che dimostra il fatto per cui le vendite di dispositivi mobile quali iPhone o iPad, si sono impennate negli ultimi anni. Dal debutto dell'App Store a oggi sono stati oltre 40 miliardi i download di applicazioni, di cui circa 20 miliardi solo nel 2012. Quaranta miliardi è un numero elevatissimo, specie se pensiamo che si tratta di download unici, cioè sono esclusi aggiornamenti e download multipli. L'App Store ha oltre 500 milioni di account attivi e in dicembre ha superato il precedente record con oltre 2 miliardi di download in un mese. La comunità ha creato oltre 775 mila applicazioni, e gli sviluppatori sono stati pagati da Apple per oltre 7 miliardi di dollari

- **interfaccia grafica**: un altra ragione per cui di è deciso di utilizzare iOS è la semplicità nel costruire interfacce grafiche, dovuta all'utilizzo di Interface Builder. Grazie a questo strumento è possibile archiviare tutte le informazioni necessarie alla creazione e alla formattazione delle interfacce in un unico file con estensione .nib. Tutti gli oggetti che lo sviluppatore può utilizzare per comporre l'interfaccia grafica della propria applicazione (finestre, menu, tabelle, viste ,bottoni, textbox, ecc..) sono contenuti all'interno dell'Object Library, che li raggruppa per categoria e ne semplifica l'utilizzo. Altri elementi importanti di IB sono l'Inspector, che permette di configurare i parametri di configurazione iniziale degli oggetti, e il Connection Panel, attraverso il quale è possibile definire le connessioni legate a un oggetto dell'interfaccia, semplicemente collegandolo, "tirando una linea", a un oggetto dichiarato nel codice e si può scegliere quale metodo di questo oggetto richiamare a seconda delle azioni effettuate sull'oggetto dell'interfaccia grafica.

3.1.2 Implementazione del Web Service

Per quanto riguarda il servizio web, necessario per interrogare il database, si è pensato di adottare java come linguaggio di programmazione e come approccio per la realizzazione REST utilizzando come framework di riferimento Jersey. Eclipse Ide è il software utilizzato per lo sviluppo del codice sorgente, mentre per il deployment il web service è stato installato come web application sotto Tomcat.

REST

REST, Representational State Transfer, è uno stile architetturale per sistemi software distribuiti. Il termine è stato introdotto e definito nel 2000 da Roy Fielding, uno dei principali autori delle specifiche del protocollo HTTP. Indica una serie di principi architetturali per la progettazione di Web Service. Concetto centrale per un sistema RESTful è quello di risorsa. Una risorsa è una qualunque entità che possa essere indirizzabile tramite Web, ciò è accessibile e trasferibile tra client e server. Spesso rappresenta un oggetto appartenente al dominio del problema che stiamo trattando. Durante un'interazione tra client e server quello che viene trasferito è una rappresentazione dello stato interno della risorsa. A differenza di altre specifiche per Web Service (es. SOAP) REST sfrutta a pieno la semantica e la ricchezza dei comandi HTTP e le sue funzionalità come, ad esempio, la negoziazione dei contenuti.

Richiesta	Significato
GET objects/3	recupera una rappresentazione dell'oggetto desiderato
POST objects/	crea un nuovo oggetto, la cui rappresentazione è nel body della richiesta
PUT objects/54	modifica l’oggetto esistente
DELETE objects/	rimuove l'intera collezione di oggetti

Perché un servizio web sia conforme alle specifiche REST deve avere alcune specifiche caratteristiche:

- architettura basata su client e server;
- stateless: ogni ciclo di request\response deve rappresentare un'interazione completa del client con il server. In questo modo non è necessario per mantenere informazioni sulla sessione utente, minimizzando l'uso di memoria del server e la sua complessità;
- uniformemente accessibile: Ogni risorsa deve avere un indirizzo univoco e ogni risorsa di ogni sistema presenta la stessa interfaccia, precisamente quella individuata dal protocollo HTTP.

Jersey

Jersey è l'implementazione di riferimento della specifica JAX-RS (Java API for RESTful Web Services) per la realizzazione di Web Service RESTful su piattaforma Java. Jersey permette di creare risorse semplicemente così come sviluppiamo POJO (Plain Old Java Object) utilizzando delle specifiche annotazioni per i metodi e le classi. In altre parole il framework si occupa di gestire le richieste HTTP e la negoziazione della rappresentazione e noi possiamo concentrarci alla soluzione del problema. Le principali annotazioni utilizzate nel servizio web tramite Jersey sono:

- @Path: indica la URI (Uniform Resource Identifier) a cui la risorsa sarà raggiungibile.
- @Produces indica il tipo MIME della risposta, che viene restituita come valore di ritorno del metodo;
- @Consumes indica il tipo MIME del request body, che viene passato come parametro al metodo
- @POST evidenzia il comando HTTP che il metodo è incaricato di gestire.

Eclipse

L'ambiente di sviluppo utilizzato per il web service Java realizzato nell'ambito del presente progetto è Eclipse. Eclipse può essere utilizzato per la produzione di software di vario genere, si passa infatti da un completo IDE per il linguaggio Java, a un ambiente di sviluppo per il linguaggio C++ e a plug-in che permettono di gestire XML, Javascript, PHP e persino di progettare graficamente un'interfaccia grafica per un'applicazione Java (Eclipse VE, "Visual Editor"), rendendo di fatto Eclipse un ambiente RAD. Il programma è scritto in linguaggio Java, ma anziché basare la sua interfaccia su Swing, il toolkit grafico di Sun Microsystems, si appoggia a SWT, librerie di nuova concezione che conferiscono ad Eclipse un'elevata reattività. La piattaforma di sviluppo è incentrata sull'uso di plug-in, delle componenti software ideate per uno specifico scopo, per esempio la generazione di diagrammi UML, ed in effetti tutta la piattaforma è un insieme di plug-in, versione base compresa, e chiunque può sviluppare e modificare tali plug-in. Nella versione base è possibile programmare in Java, usufruendo di comode funzioni di aiuto quali: completamento automatico (code completion), suggerimento dei tipi di parametri dei metodi e riscrittura automatica del codice (refactoring) in caso di cambiamenti nelle classi.

Apache Tomcat

Apache Tomcat (o semplicemente Tomcat) è un contenitore servlet open source sviluppato dalla Apache Software Foundation. Implementa le specifiche Java Server Pages (JSP) e Servlet di Sun Microsystems, fornendo quindi una piattaforma per l'esecuzione di applicazioni web sviluppate nel linguaggio Java. La sua distribuzione standard include anche le funzionalità di web server tradizionale, che corrispondono al prodotto Apache. In passato, Tomcat era gestito nel contesto del Jakarta Project, ed era pertanto identificato con il nome di Jakarta Tomcat; attualmente è oggetto di un progetto indipendente. Tomcat è rilasciato sotto la licenza Apache, ed è scritto interamente in Java; può quindi essere eseguito su qualsiasi architettura su cui sia installata una JVM.

3.2 METODI DEL WEB SERVICE

Ciascun metodo presenta la seguente inizializzazione:

```
@POST
  @Path("/methodName")
  @Produces(MediaType.APPLICATION_JSON)
  @Consumes(MediaType.APPLICATION_JSON )
  public Response mehtodContainer(ClassParameter object)
```

@POST indica che la gestione delle richieste avviene tramite il comando POST.

L'annotazione @Path("/methodName) associa la classe pubblica Response all'URI corrispondente al path di base:

```
http://localhost:8080/webserviceappcomune
```

Le annotazioni @Produces e @Consumes specificano che il tipo MIME utilizzato come formato per il request body e il response body è application_json poiché gli oggetti scambiati tra client e server sono del tipo Json.

L'ultima riga indica che il metodo POST è invocato con un parametro che rappresenta una classe, definita per ogni metodo, seguito da un parametro di tipo object. La classe contenente i vari parametri d'ingresso è definita attraverso il seguente schema XML:

```
@XmlRootElement
public class ClassParameter extends AppWsParameter {
  int parametro;
  public int getParametro() {
    return parametro;
  }
  public void setParametro(int parametro) {
    this.parametro = parametro;
  }
}
```

Tutti i metodi ritornano come risultato una stringa, convertita in Json attraverso un metodo appositamente creato denominato toJson, corrispondente a un esito positivo o negativo.

```
return Response.status(200).entity(result).build();
```

Nel primo caso si avrà come status “ok” e come message il contenuto della risposta:

```
result = pratiche.toJSON();

{
    “status”: “ok”,
    “message”:[
          “key1”:“value1”,
          “key2”:“value2”
    ]
}
```

Nel caso negativo si avrà come status “error” e come message la descrizione dell’errore generato:

```
result = errore.toJSON();

{
    “status”: “ok”,
    “message”:[
        “error”:“error description”
    ]
}
```

Di seguito sono descritte le soluzioni implementative considerate per le interrogazioni ipotizzate nella fase di progettazione e precedentemente descritte:

- Ottenimento dettagli sullo stato di una pratica anagrafica:

  query SQL senza SELECT intermedie filtrata in base al numero, la data convertita nel formato impostato e il tipo della pratica:

```
statoPraticaSql =
 SELECT NVL(an_desfas_ced, an_desfas) , an_datope_char
 FROM pratiche_anagrafiche
 WHERE an_numpra = ?
 AND an_datpra = TO_DATE(?, 'dd/mm/yyyy')
 AND an_codtipfas = ?;
```

- Ottenimento dati di una sezione elettorale:

  query SQL senza SELECT intermedie filtrata in base al codice fiscale convertito in lettere maiuscole:

```
sezioneElettoraleSql =
 SELECT SEZIONE , PLESSO , INDIRIZZO, LATITUDINE, LONGITUDINE
 FROM  sezioni_elettorali
 WHERE  CODICEFISCALE = UPPER(?);
```

- Ottenimento dei nomi dei tecnici:

  la query effettua una JOIN fra la tabella impegno e la tabella gruppo per selezionare i nomi dei tecnici che ricevono appuntamenti dalla data della richiesta “SYSDATE” in avanti e che abbiano un inizio validità antecedente a tale data. Il campo id_tipo_impegno è settato a 10 per questi tipi di prenotazioni:

```
nomeTecniciSql =
 SELECT DISTINCT g.id_gruppo, g.descrizione
 FROM impegno i , gruppo g
 WHERE i.id_gruppo = g.id_gruppo
 AND i.id_tipo_impegno = 10
 AND i.inizio >= SYSDATE
 AND NVL(i.inizio_validita, SYSDATE) <= SYSDATE
 ORDER BY g.descrizione;
```

- Ottenimento impegni per un determinato tecnico:

  La query prende come parametro l'id_gruppo e effettua una SELECT sulla tabella IMPEGNO per restituire l'id_impegno, la data d'inizio, la durata dell'impegno, il numero di appuntamenti previsti, il numero di quelli già occupati e il numero di quelli ancora disponibili. Per poter calcolare questi ultimi quattro campi è stata necessaria una inner join con una SELECT intermedia in cui si effettua una left outer join tra le tabelle IMPEGNO e PRENOTAZIONE.

  Ogni impegno è caratterizzato da un insieme di slot che va da 0 in avanti; uno slot corrisponde a una prenotazione, ciascuna da 30 minuti.

  Per esempio se l'impegno dura dalle 8:00 fino alle 10:00, vi saranno in totale 4 slot disponibili, ovvero 4 prenotazioni in quell'impegno.

  La durata dell'impegno è calcolata estraendo i minuti dalla differenza tra la data d'inizio impegno e quella di fine:

  ```
  round((i.fine-i.inizio)*24*60, 0)
  ```

  Il numero di appuntamenti previsti è calcolato dividendo la durata dell'impegno per i minuti previsti per ciascuna prenotazione, cioè 30:

```
round((i.fine-i.inizio)*24*60, 0) / COALESCE(i.minuti_per_slot, 30)
```

Per determinare il numero di appuntamenti occupati è calcolato guardando il numero di record nella tabella PRENOTAZIONE filtrata per l'id_impegno:

```
count(p.id_impegno)
```

Infine per calcolare il numero di appuntamenti disponibili è sufficiente sottrarre al numero di appuntamenti previsti il numero di quelli già prenotati.

```
round((i.fine-i.inizio)*24*60, 0) / COALESCE(i.minuti_per_slot, 30) -
count(p.id_impegno)
```

```
appuntamentiVisibiliSql =
 SELECT i.id_impegno , i.inizio,s,durata_impegno ,
 s.numero_appuntamenti , s.numero_appuntamenti_presi ,
 s.numero_appuntamenti_liberi
 FROM impegno i INNER JOIN
   (SELECT i.id_impegno ,
   round((i.fine-i.inizio)*24*60, 0) as durata_impegno ,
   round((i.fine-i.inizio)*24*60, 0) / COALESCE(i.minuti_per_slot,
   30) as numero_appuntamenti ,
   count(p.id_impegno) as numero_appuntamenti_presi,
   round((i.fine-i.inizio)*24*60, 0) / COALESCE(i.minuti_per_slot,
   30) - count(p.id_impegno) as numero_appuntamenti_liberi
   FROM impegno i
   LEFT OUTER JOIN prenotazione  p ON i.id_impegno = p.id_impegno
   WHERE i.id_gruppo = ? AND i.inizio >= SYSDATE
   AND NVL(i.inizio_validita, SYSDATE) <= SYSDATE
   AND i.id_tipo_impegno = 10;
 GROUP BY i.id_impegno, TO_CHAR(i.inizio, 'YYYY-MM-DD HH24:MI:SS'),
 TO_CHAR(i.fine, 'YYYY-MM-DD HH24:MI:SS'), round((i.fine-
 i.inizio)*24*60, 0), round((i.fine-i.inizio)*24*60, 0) /
 COALESCE(i.minuti_per_slot, 30)) s ON i.id_impegno = s.id_impegno
 WHERE i.id_gruppo = ? AND i.inizio >= SYSDATE
 AND NVL(i.inizio_validita, SYSDATE) <= SYSDATE
 AND i.id_tipo_impegno = 10;
```

- Ottenimento prenotazioni disponibili per un determinato giorno:

  per determinare le ore disponibili per un dato impegno sono necessarie due query:

  la prima permette di ottenere le informazioni relative ad un impegno, tra cui il più importante è il campo numero_appuntamenti;

  la seconda ottiene, per ciascun record, il numero dello slot che è stato occupato;

```
datiImpegnoSql =
 SELECT i.id_impegno,
 TO_CHAR(i.inizio, 'YYYY-MM-DD HH24:MI:SS') as inizio_ipegno,
 TO_CHAR(i.fine, 'YYYY-MM-DD HH24:MI:SS') as fine_impegno,
 round((i.fine-i.inizio)*24*60, 0) / COALESCE(i.minuti_per_slot, 30)
 as numero_appuntamenti,
 COALESCE(i.minuti_per_slot, 30) as minuti_per_slot
 FROM impegno i
 WHERE i.id_impegno = ?;


slotOccupatiSql =
 SELECT p.inizio as numero_slot
 FROM prenotazione p
 WHERE p.id_impegno = ?;
```

Di conseguenza, all'interno del metodo, saranno calcolate le ore disponibili per ogni slot che non risulta tra quelli già occupati:

```
cal.add(Calendar.MINUTE, (i * min_per_slot));

inizio_prenotazione = df.format(cal.getTime());

cal.add(Calendar.MINUTE, ((i + 1) * min_per_slot));

fine_prenotazione = df.format(cal.getTime());
```

- Inserimento di una prenotazione:

  la query inserisce il record relativo alla nuova prenotazione occupata nella tabella PRENOTAZIONE. Per l'inserimento del campo id_prenotazione è eseguita una SELECT per prendere il suo valore più grande nella tabella e sommarlo di un'unità.

```
InserisciAppuntamentoSql =
 INSERT INTO prenotazione
 (id_prenotazione, id_impegno , inizio, prolungato, utente, note,
 cf_richiedente, cf_utente, time_locked, user_locked,
 codice_utente_prenotazione, n_intervalli, importo, telefono, email,
 ip_domanda, timestamp_inserimento, username_richiedente )
 VALUES
 ((SELECT MAX(id_prenotazione) + 1 FROM prenotazione ),
 ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?);
```

3.3 ACQUISIZIONE DATI

Le richieste da parte del client seguono per tutti i tre servizi, escluse le news, lo stesso schema di acquisizione dati basato su una classe chiamata DataDownloader che consente di stabilire una connessione attraverso i comandi HTTP (in questo caso sono tutte in POST) verso il server remoto specificato attraverso l'annotazione ReadData e eseguendo la chiamata al metodo impostato nell'annotazione WebMethod. Di conseguenza devono essere definiti i parametri da passare come input tenendo presente che il tipo di dato inviato deve essere lo stesso che il metodo riceve in ingresso. Le informazioni ricevute dal web service in formato Json sono utilizzate come oggetti in un dizionario NSDictionary. Per ciascun oggetto nel dizionario si selezionano gli elementi necessari per l'utilizzo all'interno dell'applicazione attraverso l'annotazione objectForKey, che sono poi inseriti in stringhe o interi in base al tipo di dato. La struttura generica di ogni richiesta è definita nel modo seguente:

```
[[DataDownloader sharedDataDownloader]
     ReadData:@"http://localhost:8080/webserviceappcomune/rest/home"
    WebMethod:@"methodName" HTTPMethod:@"POST"
    Params:@{
    @"parametro1":[NSNumber numberWithInt: parametro1],
    @"parametro2":parametro2,
    @"parametro3": [NSNumber numberWithInt: parametro3]}
    CacheDuration:0 ToPath:@""
    Elabora:^BOOL(NSData *data, DataDownloaderProps *params) {

    if(!params.error) //se non ci sono stati errori nella richiesta
    {
         NSLog(@"%@",[[NSString alloc] initWithData:data
         encoding:NSUTF8StringEncoding]); //stampa tutti dati ricevuti

          NSDictionary *myvalues = [data objectFromJSONData][@"message"];
          for (id object in myvalues)
          {
             NSDictionary *currentObject = (NSDictionary*)object;
             NSString *dato = [currentObject valueForKey:@"dato"];
             [arrayDati addObject:dato];
          }
    }
    return  FALSE;
}];
```

3.4 CALENDARIO

Una parte fondamentale dell'applicazione è l'implementazione del calendario attraverso il quale l'utente sceglie la data e l'ora della prenotazione.Dato che Xcode non dispone di uno strumento per includere un calendario in stile Apple, l'opzione adottata è stata quella di integrare il progetto Kal Calendar. Questo progetto mira a fornire un'implementazione open-source dell'app calendario di Apple (MobileCal) fornendo una vista per mese. Quando l'utente tocca un giorno sul calendario, tutti i dati associati per quel giorno, che in questo caso sono le ore disponibili, saranno visualizzati in una tabella direttamente sotto il calendario. Per utilizzare Kal in modo efficiente è necessario: specificare quali sono i giorni che prevedono appuntamenti, che sarranno contrassegnati con un punto; realizzare una tabella UITableViewCells che mostra i dettagli (se presenti) per il giorno selezionato.

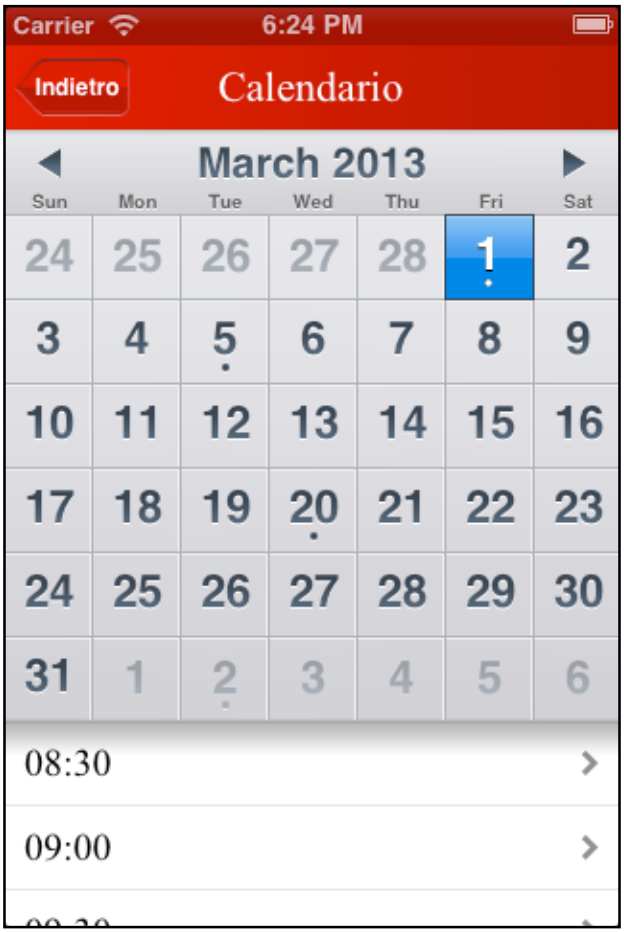


Figura 20 - Calendario

3.4 NEWS

La sezione delle news come descritto in precedenza fa utilizzo del feed RSS del Comune di Cesena per ricevere un elenco delle ultime notizie pubblicate. Al tocco da parte dell'utente su una voce dell'elenco si aprirà web view dell'url indicato dalla notizia selezionata.

Elementi necessari

All'interno del file d'implementazione NewsViewController.h sono definiti le variabili globali e i delegati per la creazione dei metodi che si riferiscono alla definizione e al popolamento della tabella che andrà a contenere l'elenco:

- parser XML, che si occuperà di leggere il feed RSS e di convertirne le notizie.

```
NSXMLParser *rssParser;
```

- NSMutableArray, ovvero una collezione di oggetti, in cui inseriremo i vari feed letti (con le varie informazioni, ovvero titolo, data, testo, etc)

```
NSMutableArray *elencoFeed;
```

- un dizionario per memorizzare ogni variabile temporanea per ogni elemento

```
NSMutableDictionary *item;
```

- le stringhe che rappresentano i valori dei campi letti dal feed

```
NSString *currentElement;
NSMutableString *currentTitle, *currentDate, *currentSummary, *currentLink;
```

- i delegati sono:

```
UITableViewDataSource,UITableViewDelegate,NSXMLParserDelegate.
```

Per il corretto popolamento della tabella il numero di righe deve essere pari al numero di elementi scaricati dal feed:

```
-(NSInteger)tableView:(UITableView*)tableView
numberOfRowsInSection:(NSInteger)section{
    return [elencoFeed count];

}
```

All'interno del metodo `-(UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath` che crea le singole celle della tabella sono inserite le istruzioni che vanno a ricavare gli elementi desiderati dalla lista (che coincidono con il numero di riga, ricavato da "indexPath.row"). Tali istruzioni estraggono quindi i campi "title" e "date" settandoli come testo della cella.

Azioni del parser

Nel metodo viewDidLoad, che inizializza la vista, è definita, una stringa "path", in cui è inserito l'indirizzo del feed RSS:

```
NSString *path =
@"http://www.comune.cesena.fc.it/flex/cm/pages/ServeFeed.php/L/IT/fmt/rss20
/feed/pages%3A4";
```

Dopodiché avviamo il parsing del feed chiamando il metodo "parseXMLFileAtURL" passando proprio l'indirizzo "path" come parametro:

```
[self parseXMLFileAtURL:path];
```

Tale metodo è definito inizializzando l'array "elencoFeed", per poi passare all'inizializzazione del parser XML e al suo avvio nel modo seguente:

```
- (void)parseXMLFileAtURL:(NSString *)URL{

    //inizializzo la lista degli elementi
    elencoFeed = [[NSMutableArray alloc] init];
    // dobbiamo convertire la stringa "URL" in un elemento "NSURL"
    NSURL *xmlURL = [NSURL URLWithString:URL];

    // inizializzo il parser XML
    rssParser = [[NSXMLParser alloc] initWithContentsOfURL:xmlURL];

    [rssParser setDelegate:self];

    // setto alcune proprietà
    [rssParser setShouldProcessNamespaces:NO];
    [rssParser setShouldReportNamespacePrefixes:NO];
    [rssParser setShouldResolveExternalEntities:NO];

    // avvio il parsing del feed RSS
    [rssParser parse];

}
```

Per funzionare correttamente, il parser ha bisogno di altri due metodi richiamati, rispettivamente, quando inizia e quando finisce un elemento XML. Nel primo caso è necessario re-inizializzare tutti gli elementi, in modo da poter leggere un nuovo elemento senza errori. Al contrario, quando un elemento XML termina, bisogna salvare tutti questi valori letti in un unico elemento ("item") e inserirlo nella lista dei feed letti ("elencoFeed").

```
-(void)parser:(NSXMLParser *)parser didStartElement:(NSString *)elementName
namespaceURI:(NSString *)namespaceURI qualifiedName:(NSString *)qName
attributes:(NSDictionary *)attributeDict{
    currentElement = [elementName copy];
    if([elementName isEqualToString:@"item"])
    {
        item = [[NSMutableDictionary alloc] init];
        currentTitle = [[NSMutableString alloc] init];
        currentDate = [[NSMutableString alloc] init];
        currentSummary = [[NSMutableString alloc] init];
        currentLink = [[NSMutableString alloc] init];
    }
}

-(void)parser:(NSXMLParser *)parser didEndElement:(NSString *)elementName
namespaceURI:(NSString *)namespaceURI qualifiedName:(NSString *)qName{
    if([elementName isEqualToString:@"item"])
    {
        [item setObject:currentTitle forKey:@"title"];
        [item setObject:currentLink forKey:@"link"];
        [item setObject:currentSummary forKey:@"summary"];
        [item setObject:currentDate forKey:@"date"];

        [elencoFeed addObject:[item copy]];
    }

}
```

Per completare la definizione del parser occorre descrivere altri due metodi.

Il primo viene richiamato ogni volta che viene letto un carattere all'interno del feed. A seconda dell'elemento considerato, si inserire il carattere letto in coda a quelli già letti dello stesso carattere, in modo da ricostruire l'informazione completa. Il secondo metodo, invece, è richiamato solo quando il parser completa la lettura del feed RSS, eseguendo il "reload" della tabella (verranno, quindi, richiamati i metodi per assegnare il numero di righe e per inserire il testo nelle varie celle).

```
-(void)parser:(NSXMLParser *)parser foundCharacters:(NSString *)string{

    //salva i caratteri per l'elemento corrente
    if([currentElement isEqualToString:@"title"])
    {
        [currentTitle appendString:string];
    } else if ([currentElement isEqualToString:@"link"])
    {
        [currentLink appendString:string];
    } else if ([currentElement isEqualToString:@"description"])
    {
        [currentSummary appendString:string];
    } else if ([currentElement isEqualToString:@"pubDate"])
    {
        [currentDate appendString:string];
    }

}


-(void)parserDidEndDocument:(NSXMLParser *)parser{
    [self.TableViewNews reloadData];
}
```

CONCLUSIONI

Il progetto è stato completamente finito e testato, la versione finale funziona perfettamente all'interno tutte le versioni di iPhone. E' stato un lungo cammino per vedere il progetto ultimato e funzionante. Il tempo totale trascorso in questo progetto è stato di circa quattro mesi, due per realizzare l'applicazione e due per completare il web service.

L'obiettivo iniziale era quello di realizzare un applicazione che consentisse l'accesso ai servizi del Comune di Cesena, già accessibili dal sito web, anche attraverso i dispostivi mobile.

All'interno dell'applicazione sono presenti solo alcuni dei servizi messi a disposizione, perciò vi sono numerose funzionalità che possono essere aggiunte per lavorare ad eventuali versioni future. In particolare si è già pensato di integrare il servizio di prenotazione presso l'Ufficio Anagrafe per il rilascio/rinnovo della Carta d'Identità elettronica.

Dato che la maggior parte delle applicazioni tende a usare una funzionalità GPS per determinare la propria posizione, un aggiornamento per la sezione sedi elettorali, che sarebbe utile al cittadino, potrebbe essere il tracciamento di un itinerario dalla propria posizione al plesso in cui è presente il seggio elettorale.

Un ulteriore miglioramento che può essere effettuato è dare la possibilità al cittadino di registrarsi inserendo i propri dati e di effettuare il login con nome utente e password per gli accessi successivi. In questo modo non ci sarà bisogno di cercare le informazioni poiché all'accesso l'applicazione scaricherà già i dati dell'utente loggato rendendola ancora più user-friendly.

BIBLIOGRAFIA

- Sistemi operativi in forte evoluzione - scenario Mobile in Italia
http://nielsen.com/it/it/news-insights/comunicati-stampa/2012/importanti-novita-nello-scenario-mobile-in-italia.html

- Global Apple iPhone sales from 3rd quarter 2007 to 1st quarter 2013
http://www.statista.com/statistics/12743/worldwide-apple-iphone-sales-since-3rd-quarter-2007

- Global annual shipments of smartphones using Android
http://www.statista.com/statistics/241947/global-shipment-forecast-of-smartphones-using-android-os/

-A Preview of Windows Phone 8 for developers
http://www.developergarden.com/en/blog/articles/article/a-preview-of-windows-phone-8-for-developers/

- Why do developers prefer iOS over Android?
http://thenextweb.com/apple/2012/03/06/why-do-developers-prefer-ios-over-android-try-75-adoption-of-ios-5-while-ics-is-stuck-at-1/

- Apple App Store: 40 miliardi di download
http://www.tomshw.it/cont/news/apple-app-store-40-miliardi-di-download-e-dicembre-boom/42158/1.html

- Perfect your app with Xcode tools for iOS developers
http://www.techrepublic.com/blog/ios-app-builder/perfect-your-app-with-xcode-tools-for-ios-developers/187

- RESTful Web Services – La Guida
http://www.html.it/guide/restful-web-services-la-guida/

- Eclipse: un editor per scrivere codice Java
http://www.mrwebmaster.it/java/guide/eclipse-editor-scrivere-codice-java_214.html