Università degli Studi di Padova

Dipartimento di Scienze Statistiche
Corso di Laurea Specialistica in

Scienze Statistiche e Tecnologie Informatiche

RELAZIONE FINALE

Sviluppo di applicazioni mobili portabili tramite il framework Apache Cordova

Relatore Prof. Mauro Migliardi
Dipartimento di Ingegneria

Laureando: Emanuele Zorzi
Matricola N° 1029416

Anno Accademico 2014/2015

INDICE

1 INTRODUZIONE

L’inizio degli anni duemila è stato caratterizzato dall’affermazione di Internet e dalla sua rapida espansione, una rivoluzione tecnologica e culturale per il fatto che permette la condivisione di informazioni in qualunque parte del mondo. Negli ultimi anni stiamo assistendo ad una seconda rivoluzione tecnologica, quella dei dispositivi mobili. Quelle che prima erano applicazioni esclusivamente eseguibili da PC, ora sono accessibili da dispositivi portatili sempre più potenti, di dimensioni tascabili. A questi “smartphone” che sono diventati veri e propri calcolatori portatili, dove la funzione di telefono è solo una delle tante funzioni disponibili, vanno aggiunti i “tablet”, che hanno caratteristiche hardware ben diverse come la grandezza o la durata della batteria, che li rendono più comodi per svolgere determinate funzioni che su smartphone diventano scomode, come la lettura o la visualizzazione di filmati.

Il mercato degli smartphone ha subito un rapido sviluppo che ha determinato una grande attenzione sia all'hardware che al software utilizzati da questi dispositivi mobili, portandoli al miglioramento delle prestazioni. Smartphone e tablet sono diventati di uso comune nella vita delle persone, quindi è diventato fondamentale migliorare questi dispositivi con applicazioni e software che risultino sempre più godibili e funzionali. Un programmatore di applicazioni quindi deve creare delle app che riescano sempre a sfruttare al meglio le potenzialità dei nuovi smartphone.

A complicare il lavoro dello sviluppatore che programma applicazioni per questa nuova tecnologia, si aggiunge la vastissima gamma di device disponibili sul mercato che si traduce nello sviluppo di una stessa applicazione per ogni tipologia di sistema operativo presente. Android, iOS e Windows Phone sono le principali piattaforme smartphone presenti nel mondo del mobile, ciascuna con la propria architettura e con le proprie caratteristiche, sulla base delle quali i programmatori si adattano nello sviluppo delle loro applicazioni. E’ di facile comprensione quanto possa risultare costoso e lungo lo sviluppo di un’applicazione che abbia lo scopo di essere eseguita in maniera efficiente sui vari dispositivi che un utente tipo può possedere.

Per questo motivo di solito i programmatori decidono di specializzarsi in uno specifico target di sviluppo (es. iOS) dividendosi il compito con altri che simultaneamente si dedicano ad altri sistemi. E` chiaro che questo metodo non sia altro che un ripiego vista l'apparente impossibilità di poter trovare una soluzione migliore che possa soddisfare le esigenze di tutti.

Il programmatore di applicazioni mobili, oltre ai problemi di sviluppo deve decidere come ideare e sviluppare l'applicazione. Per decidere il metodo di sviluppo il programmatore deve porsi delle domande: a chi è rivolta l'app, il budget disponibile, i requisiti che l'applicativo deve avere e come l'applicazione deve interagire con l'utente.

Esistono due metodologie di approccio, uno è rappresentato dall'interazione con una pagina web. L'altro approccio è la realizzazione di un applicativo realizzato in maniera specifica e precisa per quel tipo di dispositivo o sistema operativo.

Nei due casi rispettivamente si parla di Applicazione Web e di Applicazione Nativa; vediamo di seguito le differenze principali.

Un' applicazione Web è un' applicazione accessibile via web, nel caso di uno smartphone è possibile accedere tramite una connessione Wireless o con reti cellulari (2G/3G/4G) . Il suo contenuto è in genere dinamico e interattivo. Possono essere definiti web-app software come webmail, e-commerce, web forum, blog, giochi online e altro. L'ambiente di esecuzioni di queste applicazioni è il browser, quindi le applicazioni non sono direttamente in contatto son il sistema operativo ma solo con i servizi forniti dal browser.

Un'applicazione mobile nativa invece è un software realizzato appositamente per il suo sistema operativo, piattaforma. Ciò comporta l'utilizzo di un ambiente di lavoro per la programmazione adatta al sistema operativo. Oltre al giusto ambiente di lavoro, per la programmazione nativa è necessaria la conoscenza del linguaggio di programmazione adatto al sistema operativo, l'installazione di un SDK.

Attualmente il mercato di smartphone e tablet sembra sostanzialmente dividersi in due grandi universi a sè stanti che non hanno praticamente nulla in comune, quasi nessun punto d'incontro o possibilità di avvicinamento. Uno è il mondo Apple, quello dei dispositivi con sistema operativo iOS, mentre l'altro è quello di Google col rispettivo sistema operativo Android. L'approccio, nei due casi, è completamente opposto.

Uno, iOS, impone un sistema molto rigido e soggetto a una politica di chiusura, l'ideologia Apple è caratterizzata da un approccio conservativo. Non a caso nello sviluppo di applicazioni iOS si usa il linguaggio Objective C, poco conosciuto prima del boom di Apple. La realizzazione di queste applicazioni può avvenire solo tramite l'uso di X-Code, utilizzabile solamente con un computer Apple, e la distribuzione dell'applicativo può avvenire solo a fronte dell'acquisizione di una licenza di sviluppatore iOS.

Segue un' ideologia completamente diversa l'approccio di Google: lo sviluppo di un'applicazione Android si fonda sul linguaggio Java, linguaggio di grande diffusione in ogni ambiente, di conseguenza gestibile tramite qualsiasi ambiente di lavoro un esempio può essere l'ambiente di sviluppo Eclipse, disponibile per tutti i sistemi operativi, inclusi quelli Apple con un apposito SDK che permette il debugging sulla periferica e la pubblicazione sull'Android Market in maniera completamente gratuita.

Lo sviluppo esponenziale di questi dispositivi mobili e la loro enorme diffusione ha portato gran parte del mercato nella direzione del mobile business. Cresce in maniera costante il numero delle aziende che ricorrono ad applicazioni e piattaforme mobile per rimanere vicine ai loro clienti.

Il problema sembra essere ben noto e compreso da chi si occupa di dotare gli sviluppatori di strumenti informatici che permettano di ottimizzare il loro lavoro, evitare ridondanze o repliche di parti, essenziali e non, del codice. Nasce così l'esigenza di mettere lo sviluppatore in condizione di creare un codice che qualsiasi device sia in grado di comprendere e di utilizzare, indipendentemente dalla piattaforma, dal sistema operativo e dall'hardware di cui è dotato.

Il principio è quello del riutilizzo del codice secondo cui un algoritmo o una parte di programma siano scritti una sola volta e poi richiamati più volte all'occorrenza. I vantaggi sono molteplici e si concentrano in un unico concetto che è facile intuire: un codice unico dà la possibilità di "coprire" tutti i device, quindi di minimizzare i tempi di sviluppo di un applicativo, di aggiornare, modificare, e ottimizzare un applicazione in maniera uniforme e univoca. Viene così abbandonato anche se non risolto del tutto il problema di repliche di codice, di struttura o componenti, uno dei problemi informatici di più difficile gestione.

L'esigenza diventa quindi quella di avere un livello intermedio, un framework, che si occupi di offrire allo sviluppatore la possibilità di sviluppare in un linguaggio di programmazione universale e la possibilità di accedere alle risorse hardware dei vari dispositivi utilizzando un sistema univoco. Il compito di questo framework intermedio è dunque quello di "tradurre" le chiamate provenienti da un livello più elevato (quindi generale) dell'applicazione nei corrispettivi linguaggi nativi (quindi di livello più basso) e di conseguenza restituire i relativi risultati al chiamante garantendo allo sviluppatore uniformità di utilizzo di tali servizi per tutti i casi possibili di device, risorse hardware e sistemi operativi. Sarebbe però un errore, data appunto questa grande eterogeneità, pensare che il framework possa provvedere a tutto: non tutte le problematiche e i possibili casi possono essere gestiti dal framework.

Un esempio di framework dedicato a standardizzare le modalità di accesso delle applicazioni alle funzionalità fornite dal sistema nativo è Apache Cordova, che permette di utilizzare gli standard HTML 5 e CSS 3 per scrivere e disegnare la propria applicazione mobile fornendo anche la completa SDK Javascript per accedere alle risorse native del dispositivo. Si consiglia nei casi dove l'obiettivo è la fruizione sulla maggior parte di tablet e dispositivi mobile in circolazione (iOS, Android, Blackberry, Windows Phone, Bada e Symbian). Da non sottovalutare il fatto di poter utilizzare tutte le librerie Javascript esterne e di grande diffusione e utilità come jQuery, jQuery Mobile, ecc. Apache Cordova è sicuramente la migliore alternativa per il riutilizzo del codice; di contro, può causare problemi o difetti in termini di performance e fluidità.

Questo framework permette la realizzazione di applicazioni ibride, cioè coniuga i vantaggi delle web app con quelli delle app native, consentendo di utilizzare le tecnologie Web per sviluppare l'applicazione senza necessità di una connessione Internet costante e avendo accesso alle risorse locali del dispositivo. In linea di massima le app ibride si pongono a metà strada tra le app native e le web app, rappresentando un buon compromesso per la creazione di applicazioni mobili multipiattaforma con un supporto nativo.

In questa tesi andremo ad analizzare la realizzazione di un'applicazione ibrida realizzata con Apache Cordova considerandone vantaggi, svantaggi, e problematiche riscontrate.

2 PRESENTAZIONE AZIENDA

Spazio Verde, l'azienda dove ho svolto lo stage è un gruppo di consulenza e progettazione attivo nei settori agricolo, ambientale, energetico e culturale. Fondato a Padova nel 1981, dispone di uno staff interno di vasta esperienza che collabora con esperti in varie discipline, di livello nazionale ed internazionale. Supporta imprese pubbliche e private nell' organizzazione e nello sviluppo di progetti. Spazio Verde si occupa anche di organizzazione eventi, monitoraggio bandi, elaborazione progetti di massima ed esecutivi di impianti nei settori dell'agricoltura e dell'agroindustria. L'azienda ha anche un reparto IT che si occupa della progettazione e realizzazione di applicativi specialistici nei settori ambientale e territoriale con architettura client/server e web-based in ambiente Windows/Linux. Il reparto IT si occupa inoltre della realizzazione di opere multimediali, CD-ROM e banche dati per la divulgazione scientifica e la promozione di progetti, prodotti ed enti; fornisce la soluzione integrata di problemi relativi a calcoli numerici, modelli matematici, supporti cartografici ed elaborazioni statistiche.

Spazio Verde collabora con più di un centinaio di imprese vitivinicole in tutta Italia per la progettazione e la gestione di piani promozionali nei paesi terzi (USA, Cina, Giappone, Brasile, ecc.) utilizzando i fondi che l'Unione Europea mette a disposizione nell'ambito dell'Organizzazione Comune del Mercato vitivinicolo (OCM Vino). In tale settore l'azienda ha acquisito un know how specifico per la gestione tecnico-amministrativa dei programmi, investendo dal 2009 in formazione e creando professionalità specializzate di alto profilo.
Uno dei principali investimenti interni effettuati riguarda la realizzazione di GE.P.O., un software interattivo per la Gestione dei Programmi Ocm in grado di far dialogare le imprese e i fornitori di servizio, nella massima trasparenza ed efficienza, aiutando tutti gli attori coinvolti in ogni fase del progetto: progettazione, gestione tecnico-amministrativo-finanziaria, rendicontazione e assistenza nei controlli. GE.P.O., oltre a fornire in tempo reale lo stato di avanzamento dei progetti, dà la possibilità di inserire in apposite aree documentali tutti i giustificativi (amministrativi, tecnici, finanziari)

indispensabili per la rendicontazione. Con OCM-Vino, Spazio Verde Lavora in tutta Italia grazie al raggiungimento di standard operativi e procedurali collaudati con più di 100 imprese vitivinicole.

L'azienda offre un servizio "chiavi in mano" : progettazione, gestione tecnico-amministrativo-finanziaria, rendicontazione e assistenza nei controlli.

In pratica Spazio Verde in questo progetto fa da tramite tra gli enti pubblici e le aziende private che vogliono far conoscere il loro prodotto. Gli enti pubblici infatti finanziano eventi dove danno la possibilità alle aziende finanziate di far conoscere il loro prodotto; principalmente, si tratta di vino. Attraverso G.E.P.O., Spazio Verde riesce a gestire tutta l'organizzazione necessaria in questo contesto: essa comprende ad esempio la gestione dei finanziamenti, dei permessi richiesti dagli enti, degli eventi che hanno il fine di pubblicizzare i prodotti in giro per il mondo. Inoltre, grazie a G.E.P.O le aziende possono gestire e controllare i loro dati e progetti grazie ad un'interfaccia grafica. Un altro compito che riguarda più da vicino il mio stage è la raccolta di giustificativi da parte dei clienti che Spazio Verde effettua, sempre attraverso G.E.P.O. Infatti, per motivare i costi degli eventi organizzati, le aziende devono presentare dei giustificativi, che possono essere foto o video. Una volta inviati dai clienti a Spazio Verde, questi verranno selezionati dallo staff dell'azienda.

3 OBBIETTIVO DELLA TESI

L’obiettivo di questa tesi è l’analisi dell’applicazione progettata da me. Spazio Verde aveva come scopo dello stage la progettazione e realizzazione di un applicativo per smartphone (iPhone, iPad, Android, eventualmente Windows Phone) con l’obiettivo di raccolta giustificativi per motivare i costi degli eventi organizzati, le aziende devono presentare dei giustificativi, che possono essere foto o video. Tale applicativo andrà a integrare G.E.P.O., un applicativo on-line di loro realizzazione, sviluppato in PHP e MySql, per la rendicontazione dei progetti OCM Vino. Per la rendicontazione di tali progetti è necessaria anche la raccolta di giustificativi fotografici che accertino lo svolgimento degli eventi finanziati (fiere, degustazioni, ecc...). Per facilitare la raccolta delle foto da parte dei clienti di Spazio Verde (destinatari del contributo previsto dalla campagna OCM Vino) si è pensato di integrare l'applicativo on-line con una app per smartphone con accesso protetto da login che dia la possibilità di caricare i giustificativi associandoli ad un determinato evento già codificato in G.E.P.O. Gli eventi sono tutti in paesi non appartenenti alla comunità europea e in luoghi che spesso non hanno disponibilità di rete internet (ad esempio fiere, ristoranti e punti vendita), ciò implica eventuale difficoltà di gestire on-line in tempo reale l'operazione di aggiornamento del gestionale di rendicontazione. Dovrà quindi essere previsto un metodo di raccolta tale da poter eventualmente garantire il trasferimento di informazioni anche in modalità differita. Le Caratteristiche dell’ applicazione che Spazio Verde mi ha chiesto sono:

- Autenticazione e collegamento al database MySql di G.E.P.O

- Pretrattamento del materiale multimediale acquisito

- Upload delle foto in specifica cartella protetta

- Funzionamento off-line e successiva sincronizzazione

La progettazione prevede anche una preventiva valutazione della piattaforma migliore da utilizzare per lo sviluppo.

Queste erano le premesse del mio stage; in questa tesi descriverò ciò che è stato fatto, le problematiche incontrate e le decisioni prese. Prima, però, elencherò e presenterò i programmi utilizzati.

4 APACHE CORDOVA

Apache Cordova, spesso chiamato con il nome del suo predecessore “PhoneGap”, è un insieme di librerie dette API, che permettono ai programmatori di applicazioni mobili di accedere alle funzione native dei device di diverse famiglie. L’idea di Cordova è quella di concentrare gli sforzi degli sviluppatori sull’app piuttosto che perdere del tempo per crearla e poi adattarla ad ogni piattaforma.

Per realizzare un'applicazione con Cordova quindi si richiede la conoscenza di HTML, CSS e Javascript. Il progetto Open Source della Nitobi Software si occuperà quindi di tramutare l’applicazione web in una applicazione per dispositivi mobili, le pagine della nostra applicazione saranno di fatto delle vere e proprie pagine web.

E’ proprio lo standard web di HTML CSS e JavaScript che dà la certezza che l’applicativo sia a tutti gli effetti multipiattaforma e che il porting su dispositivi diversi non comporti, se non in minima parte, la necessità di modifiche al codice. Il porting verso le varie piattaforme viene fatto installando gli ambienti di sviluppo relativi (tale procedura è spiegata nel sito ufficiale) e compilando l’app realizzata. I requisiti per far funzionare le applicazioni create sono: installare correttamente i relativi SDK(Android SDK, X-Code, o altri) e gli strumenti per consentire la compilazione delle applicazioni. Le applicazioni generate con l’uso di Apache Cordova saranno compilate, come le applicazioni native, usando i relativi SDK.

Cordova, come anticipato, fornisce quindi un set di librerie Javascript che possono essere invocate. Ogni chiamata verrà poi tramutata dal framework nella relativa chiamata in codice nativo in base al device di utilizzo. Il risultato di questa operazione è un informazione che risulterà essere uniforme indipendentemente dalla piattaforma di utilizzo.

La possibilità di utilizzare Apache Cordova per e su vari sistemi operativi rende anche questo strumento particolarmente impreciso e non sempre sfrutta a pieno tutte le capacità del codice nativo per quella determinata macchina, questo deficit rende la realizzazione dell'interfaccia grafica molto scarna e difficile da personalizzare in maniera semplice e veloce.

Cordova è disponibile per le seguenti piattaforme: iOS, Android, Blackberry, Windows Phone, Palm WebOS, Bada, e Symbian.

Il pensiero di Cordova si fonda sull'idea di avere un solo codice; il principio di unicità del codice è inevitabile, ed è normale quindi che la fase ultima di deploy dell'applicazione debba essere effettuata su una piattaforma di sviluppo nativa. Una volta completato lo sviluppo basato su HTML, JavaScript e CSS, la fase finale prevede la creazione, sui vari linguaggi in base alla piattaforma, di un progetto che, sfruttando le librerie native di Cordova permetterà la traduzione al linguaggio nativo proprio della piattaforma selezionata con quello dell'applicativo scritto in linguaggio universale.

4.1 Installazione

Partire da zero con un framework completamente nuovo, come nel mio caso è stato con Apache Cordova non è stato semplice. Oltre ad averlo scelto tra varie alternative una problematica non trascurabile è stata capire l'utilizzo di queste librerie. Inizialmente pensavo che Cordova fosse un programma che mi permettesse di scrivere applicazioni per smartphone. Leggendo e informandomi in Internet invece ho capito che Cordova era un insieme di librerie che doveva essere installato nel Pc e che si appoggiava a un comune editor di testo per la scrittura del codice.
Nel sito di Cordova compare tutta la procedura di installazione che consente poi di iniziare a programmare l'app. Personalmente ho trovato difficoltà a settare tutte le impostazioni di cui Cordova aveva bisogno, quindi ho deciso di elencare ed esaminare i passaggi necessari ad utilizzarlo.
La procedura di installazione che andrò a descrivere è stata effettuata su una macchina che aveva come sistema operativo Windows 7, quindi i comandi del Prompt e le impostazioni che ho cambiato durante la procedura si riferiranno a Windows.
Per prima cosa tramite un qualsiasi browser bisogna collegarsi al sito di Cordova (il link è riportato qui sotto per semplicità).
http://cordova.apache.org/docs/en/4.0.0//guide_cli_index.md.html#The%20Command-Line%20Interface
Secondo le istruzioni che si trovano nel sito, ci sono dei prerequisiti da avere installati nel proprio computer.

Uno di questi prerequisiti è l'installazione di NodeJs

4.1.1 NodeJs

Node.js è un framework per realizzare applicazioni Web in JavaScript. La piattaforma è basata sul JavaScript Engine V8, che è il runtime di Google utilizzato anche da Chrome e disponibile sulle principali piattaforme. Ciò significa che molte delle pagine Web dinamiche, solitamente realizzate con linguaggi di programmazione come ad esempio PHP, JSP o ASP, possono essere realizzate usando una sorta di JavaScript arricchito da funzionalità server side.

I vantaggi e la comodità di Node.js sono molteplici: il vantaggio più evidente è la possibilità di realizzare applicazioni server-side senza dover imparare linguaggi di programmazione "tradizionali". Quasi ogni applicazione Web invece richiede almeno due tipi di competenze, se non addirittura tre. Partendo dal front-end abbiamo infatti grafici e web designers che si occupano della presentazione e dell'interfaccia grafica: di solito ciò richiede competenze di HTML, JavaScript, CSS. La logica dell'applicazione viene poi sviluppata usando linguaggi di programmazione "tradizionali", che possono essere PHP, Perl, Java, JSP, .NET ecc. Infine, in molti casi, è necessaria anche la figura di un esperto PL/SQL per ottimizzare l'accesso al database.

L'utilizzo di Node.js permette di unificare i primi due profili: le competenze di un web designer esperto di JavaScript possono essere applicate al framework, consentendo ad un'unica persona (o un unico team) di tenere d'occhio sia gli aspetti della presentazione, sia della programmazione vera e propria. Chiaramente ciò non vale per ogni scenario possibile, ma dipende dal contesto specifico del progetto. Considerando però che uno dei punti di forza di Node.js é la scalabilità del software, ne consegue che in linea teorica il framework possa essere applicato sia a piccoli progetti, sia ad applicazioni più grandi.

L' installazione di NodeJs è molto semplice: basta andare nel sito http://nodejs.org/ e scaricare il programma. Dopo averlo installato è necessario aggiungere il Path nelle variabili d'ambiente dell'utente e nella variabile di sistema.

Dopo aver installato NodeJs, continuando a seguire le istruzioni del sito bisogna aprire il prompt dei comandi e digitare i seguenti comandi aspettando che l'installazione vada a buon fine.

-npm install -g cordova

con questo comando si scaricano le librerie che Cordova mette a disposizione degli utenti;

-cordova create hello com.example.hello HelloWorld

Dopo aver eseguito questo comando è stato creato un progetto che può già essere eseguito; in questo progetto compare la classica scritta Hello Word.

Per eseguire il progetto bisogna entrare nella cartella del progetto con un semplice cd se ci si trova in Windows;

-cd hello

Successivamente Cordova per funzionare ha bisogno dell'installazione della Jdk che contiene la Java Virtual Machine cioè il runtime della piattaforma Java che esegue i programmi tradotti in bytecode dopo una prima compilazione. La Jdk si scarica semplicemente dal sito di Oracle. Bisogna però fare attenzione a scaricare la versione adatta al proprio sistema operativo. Dopo aver installato correttamente la Jdk vado sempre a inserire il suo path nelle variabile d'ambiente e in quella di sistema.

Dopo la Jdk bisogna installare l'Editor Eclipse e la sua sdk, cioè un software development kit che in informatica indica genericamente un insieme di strumenti per lo sviluppo e la documentazione di software. Eclipse si scarica dal suo sito ufficiale. Per avere la jdk è sufficiente scaricare il programma, sempre dallo stesso sito, e installarlo sul computer. Successivamente seguendo le istruzioni che si trovano nel sito si può installare la sdk. Durante l'installazione della sdk è possibile caricare più pacchetti: io consiglio per non perdere troppo tempo di caricare solo quelli che saranno utilizzati. Fatta l'installazione dell'editor, bisogna aggiungere il suo path alle due variabili d'ambiente.

Ora per funzionare Cordova ha bisogno di altri due programmi, o meglio di altre due librerie: la prima è ant la seconda è git.

4.1.2 Ant

Ant è una libreria JAVA sviluppata dalla Apache che permette di automatizzare il processo di sviluppo di applicazioni Java. Con Ant, infatti, è possibile creare un progetto che compila, genera la documentazione, e impacchetta l'applicativo in modo da renderlo installabile, ed effettua il deploy di un'applicazione web su un application server, tutto con il semplice lancio di un comando.

È consigliato, inoltre, aggiungere alla variabile di ambiente PATH la cartella bin di Ant e la directory bin della JDK in modo da poter richiamare il programma da qualsiasi directory. Ant è un programma che si lancia da riga di comando. I comandi che Ant esegue sono letti da un file XML, di solito chiamato build.xml. In questo file bisogna definire le operazioni (target) disponibili e, per ciascuna di esse, i comandi da eseguire (task).

4.1.3 Git

Git è un sistema software di controllo di versione distribuito, creato da Linus Torvalds nel 2005. La progettazione di Git è stata ispirata da BitKeeper e da Montone, pensato inizialmente solamente come motore a basso livello che altri potevano usare per scrivere un front-end. In seguito, Git è diventato un sistema di controllo versione, direttamente utilizzabile da riga di comando; vari progetti software (principalmente il kernel Linux) adesso usano Git per tale attività.

Fatta questa serie di operazioni Cordova è pronto per l'uso. Sempre nel sito ufficiale si trovano i comandi per arrivare all'esecuzione del progetto.

Per prima cosa bisogna aggiungere al progetto un sistema operativo su cui far girare la nostra App. I sistemi operativi installati possono essere Windows, Android, iOS, BlackBerry, ecc. Per comodità seguo l'esempio del sito e per il mio progetto carico Android. Per installare il sistema operativo per il mio progetto si deve eseguire il seguente comando:

-cordova platform add android

Ora il progetto è pronto per essere eseguito con il comando:

-cordova run android

Questi erano i passaggi che ho voluto esaminare per cominciare ad elaborare l' app in questione.

Quando si crea un progetto in Cordova, esso è diviso in cartelle. Le cartelle, o meglio i file che io sono andato a modificare si trovano nella cartella www. Il file index.html è la pagina che viene lanciata quando si testa l'applicazione. I file JavaScript e Css sono in sottocartelle.

4.2 Struttura Cartelle Progetto

Il progetto che viene creato con le istruzioni elencate prima crea una cartella che contiene dei file e delle sottocartelle:

- La cartella Hooks e il file config.xml contengono dei file di configurazione.
- La cartella Platform contiene i file che vengono caricati una volta che una piattaforma (ad esempio nel mio caso Android), viene caricata all'interno del progetto.
- La cartella Plugins, contiene le Api che vengono installate all'interno del progetto; l'installazione avviene tramite delle istruzioni che si trovano nel sito ufficiale di Cordova.
- La cartella www contiene i file modificabili. Nello specifico, si trova il file index.html, il file che viene lanciato quando testiamo l'applicazione. Nella cartella www si trovano inoltre le sottocartelle css, le quali contengono il file index.css, js che contiene il file JavaScript, e la cartella img, che contiene le immagini relative all'applicazione.

4.3 Funzionalità offerte

Le Api che si trovano nel sito ufficiale di Cordova prevedono praticamente tutte le interazioni che sono possibili con i device di ultima generazione.

Ecco una breve lista di tutte le classi di funzionalità che è possibile richiamare:

- **Accelerometer:** Cattura il movimento del dispositivo misurando la sua accelerazione lungo le direzioni x, y e z.

- **Camera:** Permette l'accesso alla fotocamera, dà la possibilità di salvare l'immagine come file oppure di ottenerla con codifica base64 (codifica utilizzata per la trasmissione tramite chiamate http). Dà, per la maggior parte delle piattaforme, anche la possibilità di accedere alle immagini già presenti nella libreria fotografica del dispositivo.

- **Capture:** Più generico del precedente oggetto Camera, permette l'acquisizione di file multimediali di tipo audio, immagine o video. E' possibile specificare diversi formati di acquisizione.

- **Connection:** Fornisce informazioni sulla connettività attuale: se non è presente, se è di tipo ETHERNET, WIFI, 2G, 3G o 4G.

- **Contacts:** Consente di accedere alle informazioni relative ai contatti, di crearne nuovi o modificarli.

- **Device:** Informazioni hardware e software del device in uso.

- **Events:** Permette di gestire diversi tipi di eventi: device pronto (framework cordova completamente caricato), pausa (quando l'applicazione va in background), online (rilevata la connettività), offline (rilevata mancanza di connettività), pressione tasto back (se presente), rilevato livello critico di batteria, basso livello batteria, rilevata pressione di bottoni hardware vari.

- **File:** Permette l'accesso al file system; è possibile leggere e scrivere dai file e cartelle, eseguirne l'upload online e ottenere i metadata.

- **Geolocation:** Permette l'accesso al sensore GPS quindi di rilevare la posizione corrente del device che corrisponderà alle coordinate fornite dal sensore (se presente) o calcolate in base alle specifiche W3C delle API di geolocalizzazione nel caso non fosse presente il sensore. Le coordinate vengono restituite sotto forma di latitudine e longitudine.

- **Globalization:** Permette di avere informazioni circa la lingue e le impostazioni internazionali correnti (lingua attuale, separatore migliaia, decimali, formato per la data ecc).

- **Notification:** Permette di eseguire azioni di notifica come Visualizzazione di messaggi (alert), vibrazione, suoni, utilizzando audio e grafica native del device utilizzato.
- **Storage:** Permette l'utilizzo dello storage del dispositivo basata sulla specifica W3C dei Database SQL Web.

Non tutte le Api sono disponibili per tutte le piattaforme: alcuni sistemi operativi non supportano alcune funzionalità messe a disposizione da Cordova. In questa immagine si potrà vedere quali Api sono disponibili per quali piattaforme.

	Android	BlackBerry 6	BlackBerry 10	iOS	Windows Phone 7	Windows Telefono 8	Windows 8	FirefoxOs Tizen
Accelerometro	✔	✔	✔	✔	✔	✔	✔	✔
Fotocamera	✔	✔	✔	✔	✔	✔	✔	✔
Cattura	✔	✔	✔	✔	✔	✔	✘	✘
Bussola	✔	✘	✔	✔ (3GS +)	✔	✔	✔	✔
Connessione	✔	✔	✔	✔	✔	✔	✔	✔
Contatti	✔	✔	✔	✔	✔	✔	✔	✘
Dispositivo	✔	✔	✔	✔	✔	✔	✔	✔
Eventi	✔	✔	✔	✔	✔	✔	✔	✔
File	✔	✔	✔	✔	nessun FileTransfer	nessun FileTransfer	✔	✘
Geolocalizzazione	✔	✔	✔	✔	✔	✔	✔	✔
Globalizzazione	✔	✔	✘	✔	✘	✔	✘	✘
InAppBrowser	✔	✔	✔	✔	✔	✔	✘	✘
Media	✔	✘	✔	✔	✔	✔	✔	✔
Notifica	✔	✔	✔	✔	✔	✔	✔	✔
Splashscreen	✔	✘	✔	✔	✔	✔	✔	✘
Archiviazione	✔	✔	✔	✔	localStorage solo	localStorage solo	✔	✔

Figura 1- Compatibilità delle funzionalità di Apache Cordova

4.4 Vantaggi e Svantaggi

A dispetto dei vantaggi apportati dal framework, derivanti dalla possibilità di usufruire di un solo linguaggio di programmazione universale basato su HTML, CSS e Java Script, bisogna evidenziarne anche i difetti.
Essi sono essenzialmente due:

• L'impossibilità di utilizzare un debugger.

• Vincoli legati alle risorse di sistema utilizzabili (processore, ram).

Per quanto riguarda il primo punto, si può pensare che a livello concettuale non sia possibile utilizzare le funzioni di debugging dell'ambiente di sviluppo dell'app.
L'applicazione per il compilatore non è altro che una semplice pagina Web. Ci sono dei programmi di debugging che fanno includere nel proprio codice HTML un riferimento a un codice JavaScript situato online, che abbia un riferimento del progetto stesso, con la possibilità di verificare i problemi collegandosi a un sito web. Personalmente non l'ho usato, anche perché questo tipo di debugging lo utilizza la funzione "Strumenti per sviluppatori", vale a dire la schermata di debugger, di Google Chrome. Sebbene l'idea sia ottima, questo tipo di debugger risolve la sola visione del codice HTML in tempo reale e del codice CSS applicato. Rimane quindi irrisolta la problematica relativa al debug del codice Javascript e la gestione delle chiamate http a server remoto.
Il secondo problema è quello riguardante i limiti delle risorse hardware utilizzabili. Non trattandosi di una vera applicazione ma di un applicativo creato all'interno del mondo Cordova, non è possibile usufruire appieno delle capacità hardware del device utilizzato. Questa limitazione si presenta in particolar modo in fase di allocazione della memoria RAM, cioè le applicazioni fatte con questo framework risentono di questa limitazione ad esempio quando si utilizzano operazioni con alto costo computazionale come può essere una interrogazione ad un database interno .
Anche sotto il punto di vista della grafica non si possono ottenere gli stessi risultati ottenuti da una vera applicazione scritta con il linguaggio nativo. Sempre a causa degli stessi limiti si potrà notare come alcune operazioni come lo scroll, o altre azioni tipiche di un dispositivo mobile, potranno risultare meno fluide rispetto alle corrispettive versioni native del prodotto.

Quindi gli svantaggi principali sono:

- una minore efficienza nel rendering grafico;
- la potenziale lentezza di esecuzione nell'accesso alle risorse locali;
- senza particolari accorgimenti l'aspetto dell'interfaccia grafica potrebbe non risultare abbastanza omogeneo con quello nativo della piattaforma.

I vantaggi di Apache Cordova sono stati già elencati sopra: già il fatto di essere un framework che permette di scrivere applicazioni per le varie piattaforme senza conoscere il linguaggio specifico è un vantaggio più che sufficiente per un programmatore che si vuole avvicinare al mondo delle applicazioni per mobile. Un altro vantaggio è che per realizzare un'applicazione con Apache Cordova serve conoscere l'HTML, il linguaggio più conosciuto per la realizzazione di pagine Web. Lo stesso vale per il CSS, mentre il Java Script è meno conosciuto, ma si basa su Java, di conseguenza non è difficile impararne le basi. Un ulteriore vantaggio è quello di avere un unico codice per tutte le piattaforme, e l'unicità del codice, come si è già detto, è molto importante. Apache Cordova gestisce poi tutti i porting verso le altre piattaforme: questo secondo me è un passaggio fondamentale, molto utile ad un programmatore che inizia ad avvicinarsi a questo mondo. Altro aspetto molto utile è infine la possibilità di accedere alle funzioni messe a disposizione da Apache Cordova, cioè le Api. Le Api disponibili sono molte e contengono le più importanti funzionalità che un dispositivo mobile può avere.

5 LINGUAGGI DI PROGRAMMAZIONE E DI MARKUP

5.1 HTML

L'HTML5 è un linguaggio di markup di pubblico dominio per la strutturazione delle pagine web, da Ottobre 2014 pubblicato come W3C Recommendation.

L'HTML è stato sviluppato verso la fine degli anni ottanta da Tim Berners-Lee al CERN di Ginevra assieme al noto protocollo HTTP che supporta invece il trasferimento di documenti in tale formato.

Nel corso degli anni, seguendo lo sviluppo di Internet, l'HTML ha subito molti aggiornamenti, revisioni e miglioramenti. Dopo un periodo di sospensione, in cui il W3C si è focalizzato soprattutto sulle definizioni di XHTML (applicazione a HTML di regole e sintassi in stile XML) e dei fogli di stile (CSS), nel 2007 è ricominciata l'attività di specifica con la definizione di HTML5.

Una caratteristica dell' HTML è che esso è stato concepito per definire il contenuto logico e non l'aspetto finale del documento. I dispositivi che possono accedere ad un documento HTML sono molteplici e non sempre dotati di potenti capacità grafiche. Proprio per questo gli sviluppatori di HTML hanno optato per un linguaggio che descrivesse dal punto di vista logico, piuttosto che grafico, il contenuto dei documenti. Questo significa che non esiste alcuna garanzia che uno stesso documento venga visualizzato in egual modo su due dispositivi. Se da una parte questo ha imposto in passato dei forti limiti agli sviluppatori di pagine Web, ha dall'altro garantito la massima diffusione di Internet.

Attualmente i documenti HTML sono in grado di incorporare molte tecnologie, che offrono la possibilità di aggiungere al documento ipertestuale controlli più sofisticati sulla resa grafica, interazioni dinamiche con l'utente, animazioni interattive e contenuti multimediali. Si tratta di linguaggi come CSS, JavaScript e jQuery, XML, JSON, o di altre applicazioni multimediali di animazione vettoriale o di streaming audio o video. HTML soffre di limiti propri di un sistema di markup ideato per scopi molto lontani da quelli attualmente richiesti dal Web Design. L'HTML è stato progettato per gestire singole pagine. Anche se ciò ha generato la definizione di un linguaggio semplice e diretto, esso non è più aderente alle necessità di siti spesso composti da migliaia di

pagine. A questo si affianca il problema incontrato dai grafici professionisti riguardo la difficoltà di posizionamento all'interno del documento di immagini e testo; questi limiti risultano fastidiosi e spesso immobilizzanti, per i professionisti.

Posizionare un'immagine, creare una banda laterale, giustificare un testo in HTML diventa un problema risolvibile esclusivamente con strumenti nati per tutt'altro scopo.

A queste problematiche HTML 4.0 cerca di far fronte adottando i fogli di stile, i quali possono essere inseriti all'interno di un documento iperstestuale in tre modi:

1. attraverso un file esterno;

2. attraverso l'intestazione generale di ogni singolo documento;

3. attraverso la specifica degli attributi nel singolo TAG.

Sfruttando i fogli di stile è possibile modificare la struttura di diverse pagine agendo su un unico file esterno che ne determina l'aspetto. La marcatura HTML dovrà quindi tornare alla sua originaria funzione di definizione della struttura logica del documento, lasciando ai fogli di stile la gestione del layout di pagina.

In realtà Apache Cordova dà la possibilità di scrivere in HTML5. HTML5 si basa però su HTML4, che io conosco e ho studiato. Nell’applicazione che ho scritto, quindi, non ho utilizzato funzionalità di HTML5, che non ho di conseguenza inserito nei linguaggi utilizzati.

5.2 CSS

Il Cascading Style Sheets (CSS) è un linguaggio che definisce lo stile, la formattazione e l'aspetto di un documento scritto in un linguaggio di markup. E' più comunemente usato assieme alle pagine web scritte in HTML o XHTML, ma può anche essere applicato ad ogni tipo di documento XML.

Il linguaggio definisce i comportamenti visivi di una pagina web. Ad ogni elemento della pagina viene associata una serie di istruzioni che modificano l'aspetto dell'elemento stesso, e il suo comportamento all'interno della pagina. L'intero codice va a creare uno script che viene associato alla pagina stessa e definisce l'aspetto degli elementi in base ad alcune azioni dell'utente (spostamenti del mouse sopra elementi della pagina, click di alcuni elementi o altro).

Inoltre esso definisce il posizionamento, la colorazione, ed il comportamento del testo, degli elementi e delle immagini.

Mentre l'autore di un documento tipicamente associa il proprio documento ad uno specifico CSS, i lettori possono utilizzare un foglio di stile proprio e sovrascrivere quello che il proprietario ha specificato. Le specifiche del CSS sono aggiornate dal W3C.

5.3 JavaScript

Javascript è un linguaggio di programmazione orientato agli oggetti e agli eventi. L'obiettivo del linguaggio è quello di consentire l'inserimento di contenuto eseguibile in una pagina web. Significa che una pagina web non resta solo HTML statico ma può includere programmi dinamici che interagiscono con l'utente, controllano il browser e creano dinamicamente contenuti HTML. Il Javascript è utilizzato soprattutto per la programmazione Web lato client per la creazione, di siti e applicazioni con effetti interattivi e dinamici create tramite funzioni script, invocate da eventi innescati dall'utente che sta utilizzando la pagina web; ciò significa che JavaScript fa in modo che si possa interagire con il CSS per modificare l'esperienza dell'utente durante la navigazione. Durante la navigazione può, dinamicamente, nascondere o mostrare ed alterare le proprietà o gli attributi associati a particolari elementi della pagina.

Fu originariamente sviluppato da Brendan Eich della Netscape Communications con il nome di Mocha e successivamente di LiveScript, successivamente è stato rinominato "JavaScript" ed è stato formalizzato con una sintassi più vicina a quella del linguaggio Java. JavaScript è stato standardizzato per la prima volta tra il 1997 e il 1999 dalla ECMA con il nome ECMAScript. È anche uno standard ISO.

A differenza di altri linguaggi, quali il C o il C++, JavaScript viene utilizzato soprattutto in quanto linguaggio di scripting, integrato, quindi, all'interno di un altro programma.

L'idea di base è che il programma ospite (che contiene il codice dello script) fornisca allo script un'API ben definita, che permette l'accesso ad operazioni specifiche, la cui implementazione è a carico del programma ospite stesso. Lo script, quando eseguito, utilizza riferimenti a questa API per richiedere (al programma ospite) l'esecuzione di operazioni specifiche, non previste dai costrutti del linguaggio JavaScript in sé. In effetti, questo è esattamente lo stesso meccanismo che viene adottato anche in un

linguaggio quale il C o il Java, nel quale il programma si affida a delle librerie, non previste dal linguaggio in sé, che consentono di effettuare operazioni quali l'I/O o l'esecuzione di chiamate a funzioni di sistema.
Uno degli usi più comuni del Javascript in ambito Web è la scrittura di piccole funzioni integrate nelle pagine HTML che interagiscono con il DOM del browser per compiere determinate azioni non possibili con il solo HTML statico: ad esempio controllare i valori nei campi di input, nascondere o visualizzare determinati elementi, ecc.

5.4 jQuery

jQuery è una libreria di funzioni Javascript per applicazioni web, che ha come obiettivo quello di semplificare la gestione degli eventi e l'animazione delle pagine HTML. È un software liberamente distribuibile e gratuito. Pubblicato per la prima volta nel gennaio 2006 da John Resig, è un progetto tuttora attivo e in continua evoluzione, gestito da un gruppo di sviluppatori. La sintassi di jQuery è studiata per semplificare la navigazione dei documenti e la selezione degli elementi DOM, creare animazioni, gestire eventi e implementare funzionalità AJAX. Due caratteristiche fondamentali di jQuery, sono la brevità del codice utilizzato, ma soprattutto il fatto che l'elemento da ricercare si possa passare alla funzione sotto forma di selettore CSS. Per chi non ha dimestichezza con framework del genere, o proviene da altre librerie, inizialmente questa caratteristica potrebbe sembrare uno svantaggio e magari produrrà qualche errore: per esempio in jQuery la funzione $('mioLink') cercherà un immaginario tag mioLink senza trovarlo. In realtà uno dei punti di forza del framework è proprio il potente e performante motore di selezione (di cui parleremo più avanti). Un'altra caratteristica che va sottolineata è che molti metodi di jQuery sono concatenabili, per rendere la scrittura e la lettura del codice molto più lineari. Oltre ai vantaggi già citati, vi sono alcuni buoni motivi per usare jQuery: anzitutto è possibile usarla in tutti progetti senza incappare in incompatibilità nel codice. jQuery ha un semplice sistema di estensione che permette di aggiungere nuove funzionalità (plugin) oltre a quelle predefinite, inoltre la sua diffusione ha fatto in modo che attorno al team di sviluppo ufficiale crescesse una numerosa community che mette a disposizione plugin e supporto di ottimo livello. Infine, per quanto possa sembrare un aspetto trascurabile, la sua sintassi sintetica ed efficiente è particolarmente apprezzabile quando si tratta di scrivere svariate linee di codice.

5.5 AJAX

Ajax è una tecnica di interazione più evoluta rispetto al classico paradigma client–server che consente un approccio di sviluppo web basato su JavaScript per la creazione di soluzioni web interattive. Consente alle pagine web di funzionare quasi come applicazioni desktop, superando la staticità delle pagine.

L'intento di tale tecnica è quello di ottenere pagine web che rispondono in maniera più rapida, grazie allo scambio in *background* di piccoli pacchetti di dati con il server, così che l'intera pagina web non debba essere ricaricata ogni volta che l'utente effettua una modifica. Questa tecnica riesce, quindi, a migliorare l'interattività, la velocità e l'usabilità di una pagina web.

Come detto precedentemente, la tecnologia Ajax si basa su uno scambio di dati in background fra browser e server in modo asincrono, che consente l'aggiornamento dinamico della pagina senza un esplicito refresh dell'utente. Ajax è definito asincrono perché i dati extra richiesti al server e caricati in background non interferiscono con la pagina esistente.

Il linguaggio standard per effettuare richieste al server è JavaScript. In genere viene usato XML come formato di scambio dei dati, anche se di fatto è possibile utilizzarne altri, come semplice testo, HTML preformattato oppure oggetti JSON.

Attraverso la combinazione di Ajax e la manipolazione del DOM13 è possibile aggiornare anche solo una sezione di una pagina web a seguito dell'effettuazione di un comando dall'utente, così da non dover ricaricare interamente la pagina in seguito ad ogni interazione con la pagina stessa, portando grandi vantaggi in termini di efficienza e velocità.

5.6 JSON

JSON, acronimo di JavaScript Object Notation, è un formato adatto all'interscambio di dati fra applicazioni client-server. È basato sul linguaggio JavaScript. Viene usato in AJAX come alternativa a XML/XSLT.

La semplicità di JSON ne ha decretato un rapido utilizzo specialmente nella programmazione in AJAX. Il suo uso tramite JavaScript è particolarmente semplice, infatti l'interprete è in grado di eseguirne il parsing tramite una semplice chiamata alla funzione eval(). Questo fatto lo ha reso velocemente molto popolare a causa della diffusione della programmazione in JavaScript nel mondo del Web.

I tipi di dati supportati da questo formato sono:

- booleani (`true` e `false`);
- interi, reali, virgola mobile;
- stringhe racchiuse da doppi apici (");
- array (sequenze ordinate di valori, separati da virgole e racchiusi in parentesi quadre `[]`);
- array associativi (sequenze coppie chiave-valore separate da virgole racchiuse in parentesi graffe);
- `null`.

6 MINI-APP

Quando mi è stato proposto il progetto, una delle cose più importanti da decidere era il modo di creare l'applicazione. Programmare l'applicazione per le piattaforme più vendute nel mercato, non è sembrata un'opzione sensata , perché significava conoscere in modo approfondito i linguaggi utilizzati dai vari sistemi operativi, quindi ho iniziato a cercare un "qualcosa" che potesse colmare la mia carenza nei linguaggi specifici e nella programmazione nativa. Cercando nella rete mi sono subito imbattuto in Apache Cordova, che, come ho già spiegato, permette di scrivere un'applicazione utilizzando solo HTML, CSS e JavaScript. L'HTML e CSS sono due linguaggi conosciuti e utilizzati dai programmatori, mentre il JavaScript è conosciuto meno ma non è di difficile comprensione e utilizzo, dato i principi di Java su cui si basa. Trovato il modo di creare la mia App, mi sono concentrato su ciò che il progetto richiedeva.

Nella documentazione nel sito di Apache Cordova ci sono degli esempi che ti permettono di iniziare a programmare, più precisamente ti mettono a disposizione dei "Plugin Apis", come si può vedere nel menù laterale nel sito ufficiale di Cordova.

Con Application Programming Interface (in acronimo API), nel mondo informatico, si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per la realizzazione di un determinato compito all'interno di un certo programma. Spesso con tale termine si intendono le librerie software disponibili in un certo linguaggio di programmazione.

Le API permettono infatti di evitare ai programmatori di riscrivere ogni volta tutte le funzioni necessarie al programma dal nulla, ovvero dal basso livello, rientrando quindi nel più vasto concetto di riuso di codice.

Sul sito la funzione di ogni Api è definita nei dettagli, sono spiegate le funzioni di alcuni parametri e la loro disponibilità sulle piattaforme e sistemi, inoltre è spiegata la procedura per poter installare le Api all'interno delle applicazioni.

Per installare un'Api basta consultare il sito ufficiale e seguire le istruzioni, solitamente, l'installazione si fa tramite il prompt dei comandi con un'istruzione che installa delle librerie nelle cartelle del progetto.

6.1 Photo

Dalla mia esperienza nella programmazione ho imparato a risolvere i problemi che mi sottoponevano con una strategia top-down e bottom-up, di conseguenza inizialmente ho cercato di risolvere i problemi singolarmente, rendendo più facile la creazione di piccole e semplici. L'approccio alla creazione dell'applicativo finale è stato questo: ho individuato le funzionalità principali, le ho studiate e realizzate singolarmente, per poi andare ad unirle creando così l'applicazione finale. Una delle funzionalità principali che mi veniva richiesta era scattare una foto con un device. Le Api mostrate nel sito sono state elencate quando ho presentato Apache Cordova; una di queste permette di scattare foto. Quindi inizialmente ho utilizzato il codice messo a disposizione nel sito e l'ho adattato alle richieste fatte dall'azienda, e al dispositivo su cui potevo testare.

```
var pictureSource;   // fonte dell'immagine
var destinationType; // imposta il formato del valore restituito
var i=0; //var per le fotoFotocamera
var j=0; //var per fotoGalleria

// Attendo che le librerie Api siano cariche
document.addEventListener("deviceready",onDeviceReady,false);

// dispositivo pronto
function onDeviceReady() {
    pictureSource=navigator.camera.PictureSourceType;
    destinationType=navigator.camera.DestinationType;
}

// Metodo che chiamo quando faccio la foto in modo corretto!
function onPhotoDataSuccess(imageData) {

    var div = document.getElementById("immaginiFotocamera");
    div.innerHTML +="<img id='immF"+(i)+"' src=\"./img\" onclick=\"window.open('imageData', '_system', ' ');\"/>";
    var immF = document.getElementById("immF"+i);
    immF.style.display = 'block';
    immF.src = "" + imageData;
    alert(imageData);
}

// Metodo che uso quando prendo dalla galleria una foto con successo
function onPhotoURISuccess(imageURI) {

    var div = document.getElementById("immaginiGalleria");
    div.innerHTML +="<img id='immG"+(j)+"' src=\"\" ></img>";
    var immG = document.getElementById("immG"+j);
    immG.style.display = 'block';
    immG.src = imageURI;
    alert(imageURI);
}

// Il pulsante capturePhoto chiama la funzione
function capturePhoto() {
// Scatta la foto utilizzando la fotocamera del dispositivo e recupera l'immagine come stringa con codifica base64
    navigator.camera.getPicture(onPhotoDataSuccess, onFail, {
        quality: 10,
        destinationType: Camera.DestinationType.FILE_URI,
        saveToPhotoAlbum: true
        });

    i=i+1;
}
```

Figura 2- App che scatta o sceglie una foto

Le Api che Cordova mette a disposizione sono dei file scritti in JavaScript. In questo caso la mia "miniApp" si strutturava in questo modo:

-Nel file index.html erano presenti due bottoni: uno che permetteva di accedere alla galleria del dispositivo e uno che accedeva alla fotocamera del dispositivo.

Nella file HTML è presente anche una sezione per mostrare a video le foto che si scattano e si scelgono.

-Nel file JavaScript invece si possono invocare delle funzioni messe a disposizione dalle librerie di Cordova per potere accedere alla galleria e alla fotocamera.

-Il File CSS, premette di dare uno stile alla pagina HTML.

La cosa interessante dell'Api messa a disposizione nella documentazione, è la facilità con cui grazie ad Apache Cordova è possibile accedere alle due funzionalità. Il pulsante adibito alla galleria richiama una funzione nel file JavaScript chiamata getPhoto().

Questi metodi, richiamano delle funzioni all'interno del file in base alla presenza o meno di errori ,e inoltre danno dei parametri, tra cui la qualità, e il formato della foto.

Se l'esecuzione del metodo procede senza errori verrà chiamata la funzione onPhotoURISuccess() che mi ritorna la foto e la stampa a video.

Nello stesso modo CapturePhoto() in base alla presenza di errori richiama diverse funzioni, nella chiamata al metodo ci sono anche dei parametri che vengono passati, tra questi i parametri per la qualità e il formato e un campo booleano che da la possibilità di salvare la foto nella galleria del device.

In questa piccola applicazione dove è possibile accedere alle fotografie della galleria e scattare una foto con la fotocamera, l'unico problema grosso è stato capire quali parametri inserire, e la loro utilità.

Una cosa su cui mi sono soffermato di più è stato capire se era possibile modificare una foto con l'Api messa a disposizione. Un parametro che l'Api mette a disposizione è allowEdit un parametro booleano, che consente delle semplici modifiche dell'immagine prima della selezione. Purtroppo sulla documentazione nel sito è possibile apprendere che questo parametro non è supportato dal sistema operativo Android. La modifica della foto era uno delle cose secondarie che mi era stata richiesta quindi l'ho messa in secondo piano e sono andato avanti con la realizzazione dell'applicazione.

6.2 Lettura dati da Server

Il secondo passo che è stato affrontato, è la realizzazione di una chiamata a un server per la lettura di dati. Le informazioni messe a disposizione dal server sono dei dati appartenenti ad un database di prova. La realizzazione di questa seconda applicazione ha avuto come obiettivo quello di imparare/ scoprire il funzionamento e lo scambio di richieste/ dati tra un cliente che utilizza il device e un server.

La prima cosa fatta è stata quella di trovare la modalità più corretta per una chiamata al server, e il metodo giusto per dialogare con esso. Apache Cordova mette a disposizione un'Api che ti permette di caricare, scaricare file da server, questa soluzione non è quella che cercavo ma l'Api FileTransfer tornerà utile più avanti.

Navigando in rete molti blog e forum indicavano come soluzione migliore per la chiamata al server con la tecnica AJAX, leggendo e informandomi ho capito che questa chiamata si può effettuare solo se si utilizza il jQuery, la spiegazione di questi è stata fatta in precedenza.

AJAX è l'acronimo di Asynchronous JavaScript and XML ed è ad oggi la tecnica più utilizzata per sviluppare applicazioni web interattive. Il concetto che sta alla base di una chiamata AJAX è quello di poter scambiare dati tra client e server senza ricaricare la pagina: lo scambio avviene in background tramite una chiamata asincrona dei dati. Questo scambio di dati è realizzato, come si può intuire dall'acronimo, mediante funzioni scritte con il linguaggio JavaScript.

```
673     $.ajax({
674         type: "POST",
675         url: "http://79.59.181.33:8080/mobile/php/mobile_documenti.php",
676         data: "username=" + username + "&password=" + password,
677         dataType: 'jsonp',
678         jsonp: 'jsoncallback',
679         timeout: 5000,
680         success: function(data, status){//se il login va a buon fine leggo il file mobile_documenti.php
```

Figura 3- Chiamata al server con AJAX

La chiamata al server ha dei parametri, si sono utilizzati solo utili per la applicazione.

- **url** : URL della risorsa alla quale viene inviata la richiesta, in questo caso l'indirizzo del server dove era stato montato il database di G.E.P.O;
- **type** : il tipo di richiesta HTTP da effettuare. Accetta GET o POST (il valore di default è: 'GET');

- **data** : che può essere un oggetto del tipo {chiave : valore, chiave2 : valore}, oppure una stringa del tipo "chiave=valore&chiave2=valore2" contenente dei dati da inviare al server, con questa modalità passo l'username e la password dell'utente;
- **dataType** : Tipo di dato che si riceve di ritorno; jQuery tenta di capirlo da solo, ma è sempre meglio specificarlo. I tipi possibili sono: "xml", "html", "script", "json", "jsonp" e "text";
- **success**(data, textStatus,) : funzione che verrà eseguita al successo della chiamata. I tre parametri sono, rispettivamente, l'oggetto della richiesta;
- **error**(XMLHTTPRequest, textStatus, errorThrown) : funzione che verrà eseguita in caso di errore nell'effettuare la chiamata. Un tipico esempio è dato dall'impossibilità di accedere alla risorsa remota perché inesistente o, comunque, non accessibile, nella mia applicazione non viene passato nessun tipo di parametro ma una semplice scritta;
- **timeout** : Tempo in millisecondi dopo i quali la richiesta viene considerata scaduta;
- **jsonp** : Quando si effettuano chiamate JSONP, si suppone che il nome della funzione di callback sia proprio callback.

In questa applicazione l'username e la password sono stringhe fisse che passo al server che le legge e tramite una risposta JSON ritorna dei valori che salvo in apposite variabili. Questa procedura sarà spiegata in dettaglio più avanti, perché sarà una funzione importante per l'applicazione finale.

6.3 Upload Immagine

La terza "mini-app" realizzata prima di iniziare l'applicazione finale è stata quella per caricare sul server un'immagine.

Questa terza applicazione utilizza un metodo già usato, cioè quello che permette lo scatto di una foto con il dispositivo, e un metodo nuovo che serve per caricare la foto sul server. Per caricare un file sul server ho utilizzato l'Api messa a disposizione da Apache Cordova. L'Api che ho utilizzato è quella del FileTransfer, una serie di librerie che permettono di scaricare, caricare file su un server scelto attraverso delle funzioni messe a disposizione.

FileTransefer utilizza la funzione upload eseguita su una variabile di tipo FileTranserf ha dei parametri. Il primo è l'immagine che è stata scattata dal device, seguito dall'indirizzo del server. Win è la funzione chiamata se la procedura va a buon fine, questa funzione dà la possibilità di stampare a video la risposta che il server può ritornare. La funzione fail, invece viene chiamata se si verifica un errore, questa permette la stampa a video della tipologia di errore. La variabile params, che qui non è visualizzata, è un array che permette di passare dati all'interno della chiamata al server. In questo caso semplice ho provato a passare una stringa d'esempio. La variabile params, a sua volta viene inglobata dalla variabile option che poi verrà passata all'interno della chiamata.
La vera e propria chiamata sfrutta la libreria FileTransfer, più precisamente una funzione di essa. Naturalmente c'è la parte di risposta da parte del server ma questa la spiegherò nell'app finale.

Queste tre piccole applicazioni ora mi permettono di essere ad un buon punto per incominciare a elaborare e programmare l'applicazione finale.

7 APPLICAZIONE

La raccolta dei giustificativi per Spazio Verde è sempre stata una procedura lenta e per molte sfaccettature complessa e tediosa. Il cliente che partecipava all'evento o alla manifestazione sponsorizzata dai finanziamenti degli enti pubblici, aveva la responsabilità di raccogliere i giustificativi. La raccolta di immagini e video veniva fatta con macchine fotografiche e con videocamere. Dopo la fine dell'evento, il cliente aveva il compito di raccogliere i media in file zip o chiavette e farle recapitare ai tecnici di Spazio Verde. Qui iniziava la parte più lunga e noiosa per i tecnici, che dovevano analizzare e selezionare le foto e i filmati adatti ad essere utilizzati come giustificativi. Le problematiche di questa procedura sono abbastanza evidenti, la prima è la lunghezza del processo: per avere dei dati su cui lavorare, Spazio Verde doveva aspettare la fine della manifestazione e l'invio da parte dei clienti dei file. La seconda è che la selezione delle immagini veniva attuata dai tecnici di Spazio Verde ,tra tutto il materiale inviato dal cliente indipendentemente dalla quantità e dalla qualità.

Da queste problematiche è nata la necessità da parte dell'azienda di vagliare altre possibilità, tra le quali la creazione di un applicativo che velocizzasse la procedura utilizzata fino a quel momento da Spazio Verde.

Accettato lo stage in questa azienda e capita le necessità, mi sono messo in moto per realizzarla.

La mia esperienza in campo informatico è un esperienza scolastica, universitaria e questa è stata la prima esperienza lavorativa come programmatore.

A scuola e all'università ho studiato in maniera abbastanza profonda la programmazione con il linguaggio Java e la programmazione Web, quindi l'utilizzo di HTML e CSS. La prima decisione che ho dovuto prendere nel mio stage è stata quella di scegliere il modo in cui realizzare l'applicazione che mi era stata richiesta. Ho escluso immediatamente l'ipotesi di scrivere codice nativo, in quanto una delle richieste dell'azienda era quello di realizzare un'applicazione che funzionasse almeno sulle principali piattaforme. Mi sono subito imbattuto nel framework Apache Cordova, di cui inizialmente non riuscivo ad afferrare il funzionamento, fortunatamente dopo aver fatto qualche ricerca ne ho capito l'utilizzo e le potenzialità.

Per un programmatore che si avvicina per la prima volta al mondo del mobile, Apache Cordova è il modo perfetto per ottenere subito dei buoni risultati. Apache Cordova mi permetteva di programmare utilizzando HTML CSS, linguaggi che conosco bene, e il JavaScript linguaggio meno conosciuto ma comunque un linguaggio che ho visto e usato per un periodo. Deciso lo strumento per l'elaborazione dell'applicazione ho iniziato a studiare e capire le Api che Apache Cordova metteva a disposizione, cercando di adattarle alle mie esigenze e agli strumenti che avevo a disposizione.

Per semplificare la comprensione di chi legge questa tesi, ora cercherò di spiegare quale utilizzo un cliente di Spazio Verde fa della mia applicazione.

Le aziende che si appoggiano a Spazio Verde, propongono dei progetti ad un ente pubblico, che, dopo essere stati approvati vengono gestiti dall'azienda tramite il software G.E.P.O. Ogni progetto è ambientato in più nazioni. In ogni nazione possono esserci più manifestazioni riguardanti il progetto. Ogni manifestazione che può essere una fiera o un' evento singolo, ha vari "ordini" che un cliente deve soddisfare. In questi ordini vengono fatte della azioni che devono essere documentate. Questi documenti sono i giustificativi che il cliente deve raccogliere tramite la mia applicazione per giustificare il dispendio dei finanziamenti da parte dell'ente pubblico.

7.1 Spiegazione G.E.P.O. e Tabelle Utilizzate

Per comprendere meglio l'utilizzo dei giustificativi che i clienti andranno a raccogliere con l'applicazione è necessario spiegare almeno in modo superficiale la funzione del software G.E.P.O.

G.E.P.O., è un software interattivo per la Gestione dei programmi utilizzati da Spazio Verde in grado di far dialogare le imprese e i fornitori di servizio, nella massima trasparenza ed efficienza.

In parole più semplici è G.E.P.O è un software gestionale che Spazio Verde utilizza per la amministrazione dei suoi clienti, elaborando dati, gestendo gli ordini, i fornitori, i documenti e tutte le informazioni che servono.

G.E.P.O comprende un database fatto con MySql, un insieme di più di 30 tabelle che Spazio Verde utilizza per la gestione della sua attività. Le tabelle sono molte e per una persona estranea al progetto G.E.P.O. sono di difficile comprensione, la vastità della gestione di questa azienda è stata tradotta in SQL e tutte le particolarità tecniche sono state inserite come campi all'interno di esse.

Di queste 30 tabelle, che sono state popolate con gli anni di attività dell'azienda, all'applicazione ne servono 6.

```
tx.executeSql('CREATE TABLE IF NOT EXISTS UTENTE (ID_utente, nome, cognome, username, password)');
tx.executeSql('CREATE TABLE IF NOT EXISTS PROGETTI (ID_progetto, progetto_nome, ID_azienda_destinataria_nota, ID_variante)');
tx.executeSql('CREATE TABLE IF NOT EXISTS NAZIONI (ID_nazione, nazione_nome)');
tx.executeSql('CREATE TABLE IF NOT EXISTS PROGETTI_NAZIONI (ID_progetto_nazione, ID_progetto, ID_nazione, nazione_nome)');
tx.executeSql('CREATE TABLE IF NOT EXISTS AZIONI_ORDINI (ID_azione, ID_ordine, ID_progetto_nazione, codice, titolo)');
tx.executeSql('CREATE TABLE IF NOT EXISTS DOCUMENTI (ID_documento, nome_assegnato, descrizione_documento, ID_ordine, ID_azione, online)');
```

Figura 4- Tabelle del database interno al dispositivo

Queste sono le tabelle che ho utilizzato di G.E.P.O. Nel database di Spazio Verde le suddette tabelle hanno molti più campi ma io ho preso in considerazione solo quelli utili all'applicazione. La scelta delle tabella fondamentali da utilizzare, è stata fatta dopo aver analizzato il database di G.E.P.O con il tecnico di Spazio Verde, e deciso insieme quali fossero necessarie per l'applicazione. Per fare ciò abbiamo dovuto creare con carta e penna un database per capire se tutte le informazioni necessarie fossero presenti nelle tabelle scelte. Con lo schema ER è stato possibile capire come erano strutturate e collegate tra di loro tutte le tabelle. In Apache Cordova non è possibile mettere le chiavi primarie e i vincoli di integrità referenziale, perché le istruzioni non vengono lette, il linguaggio SQL incorporato dentro al framework di Apache è un linguaggio di base dove non tutti i comandi SQL vengono presi, ma sono accettate solo le istruzioni base.

- *Tabella UTENTE*

Ogni cliente che può accedere a G.E.P.O. da un qualsiasi dispositivo tramite browser è fornito di username e password, di conseguenza i tecnici di Spazio Verde hanno fatto una tabella per gestire tutti i clienti che collaborano con loro. Questa tabella è stata mantenuta per selezionare quali clienti possono accedere e quali no. Per semplificare questa tabella ho tenuto i campi username e password, in più ho memorizzato l'id dell'utente, il nome e il cognome.

Nell'applicazione come è intuibile l'username e la password le ho memorizzate per permettere il login al cliente che può accedere solo alle sue informazioni. L'id utente è il campo più importane e infatti è la chiave primaria della tabella, da questo id posso risalire appunto al cliente e mi permette di capire a quali dati farlo accedere o meno.
Il nome e il cognome sono dei campi superflui che ho memorizzato esclusivamente per una questione estetica, cioè quando un cliente si logga, nell'interfaccia grafica dell'applicazione viene visualizzato nome e cognome.

- *Tabella PROGETTI*

Come suggerisce il nome della tabella, qui memorizzo i progetti che ci sono all'interno di G.E.P.O., cioè i progetti a cui un cliente sta lavorando. La tabella è stata mantenuta perché fondamentale per un cliente accedere ai suoi progetti e visualizzarli.
La chiave primaria di questa tabella è l'id del progetto; da questa informazione si può risalire ai vari progetti in maniera più veloce e si evitano omonimie. Oltre all'id memorizzo altre indicazioni tra cui il nome, l'id della azienda che sta partecipando a progetto e un id variante. Questi ultimi due campi sono utili per la comunicazione con il server, che li utilizza per l'elaborazione delle informazioni che verranno inviate da un cliente attraverso l'applicazione.

- *Tabella NAZIONI*

I progetti vengono sviluppati tramite le manifestazioni, che possono essere fiere dove viene pubblicizzato il prodotto, oppure semplici eventi che avvengono in più parti del mondo. Da qui la necessità di memorizzare il nome della nazione dove il progetto viene realizzato. Questa scelta è stata fatta per semplificare al cliente la ricerca dei propri dati attraverso la selezione del nome dei paesi presenti nella tabella.
La chiave primaria di questa tabella è l'id della nazione.

- *Tabella PROGETTI_NAZIONI*

L'elenco unito dei progetti e delle nazioni è necessario per diminuire le tempistiche di ricerca delle informazioni, e serve da tramite tra le tabelle PROGETTI e NAZIONI.
La tabella è formata da una chiave primaria che è un id e poi dai due campi rimanenti che hanno una chiave esterna con le tabelle nominate in precedenza.

- *Tabella AZIONI_ORDINI*

AZIONI_ ORDINI è un insieme di due tabelle di G.E.P.O. che unisce le informazioni per individuare l'azione e l'ordine il quale deve essere giustificato dai documenti da parte del cliente. L'unione delle due tabelle è stata fatta per diminuire le interazioni tra le due. Un'altra ragione per la scelta di cui sopra, è che un cliente individua i suoi documenti attraverso il binomio ordine e azione, in quanto una non può prescindere dall'altra.

La tabella è formata da tre id e due campi di descrizione che sono il nome e il codice, campi più riconoscibili da parte dell'utente rispetto agli id.

- *Tabella DOCUMENTI*

Questa è la tabella che memorizza le informazioni riguardanti i documenti e le informazioni per recuperarli. I documenti sono i giustificativi che il cliente deve fornire a Spazio Verde: generalmente sono fotografie o video ma possono essere anche essere documenti di testo o audio. Questa tabella è stata mantenuta appunto per permettere al cliente di verificare i media presenti sul server e sapere i giustificativi da inviare.

La tabella ha più campi tra cui un id sequenziale associato agli id degli ordini e delle azioni a cui il documento appartiene. Oltre a questo ci sono dei campi di descrizione del documento, importanti perché da questi i tecnici di Spazio Verde capiscono la tipologia a cui si riferiscono. L'ultimo è un campo aggiunto da me, una variabile booleana che serve per capire se il documento è online o deve essere ancora caricato.

7.2 Premesse

Nel poco tempo che ho avuto a disposizione per elaborare l'applicazione richiesta, non sono riuscito a realizzare un applicativo che potesse essere utilizzato dall' azienda e dai sui clienti. All'App come già detto manca totalmente la parte grafica che non è stata presa in considerazione, ritenendo, che sviluppare le funzioni che sono state richieste fossero più importanti e più stimolanti da realizzare. La parte grafica può essere sviluppata in un secondo momento anche da un tecnico che non conosce tutti i dettagli

del codice scritto. Per la creazione dell'interfaccia è necessaria una conoscenza profonda dell'HTML e del CSS che non necessariamente fa parte del bagaglio di conoscenze di un programmatore.

La mancanza di tempo, oltre alla parte grafica mi ha privato anche della parte dei test di funzionalità, perciò l'applicazione non è stata provata sul vero database G.E.P.O, ma su una sua copia di prova messami a disposizione dai tecnici di Spazio Verde.

E` stato creato un server dove è stata montata una copia ridotta di G.E.P.O, perfettamente uguale ma con dati fittizi, in modo che si potessero fare tutte le prove senza andare a intaccare dati importanti dell'azienda.

Le pagine importanti del server saranno analizzate più avanti, ma voglio subito precisare che sono state scritte dal tecnico di Spazio Verde in Php. Inoltre anche le problematiche relative al server che ho incontrato durante lo stage sono state risolte insieme.

Un' altra premessa da fare sono state alcune scelte fatte.

Una di queste scelte è stata il non utilizzo dell'ambiente di lavoro Eclipse. Quando mi sono avvicinato al mondo di Apache Cordova per installare e lavorare con Android mi è stato chiesto di installare la Sdk di Android. Per testare l'applicazione si doveva settare l'emulatore che Eclipse metteva a disposizione.

Dopo giorni passati a settare nella maniera corretta l'emulatore e installare nella giusta maniera la Sdk, mi sono accorto che per il pc che avevo a disposizione allo stage la procedura di esecuzione dell'applicativo era decisamente troppo lenta: per vedere i risultati a video ci impiegavo 4/5 minuti. Di conseguenza ho deciso di settare il pc per la prova sui dispositivi, notando che la procedura era più veloce.

Da Questa scelta di non usare l'emulatore e quindi di non usare l'ambiente di lavoro Eclipse, ne derivano altre che hanno influenzato il mio lavoro in maniera notevole.

La prima è stata la scelta del dispositivo su cui provare l'applicazione. La scelta del dispositivo è stata abbastanza forzata, avendo a disposizione un Pc che montava il sistema operativo Windows 7, ho abbandonato in partenza la possibilità di programmare in XCode cioè il linguaggio di programmazione del mondo iOS. Le alternative erano quindi BlackBerry, Windows e Android. Anche in questo caso la scelta è stata forzata, il dispositivo presente è stato uno smartphone Android. Avendo a disposizione solo

questo smartphone ho iniziato a programmare installando solamente la piattaforma Android al mio progetto.
Incontrando difficoltà con l'uso di Eclipse, ho ripiegato sull'utilizzo di Notepad++, editor più semplice da usare. Facendo questa scelta mi sono precluso delle comodità che Eclipse metteva a disposizione (di questo ne parlerò in un capitolo a parte).

Prima di analizzare l'applicazione, quindi faccio un riassunto delle premesse:

- Ogni cliente di Spazio Verde che possiede questa applicazione è munito di Username e Password per entrare nel database.
- Al primo utilizzo dell'App bisogna inserire Username e Password per connettersi a G.E.P.O
- Le tabelle che ho utilizzato nella applicazione sono semplificate rispetto a quelle usate nel database dell'azienda.
- Per mancanza di tempo i giustificativi che inizialmente potevano essere foto, video, audio, li ho ridotti alle semplici foto, recuperabili dalla galleria o fatte al momento
- Ho testato e sviluppato solo la parte relativa al mondo Android per mancanza di dispositivi su cui testare il lavoro fatto.
- Un cliente può avere l'applicazione solo se viene installata nel dispositivo.

7.3 Funzionamento e Sviluppo

Le problematiche della procedura di raccolta dei giustificativi, hanno portato l'azienda a vagliare altre possibilità. L'applicazione che ho realizzato ha come obiettivo la riduzione dei tempi di raccolta delle foto. Con la realizzazione dell'applicativo un cliente può inviare in tempo reale i giustificativi dell'evento a cui ha partecipato, facilitando l'inserimento dei dati in G.E.P.O. L'applicazione andrà ad inserire il giustificativo già codificato in maniera corretta all'interno del database.
Oltre a realizzare l'App, ho cercato di renderla semplice e intuitiva per i clienti di Spazio Verde, nonostante il poco tempo che non mi ha permesso di concentrarmi sulla parte grafica, ma ho provato a scrivere un software che soddisfacesse le richieste fatte.
Dopo l'installazione dell'applicazione il cliente troverà nel suo dispositivo, un'icona, che come tutti i dispositivi touch per lanciarla basterà toccarla.

Alla prima esecuzione, appare a video un bottone che se pigiato richiama una serie di funzioni, queste funzioni eseguono dei controlli che alla fine portano al richiamo della funzione login().

Nella funzione login() grazie alla possibilità del linguaggio JavaScript di interagire con la pagina HTML con cui è legato, utilizzo delle istruzioni per scrivere codice HTML nella pagina index.html. dove faccio comparire a video un form per l'username e password.

Il cliente quindi si ritrova con un semplice form da compilare con le proprie credenziali, e un tasto di conferma che deve essere premuto dopo aver inserito username e password.

Il tasto di conferma richiama la funzione download() del file JavaScript.

7.4 Funzione Download

Un cliente che vuole fare una foto deve poter decidere dove questa deve essere caricata, cioè per quale evento deve essere un giustificativo. Per avere queste informazioni nell'applicazione e navigarci è necessario prima di tutto recuperale .

L'obiettivo della funzione download() è quello di recuperare informazioni dal database di G.E.P.O. Questi dati sono le informazioni che un cliente di Spazio Verde può trovare nell'interfaccia Web, in altre parole sono informazioni memorizzate nel database dell'azienda che riguardano i progetti a cui il cliente sta lavorando. L' idea di base della funzione è quindi salvare solo le informazioni che il cliente può visualizzare e ciò è possibile interrogando il server leggendo così il database online. Questi dati devono essere letti e memorizzati in maniera efficiente per dopo far si che si possano riutilizzare facilmente. L'unico modo per arrivare al database online è tramite una chiamata al server dove andremo a richiedere delle informazioni.

La chiamata al server l'ho fatta in AJAX. AJAX è l'acronimo di Asynchronous JavaScript and XML ed è ad oggi la tecnica più utilizzata per sviluppare applicazioni web interattive. Il concetto che sta alla base di una chiamata AJAX è quello di poter scambiare dati tra client e server senza ricaricare la pagina: lo scambio avviene in background tramite una chiamata asincrona dei dati, il tutto realizzato con il linguaggio JavaScript.

Dopo aver aggiunto le librerie jQuery all'interno della mia applicazione è stato possibile utilizzare la chiamata al server di AJAX.

```
667         //prendo i valore del login
668         var username = $("#username").val();
669         var password = $("#password").val();
670         }
671         //chiamata al server
672         $.ajax({
673             type: "POST",
674             url: "http://79.59.181.33:8080/mobile/php/mobile_documenti.php",
675             data: "username=" + username + "&password=" + password,
676             dataType: 'jsonp',
677             jsonp: 'jsoncallback',
678             timeout: 5000,
679             success: function(data, status){//se il login va a buon fine leggo il file mobile_documenti.php
680                 sessione=1;
681
682                 $.each(data, function(i,item){ //scorro le stringhe del php
683
684                     if(item.ID_utente != 'no_valid_login'){ //if che mi controlla la risposta adel json se è andata bene la connessione
```

Figura 5- Chiamata al Server

I parametri della chiamata AJAX sono già stati spiegati, l'unico a cambiare è l'indirizzo del server.

Da parte del cliente, nel mio caso da parte del dispositivo non aveva senso interrogare il database online, per motivi di complessità e di elaborazione dei dati. Questa funzione di interrogazione viene fatta in una pagina PHP, la stessa a cui sarà fatta la richiesta. L'unica informazione necessaria che doveva provenire dal dispositivo erano le credenziali del cliente. Questa informazione viene letta e verificata all'interno della pagina PHP del server.

La pagina PHP quindi controlla le credenziali, in caso negativo mi avvisa con una risposta di malfunzionamento.

Questa pagina è stata fatta dal tecnico di Spazio Verde. Un ostacolo per lui è stato capire la modalità per la risposta da far leggere al mio codice le informazioni. Cercando in internet ho capito che una modalità per leggere le risposte del server era il linguaggio JSON.

JSON è un formato adatto all'interscambio di dati fra applicazioni client-server. È basato sul linguaggio JavaScript. Viene usato in AJAX come alternativa a XML/XSLT. Per il tecnico scrivermi una risposta in JSON è stato semplice utilizzando un esempio trovato online. La risposta del server contiene il risultato di un'interrogazione al database che sfrutta le credenziali che il client gli ha passato, l'interrogazione nella pagina PHP è fatta con una query SQL. La risposta affermativa della pagina PHP mette

a disposizione le informazioni che mi servono, il risultato dell'interrogazione viene messo in un array che sarà inserito nella risposta JSON. L'array conterrà tante stringhe quante sono le righe della risposta all'interrogazione. La prima stringa dell'array però è diversa dalle altre, è la stringa in cui io capisco se qualcosa è andato storto dalla parte server, ad esempio se le credenziali sono errate il server mi manda un messaggio di errore altrimenti con una verifica positiva delle credenziali, in risposta il server manda l'id dell'utente con nome e cognome. Per capire se ci sono stati errori, e bloccare eventualmente tutto il processo, per prima cosa analizzo la prima stringa dell'array salvato. In particolare vado a verificare con un controllo if se la prima stringa è diversa dalla stringa di errore che la pagina Php mi dovrebbe mandare. Se il controllo è positivo procedo con la memorizzazione in array specifici di tutte le altre stringhe. Nel leggere il resto dell'array di risposta, memorizzandomi i valori, che fossero stringhe o variabili intere, mi sono accorto che avevo dei problemi poi nella scrittura del database con le variabili che non avevano valore, di conseguenza ho dovuto far dei controlli e specificare in queste variabili il loro valore 'null'. Uno dei problemi che ho riscontrato nel creare questa funzione è stato quello di sapere quando avevo bisogno di avere le credenziali oppure quando l'utente voleva solo aggiornare la sessione in cui si era già loggato. Il problema è stato risolto semplicemente aggiungendo alla funzione un parametro con un valore booleano così da riconoscere i casi. Se le credenziali sono appena state inserite si salvano in variabili.

Per la realizzazione di questa funzione non ho utilizzato nessuna Api di Cordova, la difficoltà principale è stata quella di trovare una modalità per fare una richiesta al server. La chiamata AJAX si è rivelato un modo semplice e abbastanza intuitivo da utilizzare. Si sono riscontrati alcuni problemi quando c'era un basso livello di segnale, in quel caso l'applicazione poteva metterci dei minuti per segnalare l'errore o per concludere la procedura.

L'altra piccola difficoltà è stata quella di trovare il modo più semplice per passare le informazioni dal server al cliente, JSON è stata la soluzione.

Scaricato le informazioni richieste e memorizzate in variabili, il passo successivo da fare era quello di trovare un modo per poter leggere quelle informazioni.

7.5 Funzione Populate

Lette e memorizzate le informazioni, dovevo trovare una modalità che permettesse al cliente di visualizzare queste informazioni. Ho optato per la più classica e immediata delle idee, ricreare il database che ho interrogato tramite la chiamata al server nel mio dispositivo. Le tabelle che ho utilizzato nel database interno le ho spiegate in precedenza, comunque riassumendo, ho ricreato un database più piccolo ed essenziale rispetto al database di G.E.P.O.

La modalità per creare un database all'interno di un dispositivo ,viene fornita da Apache Cordova con l'Api integrata. Con l'istruzione executeSql() posso scrivere in linguaggio SQL, creando, cancellando e popolando tabelle. L'idea che ho utilizzato è stata quella di cancellare tutte le tabelle che esistevano per poi ricrearle e popolarle. La fase di cancellazione può sembrare obsoleta la prima volta che si crea il database all'interno del dispositivo, ma è fondamentale quando invece il database lo voglio aggiornare, in modo da ripulire la memoria interna del dispositivo. Dopo aver ricreato le tabelle con le istruzioni classiche dell'SQL vado a popolarle una alla volta.

Con un ciclo for, e utilizzando le variabili memorizzate nella funzione download(), ho popolato in maniera automatica tutte le tabelle. Creare questa funzione è stato semplice grazie all'Api integrata di Apache Cordova, poter utilizzare il linguaggio SQL in maniera così intuitiva mi ha facilitato la creazione del database interno. Non sono riuscito a capire dove il database venisse salvato all'interno del progetto e come poter migliorare la sua gestione. Durante la creazione del database ho riscontrato alcuni limiti nel linguaggio SQL adottato da Cordova. Apache Cordova non ha installato l'ultima versione di SQL disponibile, il problema si è presentato in fase di creazione delle tabelle, quando ho provato a collegare le tabelle tra di loro con la integrità referenziale. L'applicazione non funzionava quando provavo a inserire queste clausole, quindi ho utilizzato un linguaggio SQL semplificato. Secondo me Apache Cordova ha fatto una scelta, ha tenuto solo i comandi base di SQL per evitare rallentamenti su le operazioni interne del dispositivo. Eseguire operazioni e elaborare tabelle con istruzioni complicate può avere un costo computazionale notevole, e visto che non tutti i dispositivi hanno le stesse caratteristiche Cordova non ha voluto inserire l'intero linguaggio SQL. Nello specifico della mia applicazione questa mancanza non mi ha limitato in nessun modo,

l’unica cosa è stata appunto una costruzione delle tabelle senza chiavi primarie senza integrità referenziali e quindi meno precisa di quello che avrei voluto.

Finita la creazione del database, erano disponibili all’interno del device tutte le informazione che un cliente può avere.

7.6 Interfaccia e Navigabilità

Il passo successivo dopo aver recuperato le informazioni e salvate all’interno del dispositivo, era quello di trovare una modalità per poterle visualizzare in maniera semplice ed efficace. L’idea di base che ho utilizzato è stata quella di ricreare a livelli il database, facendo visualizzare le informazioni più importanti. Il mio obiettivo finale era quello di creare un interfaccia abbastanza fluida per permettere al cliente di muoversi all’interno dell’applicazione in maniera veloce. Non avendo mai lavorato con dispositivi touch e non conoscendo tutte le sfaccettature di questa tecnologia, sono partito da zero con il problema della navigabilità all’interno dell’applicazione, cioè li modo in cui un’utente poteva muoversi senza problemi tra le funzioni che l’app metteva a disposizione. Non conoscendo le modalità con cui i programmatori di applicazioni creano le interfacce per gli smartphone, non sapevo come ricreare quella fluidità che caratterizza la navigazione nelle applicazione smartphone. Una cosa che potevo sfruttare sicuramente era sicuramente la funzione del touch che è intergrata nel pacchetto di Apache Cordova. Per sfruttare la funzionalità messa a disposizione sono state utilizzate le caratteristiche del JavaScript che attraverso i bottoni e le funzioni da richiamare. JavaScript resta in ascolto, legge ogni tocco da parte dell’utente sullo schermo del dispositivo. Inizialmente volevo ricreare una navigazione tra pagine, ma la complicazione di avere solo un file JavaScript mi ha fatto cambiare idea. Per ogni livello di informazione relativo al database ho creato un’ interfaccia a video. Per passare da un livello all’altro ho sfruttato la capacità di JavaScript di intervenire e scrivere codice sulla pagina HTML, in particolare l’istruzione empty() e append().

Il primo livello che il cliente trova dopo aver effettuato l’accesso, è il livello con tutti i progetti a cui sta lavorando, e per ogni progetto c’è un bottone. Per fare ciò ho utilizzato sempre le librerie di Cordova andando prima a interrogare il database interno del dispositivo con una semplice query, la risposta che viene memorizzata in una apposita stringa contenente tutti i progetti a cui il cliente sta lavorando. Per far

visualizzare a video ciò, sono andato a scandire la stringa e a creare per ogni progetto dei bottoni sfruttando la funzionalità di JavaScript di scrivere codice HTML, in questo modo cioè interrogando il database e mettendo a video le risposte ho creato delle funzioni per ogni livello che il cliente può visitare.

I problemi principali erano quelli di sostituire i pulsanti del livello precedente con quelli nuovi al tocco del bottone che richiamava la funzione, e memorizzare i bottoni del livello precedente. La difficoltà è stata quella trovare una modalità per la memorizzazione delle informazioni che il cliente sceglieva per andare ad interrogare in modo adeguato il database.

Quando il cliente si collega una delle variabili che mi salvo è l'id utente del cliente stesso, questo mi consente di avere un identificativo unico, che mi permette di fare una selezione delle informazioni. Per il livello dei progetti la query di interrogazione al database è molto semplice.

```
73      //query che legge i progetti, o richiama errorCB
74      tx.executeSql('SELECT * FROM PROGETTI ', [], querySuccess, function(err){errorCB(err,1)});
```

Figura 6- Istruzione tx.executeSql

La query trova semplicemente tutti i progetti che sono presenti all'interno del database (il database ora presente all'interno del dispositivo riguarda solo il cliente che ha effettuato l'accesso). L'istruzione tx.executeSql richiama querySucces(). In questa funzione vado a leggere il risultato e a scrivere il codice HTML che crea i bottoni. Nel database su cui ho provato questo primo livello dei progetti in molti clienti non era sviluppato, quindi il primo livello diventavano le nazioni su cui il progetto era attivo. Nella maggior parte dei clienti la prima interfaccia poteva essere quella delle nazioni, la metodologia usata è sempre la stessa. La funzione nazione() ha il parametro di default che utilizza Cordova per lavorare con i database e un parametro che ho usato per memorizzare l'id selezionato in questo caso m sta per l'id del progetto. La query mi darà tutte le nazioni dove il progetto è attivo.

```
114  //funzione che fa la query che mi da le nazioni per l'ID_progetto schiacciato
115  function nazioni(tx, m) {
116      g=m;
117      tx.executeSql('SELECT * FROM PROGETTI_NAZIONI WHERE PROGETTI_NAZIONI.ID_progetto='+m, [], outputNazioni, errorCB);
118  }
```

Figura 7- Funzione nazioni

In outputNazioni renderò visibili le nazioni a video.

Il problema di trovare il percorso giusto è stato risolto, semplicemente,

aggiungendo un parametro alle funzioni che andavano ad interrogare il database con le informazioni. Di solito queste sono degli id e saranno messe sempre nella clausola WHERE dell'interrogazione SQL.

Il secondo problema incontrato nel programmare l'interfaccia dell'applicazione, è stata appunto la navigazione attraverso i livelli. Ricercando in internet mi si sono aperte più strade. Una di queste era l'apertura di pagine nel momento in cui il bottone viene premuto, un po' come avviene nelle pagine HTML che sono destinate al browser per intenderci una normale navigazione nei siti Web. Questa soluzione è stata scartata quasi subito, perché significava collegare più pagine a un unico JavaScript con conseguenti problematiche o fare più file Javascript che però diventavano di difficile gestione.

La soluzione adottata forse non è molto elegante ma sembrava la più fluida e più veloce da sviluppare, visto il poco tempo a disposizione.

Nel file index.html è stata creata una divisione con il tag div, la grafica e l'interfaccia che riguarda la navigazione è creata interamente all'interno di questo tag. Ad ogni livello con l'istruzione empty() si cancella tutto quello che c'è all'interno del div denominato output1. Nella variabile database viene scritto il codice HTML che compare a video e con l'istruzione append() viene aggiunto al div.

```
120  //funzione che scrive le nazioni a video
121  function outputNazioni(tx, results) {
122      var len = results.rows.length;
123      $("#output1").empty();
124      for (var i=0; i<len; i++){
125          var database = '<button onclick="successOR('+results.rows.item(i).ID_progetto_nazione+');" class="bottoniOutpu
126          $('#output1').append(database);
127      }//for
128      var back='<div id="back"><button onclick="successCB();">back</button></div>';//bottone che mi fa tornare indietro
129      $('#output1').append(back);
130  }
```

Figura 8- Funzione outputNazioni

Cosi facendo ad ogni funzione che viene richiamata si ricrea l'output voluto a video, così che sia possibile la gestione del pulsante back che fa tornare al livello precedente.

Con le chiamate alle funzioni con i giusti parametri per le interrogazioni al database con questo metodo si può navigare in modo fluido all' interno dell'applicazione.

Per realizzare l'interfaccia e la navigabilità dell'applicazione non è stata utilizzata nessuna Api nuova, fondamentale è stata l'Api integrata che permetteva l'interrogazione al database. La parte dell'interfaccia è stata ricreata sfruttando le potenzialità del JavaScript. Le problematiche dell'interfaccia erano appunto reperire le informazioni che riguardavano il solo cliente e la creazione di un'interfaccia abbastanza fluida.

7.7 Media

Ora l'utente che si trova in mezzo ad una manifestazione per promuovere il suo prodotto, può documentare le sue azioni, fornendo dei giustificativi a Spazio Verde, per giustificare la spesa che gli enti pubblici gli hanno concesso.

Ora il cliente è in grado di navigare in maniera semplice e intuitiva all'interno dell'applicazione ed è in grado di visualizzare gli ordini e le azioni che devono essere giustificati con i documenti. Possono controllare gli ordini che sono già online e decidere dove le foto fatte devono essere caricate. Il problema che sono state affrontate con le seguenti funzioni riguardano i media, non solo la parte dove la foto viene fatta o selezionata, ma l'intera gestione che l'applicazione fa della fotografia scelta.

Come detto, la sola tipologia di documento su cui si è provato sono state le fotografie. Il cliente con il suo dispositivo ora è in grado di loggarsi e navigare all'interno di G.E.P.O per vedere le sue informazioni, e ora ha la possibilità di caricare o aggiungere dei media nei documenti relativi alla manifestazione. Quando l'utente arriva al livello dei documenti scelto, visualizza quelli presenti, da qui l'utente ha la possibilità di aggiungere una foto pigiando il pulsante che compare a video che richiama la funzione sceltaFoto().

Il cliente ora si troverà due possibilità o scegliere una foto dalla galleria o fare una foto istantanea utilizzando la fotocamera. La funzione sceltaFoto() fa comparire video due pulsanti per la selezione. Ovviamente i due pulsanti richiamano due funzioni diverse una per la galleria e una per la fotocamera.

Le due funzioni per le foto le ho già spiegate in precedenza, e la loro funzionalità non è cambiata: la prima accede alla galleria del dispositivo per scegliere la foto desiderata. La seconda fa scattare una foto in modo istantaneo passando per la fotocamera del dispositivo.

La gestione delle foto si è rivelata problematica da più punti di vista. La prima App creata scattava, sceglieva una foto e la caricava istantaneamente sul server. Per la mia applicazione questa funzione non aveva più senso perché così le tempistiche e le richieste dell'azienda non venivano soddisfatte. Quindi si è trovato un modo per memorizzare le foto scelte/scattate in base a quell'ordine e quella azione e caricarle in un secondo momento.

La prima idea valutata è stata quella di salvare le foto fatte nella galleria del dispositivo e quindi di non gestire in maniera interna all'app la memorizzazione delle foto. Questa soluzione è una scappatoia, nel senso che l'applicazione non influenza minimamente la decisione di cancellazione di foto della galleria del dispositivo, cioè l'utente una volta che cancella una foto che riguardava la manifestazione dalla galleria non è più recuperabile o magari non sapeva che quella foto era destinata a giustificativo. Un altro motivo per cui non è stata scelta questa soluzione è che veniva meno l'utilità di fare la foto attraverso l'applicazione perché si scelgono solo le foto dalla galleria.

Quindi ho cercato un metodo per salvare le foto in un posto diverso dalla galleria ma naturalmente utilizzando sempre la memoria del dispositivo.

Studiando una risoluzione mi sono accorto di un nuovo problema, trovare un modo di collegare le foto alle giuste azioni, ordini, documenti.

La soluzione più semplice e adeguata che è stata quella di aggiungere una tabella al mio database interno, una tabella che nemmeno in G.E.P.O c'è. La tabella serve per la gestione delle foto, cioè per ogni foto associo un id e a quell'id associo il documento a cui appartiene. Ciò risulta facile grazie a Cordova, che con una semplice istruzione dopo la creazione della tabella, mi permette di aggiungere una riga, cioè una foto. Trovata la soluzione al problema dell'associazione delle foto con i documenti e i livelli successivi ne è apparso subito un altro.

Ora la difficoltà era quella di automatizzare tutta la procedura senza andare a sovrascrivere le informazioni nel database, cioè trovare una soluzione per non mettere lo stesso id a due foto e quindi sovrascrivere le informazioni con la conseguenza di

perdita di informazioni riguardanti una foto. Il problema è stato risolto in questa maniera. Al primo accesso di un utente o all'aggiornamento del database, quando vengono create le tabelle che servono per l'interfaccia utente, viene creata una tabella in più denomino FOTO.

La tabella FOTO viene creata se non è già esistente, subito dopo viene interrogato il database con lo scopo di visualizzare quante righe sono presenti all'interno della tabella, la risposta viene analizzata nella funzione queryFoto(). In queryFoto() viene fatto un controllo di quanto è lunga la risposta all'interrogazione, e se è pari a 0 si richiama un'altra funzione che inserisce una riga con i valori 0 e 0 per avere almeno una riga nella tabella, cosa che ritornerà utile per quando verranno salvate le foto. Ora vado a riepilogare quello che ho fatto. Dopo la scelta da parte del cliente se scegliere una foto della galleria o farla con la fotocamera, se le operazioni procedono senza errori viene richiamata la seguente funzione.

```
279  //funzione chiamata quando va a buon fine l'operazioni
280  function resolveOnSuccess(entry, d, o, a){
281      var db = window.openDatabase("Database", "1.0", "Cordova Progetti", 200000);//apro il db
282      db.transaction(function(tx){ foto(tx) }, errorCB);
283      //funzione che trova l'id massimo delle foto per non sovrascrivere foto
284      function foto(tx){
285          var id_f;
286          tx.executeSql('SELECT *, MAX(ID_foto) FROM FOTO', [], fo, errorCB);
287          function fo(cx, result) {
288              var len = result.rows.length;
289              for (var i=0; i<len; i++){
290                  id_f = result.rows.item(i).ID_foto;
291              }//for
292          }
293          db.transaction(inserisci, errorCB);
294          //funzione che inserisce la foto nella tab FOTO
295          function inserisci(tx){
296              tx.executeSql('INSERT INTO FOTO (ID_foto, ID_documento) VALUES ('+(id_f+1)+','+(d+1)+')');
297              var n = id_f+1;
298              //new file name
299              var newFileName = n + ".jpg";
300              //funzione che mi sposta il file nella cache
301              window.requestFileSystem(LocalFileSystem.PERSISTENT, 0, function(fileSys) {
302              //The folder is created if doesn't exist
303              fileSys.root.getDirectory( "Android/data/com.example.datage/cache",
304                  {create:false, exclusive: false},
305                  function(directory) {
306                  entry.moveTo(directory, newFileName, function(entry){ successMove(entry, d, o, a)}, resOnError);
307                  },
308                  resOnError);
309              },
310              resOnError);
311          }
312      }
313  }
```

Figura 9- Funzioni per la gestione delle foto

Dopo aver aperto il database viene trovato l'id più alto all'interno della tabella FOTO, in questo modo si risale al nominativo dell'ultima foto che è stata salvata all'interno del

dispositivo. Questo viene fatto andando a interrogare il database interno con una semplice query, e il valore trovato viene memorizzato in una variabile temporanea. La fase successiva consiste nell'inserire nella tabella FOTO l'immagine appena scelta, la foto viene salvata nella tabella con il nome del'id appena trovato aumentato uno. In questo modo ad ogni foto che viene salvata non c'è il rischio di sovrapporre nessuna foto.

Le istruzioni successive appartengono ad una procedura trovata in internet che permette di memorizzare la foto dove si desidera all'interno della memoria del dispositivo. Quindi individuato il percorso, con quelle istruzioni si salva la mia foto appena scelta dall'utente e inserita nella tabella FOTO nella memoria cache del dispositivo.

Un problema no risolto è stato quello di automatizzare anche questo aspetto, cioè la decisone di salvare le foto in un determinato percorso. Il percorso dove si salvano le foto nel mio dispositivo, può non andare bene per gli altri dispositivi, quindi in fase di miglioramento dell'applicazione si dovrà tenere conto di questo aspetto. La cosa non è di facile risoluzione perché nei vari sistemi operativi ci sono molte differenze, un esempio è la piattaforma di iOS, che non lascia molta libertà di manovra nella memoria del dispositivo, come può lasciarla quella Android. Le motivazione sono state accennate nell'introduzione, il mondo Apple è molto "chiuso" e quindi lascia poca libertà per lavorare con le proprie funzionalità soprattutto se non viene utilizzato codice nativo. La difficoltà sarà trovare una modalità che vada bene per ogni tipo di dispositivo, e questa è una grande pecca del'applicazione, che se provata in un dispositivo differente non funzionerà bene.

Dopo aver salvato la foto in una determinata parte di memoria, l'applicazione ha pieno potere sulla gestione di queste foto. Il cliente dopo queste istruzioni può dare un nome e una descrizione alla fotografia appena scelta, campi che servono per il database di G.E.P.O. e per una visione più intuitiva del documento appena caricato, che è visualizzato nella lista di documenti nel dispositivo, con l'unica differenza che, nel campo online c'è un 1, campo che è stato utilizzato per distinguere i documenti presenti online dai documenti ancora da upplodare sul server.

Pieno potere di gestione sulle foto significa poter scattare, cancellare e modificare le foto presenti. La creazione è già stata spiegata e sviluppata, la parte di modifica invece non è stata sviluppata.

Ogni foto salvata, può essere eliminata; per cancellare una foto oltre a rimuovere la foto dalla memoria del dispositivo, bisogna annullare ogni collegamento con i documenti, quindi bisogna cancellare anche la riga nel database.

La cancellazione delle foto può avvenire in due modi, o singolarmente cioè l'utente decide che la foto denominata "x" non va bene quindi preme il pulsante "cancella" oppure le foto vengono eliminate quando un utente fa il logout.

Quando l'utente vuole cancellare un giustificativo dai documenti richiama una serie di funzioni. La prima funzione è quella che cancella la foto dalla tabella dei documenti.

```
1071  //funzione che mi cancella i documenti
1072  function cancellaTabDoc(m){
1073
1074      var id_f;
1075      var db = window.openDatabase("Database", "1.0", "Cordova Progetti", 200000);//apro il database
1076      db.transaction(idfoto, errorCB);
1077      function idfoto(tx){
1078          tx.executeSql('SELECT FOTO.ID_foto FROM DOCUMENTI join FOTO using(ID_documento) WHERE DOCUMENTI.ID_documento='+m, [], param, errorCB);
1079          function param(cx, result) {
1080              var len = result.rows.length;
1081              id_f=result.rows.item(0).ID_foto;
1082              modificaOnline(id_f);
1083              cancellaRigaFoto(id_f);
1084          }
1085      }
1086      db.transaction(cancellaTD, errorCB);
1087      function cancellaTD(tx){
1088          tx.executeSql('DELETE FROM DOCUMENTI WHERE DOCUMENTI.ID_documento='+m);
1089      }
1090      successCB();
1091  }
```

Figura 10- Funzioni per la cancellazione dalle tabelle

In questa funzione per prima cosa si risale all'id della foto, per farlo basta interrogare la tabella FOTO utilizzando una query, quindi si trova la foto associata al documento che si vuole cancellare e si memorizza questa informazione in una variabile. Successivamente con delle istruzioni SQL viene cancellata la riga dalla tabella documenti. Questa parte è stata semplice, poiché è stata sfruttata la possibilità di utilizzare il codice SQL e quindi la parte relativa al database interno è di semplice gestione. Il problema è la cancellazione della foto all'interno della memoria del dispositivo. Le istruzioni per la cancellazione in una parte di memoria sono state trovate in internet.

La Funzione cancellaRigaFoto() è la funzione che va ad eliminare la foto dalla memoria del dispositivo, queste istruzioni mi permettono di cancellare la foto nel percorso che era stato scelto.

```
582  //funzione che cancella la foto
583  function cancellaRigaFoto(id_f){
584      var relativeFilePath = "Android/data/com.example.datage/cache/"+id_f+".jpg";//path
585      window.requestFileSystem(LocalFileSystem.PERSISTENT, 0, function(fileSystem){
586          fileSystem.root.getFile(relativeFilePath, {create:false}, function(fileEntry){
587              fileEntry.remove(function(file){
588                  alert("File removed!");
589              },function(){
590                  alert("error deleting the file " + error.code);
591              });
592          },function(){
593              alert("file does not exist");
594          });
595      },function(evt){
596          console.log(evt.target.error.code);
597  });
```

Figura 11- Funzioni per la cancellazione delle foto dalla memoria

Si inserisce il percorso nella variabile, e si richiama l'istruzione per cancellare la foto che interessa, la funzione darà una risposta con un alert a video.

Questa era la procedura di cancellazione di un solo documento, l'altra avviene quando un utente vuole fare il logout o vuole aggiornare il database .

Quando l'utente si "slogga" gli appare un avviso a video che gli comunica o meno la presenza in memoria di foto riguardanti i progetti su cui lui sta lavorando, quindi foto che lui aveva scelto come giustificativi. Questo avviso è un semplice alert con il numero di foto ricavato da una query che interroga il database. Qui all'utente viene data una scelta se procedere con il logout e quindi cancellare tutte le foto che rimangono o annullare e caricare i giustificativi sul server.

Se viene scelta la prima opzione verrà richiamata la funzione cancellaRigaFoto() per quanti risultati ha dato l'interrogazione al database dove chiedevamo il numero di righe presenti sulla tabella FOTO.

```
1129  //funzione che mi dice quante foto ci sono e mi da la scelta se cancellarle e sloggarmi o restare loggato
1130  function numFoto(tx, results) {
1131      var len = results.rows.length;
1132      if(len==1){
1133          login();
1134      }else{
1135          var r= confirm(len+' foto vuoi continuare con il logout?');
1136          if(r == true){
1137              for (var i=0; i<len; i++){
1138                  cancellaRigaFoto(results.rows.item(i).ID_foto);
1139              }//for
1140          }else{
1141          successCB();
1142          }
1143      }
1144  }
```

Figura 12- Funzione numFoto

In questo modo tutte le foto che restano in memoria vengono cancellate.

Questo è un punto importante, che si è voluto affrontare perché sembrava di importanza rilevante. Lasciare che l'utente faccia il logout senza cancellare le foto che riguardano quella sessione, comporta l'uso ingiustificato di memoria con delle foto che non possono più essere caricate, foto salvate che non possono più essere utilizzate dall'applicazione perché si perde ogni riferimento di esse. Il problema è che le foto non si trovano in una parte di memoria facilmente accessibile da parte del cliente. Se l'utente poi non è esperto di smartphone fa fatica a cancellare in modo manuale le foto per liberare la memoria che occupano. Per poter cancellare le foto in maniera manuale è necessario avere un programma esterno che permetta la navigazione all'interno delle memorie dei dispositivi. Essere riusciti a sviluppare una sequenza di istruzioni che automatizzano il tutto è stato molto importante per i motivi appena elencati. La stessa procedura avviene quando un utente vuole aggiornare il database, le foto che restano in memoria vengono cancellate, e con loro ogni riferimento all'interno del database.

La problematica principale di queste funzioni è stata la gestione delle foto. La decisone della creazione di una tabella ha risolto la problematica della reperibilità della foto, cioè grazie alla creazione di questa tabella è stato possibile associare ogni foto alle informazioni necessarie, oltre a questo è stato possibile rendere unica la foto è quindi renderla facilmente accessibile. La decisone di aggiungere la tabella al database interno ha risolto la parte di gestione che riguarda la parte interna dell'applicazione. Scattare e scegliere le foto è stato possibile grazie all'Api Camera messa a disposizione del sito ufficiale di Apache Cordova. Grazie a questa scattare le foto è stato semplice, come è stato semplice adattare l'Api all'applicazione. La parte un po' più complicata è stata la decisone di non salvare la foto nella galleria, ma in una parte diversa della memoria del dispositivo. Questa è stata una decisone che ha complicato la realizzazione dell'applicazione ma una decisone da fare per la buona riuscita del prodotto finale. Spostare le foto in una parte di memoria differente dalla galleria ha complicato gestire la cancellazione della foto che però era necessaria per non utilizzare la memoria inutilmente.

7.8 Upload

L'ultimo step che rimaneva da affrontare era quello di caricare il giustificativo online.
Ora l'utente può navigare all'interno dell'applicazione in maniera fluida, può raggiungere tutte le informazioni che desidera, può raccogliere i giustificativi, nominarli e salvarli, può aggiornare il database o fare il logout.
L'utente ora può raggiungere all'interno dell'App il livello relativo ai documenti, qui trova una serie di foto o comunque una sere di media, e può decidere se aggiungerne, eliminarne o caricali sul server. Il cliente come già detto può controllare lo stato dei documenti, se sono online o meno questo è possibile grazie ad un attributo presente all'interno del database. I problemi che affiorano adesso sono trovare la metodologia per caricare online le foto e gestire gli eventuali errori. Un cliente può aggiungere giustificativi che riguardano azioni diverse, cioè foto di eventi diversi, e con i tecnici di Spazio Verde abbiamo deciso che l'utente poteva avere la possibilità di caricare tutti i documenti in maniera automatica oppure caricare i documenti relativi ad un solo ordine. Per caricare un file è stata utilizzata l'Api messa a disposizione da Apache Cordova, l'Api per caricare file sul server è FileTransfer. E` stata cambiata in maniera sostanziale la mini applicazione fatta in precedenza che caricava una sola foto. Prima di spiegare il modo in cui è stato FileTransfer, verrà spiegato come sono state reperite le informazioni utili per poter caricare le foto. Per scegliere quali documenti caricare è stato interrogato il database interno al dispositivo, la tabella su cui si è fatta l'interrogazione è la tabella DOCUMENTI. L'interrogazione cambia in base alla volontà del cliente, cioè se vuole caricare tutti i giustificativi o se vuole caricare solo quelli relativi ad un solo ordine. La differenza tra i due metodi in realtà è molto semplice, l'unica cosa che differisce tra di loro è l'aggiunta di una clausola AND all'interno della query che da l'elenco di tutti i documenti che devono essere caricati, quando si caricano solo quelli di un ordine, si aggiunge l'ordine di appartenenza dei documenti. Trovati i documenti e memorizzati in un array, un'altra cosa che serve è recuperare l'id utente, questo viene fatto attraverso una funzione che interroga il database.
Per caricare più foto contemporaneamente si leggono uno a uno i risultati dell'interrogazione e ad ogni documento si compilano i campi necessari per la funzione FileTransfer. Oltre ai parametri dell'Api bisogna trovare tutti i parametri che la pagina Php sul server ha bisogno per caricare la foto. Questi parametri sono dei campi da

riempire, delle informazioni reperibili attraverso un' interrogazione del database interno del dispositivo. La query è una serie di JOIN attraverso le tabelle che servono a trovare tutte le informazioni necessarie. Eseguita la query si legge il risultato e si memorizzano i dati in apposite variabili.

```
393                 var options = new FileUploadOptions();
394                     options.fileKey="blob";
395                     options.fileName=''+ID_f+'.jpg';
396                     options.mimeType="image/jpeg";
397                     options.headers = {
398                         Connection: "close"
399                     };
400                     options.chunkedMode = false;
401 
402                 var blob = "blob";
403                 var descrizione = descr;
404                 var ID_tipo_documentazione = 22;
405                 var ID_azienda_destinataria =ID_des_no;
406                 var comando = "file_ins";
407                 var upload_to = "progetti/"+ID_p+"/gestione/ordini"
408                 var ID_area_documentazione = 7;
409                 var nome_area_documentazione = "tecnica";
410                 var inserimento_nota = 1;
411                 var ID_tabella = 4;
412                 var ID_ordine = ID_or;
413                 var ID_azione = ID_az;
414                 var ID_variante = ID_va;
415                 var ID_progetto = ID_p;
```

Figura 33- Campi della funzione FileTransfer

Dopo aver compilato tutti i campi si procede con la funzione FileTransfer(). I parametri della funzione ft.upload sono, il percorso dell'immagine che si vuole caricare, l'indirizzo della pagina del server, e le funzioni da richiamare se le operazioni vanno a buon fine o meno.

Qui entra in ballo un discorso più ampio che non è stati approfondito a dovere.

La funzione FileTransfer() è una funzione asincrona, così come le funzioni jQuery. Prima di spiegare come è stato risolto il problema, riepilogo velocemente le differenze tra codice sincrono e codice asincrono.

Un codice sincrono viene eseguito in "serie", ovvero in modalità "catena di montaggio", riga dopo riga. Ad esempio, se l'utente clicca su un pulsante attivando una funzione sincrona, il browser resterà bloccato fino al termine della funzione.

Viceversa, se usiamo del codice asincrono, subito dopo aver cliccato il pulsante l'utente avrà di nuovo il controllo del browser. L'esecuzione di una chiamata asincrona avviene in "parallelo", ovvero segue una strada diversa rispetto a quella principale, che rimane in ascolto delle azioni dell'utente. In termini più rigorosi ciò significa che nel browser non abbiamo la possibilità di scrivere codice multi-threading: il codice JavaScript viene eseguito con un unico thread, per cui l'unico modo di eseguire azioni (apparentemente) simultanee è proprio il meccanismo di chiamata asincrona che utilizza thread diversi che hanno cicli di vita diversi che non sono direttamente accessibili direttamente dall'utente.

Per questo motivo, inizialmente pensavo che l'esecuzione delle istruzioni avvenisse in modo sincrono, cioè che ad ogni tentativo di caricare un immagine, la funzione tr.upload(), mi richiamasse le funzioni di errore o di buona riuscita dell'operazione. Facendo delle prove invece si ènotato che la funzione richiama n volte l'istruzione tr.upload(), quante sono le foto da caricare sul server, e poi richiama le funzioni che una alla volta danno gli esiti per ogni immagine. La funzione che viene chiamata se le operazioni vanno a buon fine è l'operazione win() mentre quella per l'errore è la funzione fail(). Se la funzione non fosse stata asincrona, ad ogni tentativo di caricare sul server la funzione win o fail sarebbero state richiamate. Questo è un problema, la funzione asincrona non permette di capire quando c'è un problema e su quale tentativo di caricamento sul server sia avvenuto. Oltre ad individuare l'errore, cosa che si può fare mettendo degli alert sulle due funzioni con i dovuti parametri per capire dove sono stati riscontrati gli errori, c'è il problema delle foto da cancellare. Ogni foto che viene caricata sul server deve essere eliminata dalla memoria e dai vari database che ne tengono conto. Questo con la chiamata asincrona e con i parametri originali della funzione ft.upload() diventa impossibile, perciò si sono aggiunti dei parametri alla funzione win(). I parametri aggiunti sono tre, uno per l'id della foto uno per sapere quante foto sono da caricare, cosi che quando tutte sono state caricare facendo un controllo si può richiamare la funzione che aggiorna il database interno con un parametro per far capire di che tipologia è.

Ora sono state analizzate tutte le principali funzioni dell'applicazione, quindi passerò ad alcune analisi fatte.

8 VALUTAZIONI

8.1 Debugging e Errori

Uno dei problemi più importanti con cui mi sono dovuto scontrare è stato quello del debugging. Non avendo utilizzato Eclipse mi sono precluso le funzionalità che l'ambiente di sviluppo mi poteva dare. La cosa migliore sarebbe stato utilizzare un compilatore, ma non sono riuscito a trovare niente del genere anche se Cordova metteva a disposizione dei plugin su Chrome. Per visualizzare gli errori mi sono dovuto arrangiare come potevo. Inizialmente per la piccola parte grafica e per provare le chiamate a funzioni JavaScript ho utilizzato gli strumenti di sviluppo di Chrome o Firefox, qui potevo vedere dove il codice si fermava e potevo facilmente individuare un errore e correggerlo. Gli strumenti di sviluppo sono facili da utilizzare, basta semplicemente lanciare la pagina HTML dell'applicazione e aprirli dagli strumenti del browser. Come già detto questo strumento l'ho utilizzato solo per la parte grafica e per la parte dell' interfaccia, alla fine per la fase della navigazione sul dispositivo è stato come sviluppare una pagina web. Il mouse andava a sostituire il touch dell'utente e il resto del funzionamento era lo stesso di una pagina web. La parte di debug difficile è iniziata quando sono andato a testare le funzionalità del dispositivo e le Api di Cordova. Per testarle dovevo lanciare l'applicazione nel dispositivo e quindi vedere il risultato a video. Naturalmente qui non avevo nessun tipo di riscontro dal dispositivo, quindi mi sono munito di pazienza. Non sapendo come visualizzare i problemi e gli errori, ho utilizzato pesantemente l'inserimento di messaggi diagnostici all'interno del codice. Un sistema un po' grezzo e poco informatico ma in mancanza di strumenti e tecnologie adeguate mi sono dovuto adattare con le possibilità che avevo. Grazie agli alert potevo capire quali funzioni venivano eseguiti e quali no, individuare la riga dove avveniva l'errore e provare a risolverlo. Qui sono stati importanti anche le chiamate alle funzioni di errore. Le due funzioni grazie sempre a dei messaggi a video mi hanno aiutato a risolvere molti errori, soprattutto quelli relativi alle chiamate al server. Quando ho testato le chiamate al server o l'upload di foto, le chiamate di errore mi avvisavano di che tipo di errore c'era stato, ad esempio se avevo avuto problemi di connessione. La mancanza di un vero e proprio strumento per il debug è stato uno degli svantaggi del framework Apache Cordova che più mi ha causato problemi nella mia applicazione.

8.2 Sviluppi Futuri

Ora l’applicazione è stata analizzata sotto tutti i punti di vista, e le principali richieste fatte da Spazio Verde sono state soddisfatte dal mio codice. Sfortunatamente soprattutto per la mancanza di tempo non ho sviluppato degli aspetti e delle funzionalità per la mia applicazione.+

8.2.1 Grafica, HTML e CSS3

Uno degli aspetti su cui non mi sono soffermato è stata la parte grafica. Questa parte sarebbe stata interessante da sviluppare non solo per abbellire l’applicazione ma per sfruttare tutta la potenzialità che Cordova ti da. Usando a pieno potenziale i due nuovi linguaggi di programmazione si può arrivare ad un risultato più che soddisfacente. L’uso di HTML5 comporta maggiori possibilità di realizzare un codice pulito e semplice, utilizzare le sue nuove potenzialità come la realizzazione di pagine più leggere. L’utilizzo di alcuni tag nuovi ti permette di etichettare più facilmente determinate porzioni di video e realizzazioni semplificate di form per la comunicazione con gli utenti, addiurittura con l’ HTML5 si riesce ad validare alcuni campi senza l’uso del JavaScript. Alcuni tag HTML5 consentono di disegnare sulla pagina web senza l'uso di plugin o librerie di terze parti come avveniva prima. Introduzione dell’elemento denominato canvas. Il nuovo elemento introdotto dal nuovo standard, ormai supportato da anni dalla gran parte dei browser ad esclusione di IE8 e versioni precedenti, permette funzioni aggiuntive per la manipolazione delle superfici generate e include funzioni aggiuntive per la formattazione del testo. Con l'HTML5 i video sono supportati nativamente dal browser senza l'uso di plugin di terze parti come flash e Silverligth, così come l’audio. Introduzione di modifiche di vecchi tag che aumentano notevolmente le loro potenzialità. L’utilizzo del CSS3 invece ti permette tempi di sviluppo e mantenimento inferiore. Molte tecniche CSS3 possono sostituire le immagini, questo vi libera dal dover dedicare tempo alla creazione, sezionamento e ottimizzazione di tutte le immagini che siete abituati a inserire nei vostri progetti. Il CSS3 richiede meno DIV e Classi e questo si traduce in minore lavoro. Non dovendo utilizzare elementi nidificati, la stesura del codice risulta più semplice e scorrevole. Queste nuove caratteristiche ti

permettono una migliore realizzazione della parte grafica che facilitano l'utilizzo della tua applicazione. La parte grafica sicuramente mi avrebbe permesso di migliorare l'interfaccia e la navigabilità rendendola più fluida e migliorandone l'aspetto.

Ho deciso di non soffermarmi sulla questa perché personalmente la ritengo una cosa secondaria, senza il codice funzionante un'applicazione anche se bella graficamente non è utilizzabile. La parte grafica può essere sviluppata anche da un esterno che non conosce a fondo il codice dell'applicazione, mi è dispiaciuto non averla sviluppata soprattutto per non aver approfondito la conoscenza dei nuovi linguaggi per la realizzazioni di applicazioni web, che sicuramente per un programmatore sono un valore aggiunto per il futuro.

8.2.3 Miglioramenti, Test e Porting

Uno degli aspetti da migliorare è quello dei media. Nella mia applicazione ho trattato solamente la foto come documenti, Spazio Verde dai propri clienti riceve file video audio e anche alcuni file di testo. L'implementazione dei media comporta non solo un lavoro sul codice dell'applicazione ma anche un miglioramento della parte Server. Nel sito di Cordova ci sono esempi di file video, quindi non dovrebbe dare troppi problemi l'amplificazione dei media. Il problema potrebbe essere quello della gestione di tutte le tipologie di file che andranno a diventare giustificativi. Premettendo che anche l'aspetto della gestione delle foto che ho utilizzato ha dei limiti, come quello che se c'è un particolare errore durante l'esecuzione si possono perdere le informazione delle foto e quindi perdere la possibilità di cancellarle dalla memoria, andando a occupare spazio inutilmente. Quindi avere più tipologie di file può complicare la cancellazione di esse quando il loro ciclo vitale all'interno dell'app è concluso. Un altro aspetto da migliorare è l'automazione del percorso dove salvare le foto, da modello a modello cambia, nell'applicativo è stato testato solo nel mio dispositivo, quindi se provato con un altro è da verificare la funzionalità. Oltre al miglioramento del ciclo di una foto nell'applicazione si può integrare anche la possibilità di modifica di una foto. Questo aspetto può essere utile per uniformare tutte le foto, portarle tutte ad uno stesso formato, facilitando poi il lavoro dei tecnici.

Purtroppo è mancata completamente la parte di test sull'applicazione. Con i tempi ristretti non sono riuscito a testare l'applicazione. La parte dei test è molto importante perché permette di testare tutte le situazioni e capire se l'applicazione risponde in maniera corretta ad esse. Una delle parti più importanti era quella di capire come si comportava l'applicativo durante scaricamento o caricamento di dati con l'interruzione di connessione internet. Tutte le cose che ho provato le ho provate con una presenza di collegamento wireless, notando problemi quando il segnale era di bassa potenza. L'applicazione avverte quando si interrompe per di mancanza di segnale, ma la cosa interessante sarebbe stato capire come l'applicazione si comporta con interruzione di afflusso dei dati. Capendo come l'applicativo risponde poi si può andare a risolvere nella maniera scelta il problema. Si doveva testare se l'app funzionava non solo con il collegamento ad una rete ma con il solo uso della connessione dati 3G. Un altro test che sicuramente andava fatto era quello di provare l'applicazione con il database vero dell'azienda e testare anche la parte relativa alla sicurezza. Dietro alla fase test c'è poi tutto il lavoro di gestione degli errori che si trovano e si risolvono per rendere l'applicazione pronta a gestire ogni situazione. Si dovevano testare tutte le situazioni, e non avendo fatto questo l'applicazione non è pronta per essere utilizzata dall'azienda.

Fatta la parte test, si deve fare il porting su tutte le piattaforme desiderate, il processo lo si fa attraverso Apache Cordova. E come cosa finale la pubblicazione nei vari store dell'applicazione su tutte le piattaforme.

9 CONCLUSIONI

Al termine dello sviluppo dell'applicazione si devono fare delle considerazioni generali sulle questioni affrontate e sulle problematiche incontrate. L'utilizzo del framework in un progetto completo dà la possibilità di valutare tutti i pregi e difetti che sarebbero difficili da valutare a priori.

I vantaggi chiave che ho incontrato utilizzando Apache Cordova sono questi:

- La promessa dello sviluppo di applicazioni attraverso un linguaggio di programmazione universale viene mantenuto appieno e messo in atto con molta facilità. Si tratta della motivazione principale per cui si è scelto Apache Cordova, testato solo su una piattaforma ma senza problematiche particolari, manca il porting sulle altre principali piattaforme.
- La presenza di Api che permette l'accesso all'hardware nativo, è stata fondamentale per la realizzazione dell'applicazione. Questa caratteristica rende unico Apache Cordova ed è qui che si differenza da una normale web application.
- L'utilizzo di standard web di ultima generazione e di grande diffusione come HTML 5, CSS 3 e Javascript è una scelta ottimale per unificare in un unico linguaggio tutte le diverse piattaforme supportate da Apache Cordova. Si tratta di standard in continua crescita e giunti a un livello di potenza elevato. HTML e CSS sono molto conosciuti quindi per un programmatore è facile utilizzarli, stessa cosa vale per il JavaScript che cambia relativamente poco dal Java.
- Diretta conseguenza del punto precedente è la possibilità di sfruttare tutte le potenzialità di JavaScript e quindi di usufruire di tutte librerie disponibile online per tutti i gusti. In tal senso le più rilevanti in ambito mobile sono certamente jQuery, AJAX. Queste librerie, già viste nel dettaglio, garantiscono un tutt'altro spessore rispetto al tradizionale approccio Javascript. La scelta di Javascript è appropriata anche per il possibile utilizzo di JSON come standard di comunicazione con i Web Service. In molte applicazioni viene preferito all'uso di XML anche grazie all'uso massivo di AJAX. JSON, come già spiegato, viene

gestito nativamente da Javascript, il risultato è quindi un integrazione coi Web Service molto agevolata.

Bisogna però fare i conti con il lato rovescio della medaglia cioè con le cose che hanno rallentato lo sviluppo dell'applicazione, i difetti riscontrati. La realizzazione in prima persona ha evidenziato, anche in un applicazione come la mia che non presenta un livello di logica e funzionalità che prevedano elevata complessità algoritmica, l'assenza di un debugger vero e proprio rappresenti un ostacolo con cui lo sviluppatore non è abituato a convivere. Tutte le piattaforme di sviluppo, infatti, hanno un debugger e offrono quindi la possibilità di scorrere il proprio codice cercando riga per riga e trovando l'errore. Le modalità presentate sono qualcosa ma sicuramente questa mancanza grava molto sulla realizzazione dell'applicazione. Altro difetto rispetto alle applicazioni native è l'impossibilità di accedere al 100% delle capacità hardware per quanto riguarda processore e RAM. L'applicazione quanto a utilizzo di risorse non è di grandi pretese, ma ugualmente si ha spesso l'impressione della presenza di ritardi di risposta agli input degli utenti. Anche lo scroll degli elementi sembra non essere fluido come le applicazioni native.
Il proposito del framework è quello di permettere allo sviluppatore di lavorare basandosi solo sulla conoscenza di Javascript, ma la corretta implementazione di alcune operazioni implica comunque delle modifiche o integrazioni da effettuare in linguaggio nativo. I guadagni derivanti da un approccio nativo sono legati a: un incremento delle prestazioni, una precisione superiore nella creazione dell'interfaccia utente, un maggiore controllo nella gestione degli eventi, possibilità di interfacciamento con tutte le possibilità hardware del dispositivo. A fronte di un maggiore impegno progettazione e codifica esiste la possibilità disegnare l'applicazione in ogni dettaglio in modo da rendere il suo aspetto unico.

La domanda quindi nasce spontanea, sarebbe stato meglio sviluppare l'applicazione con una programmazione nativa?
Adesso andrò ad analizzare le principali funzioni della mia applicazione per vedere se la scelta di Apache Cordova è stata quella giusta.
La funzione che scatta o scegli la foto è stata di facile gestione perché il framework fornisce un API stabile, che ti permette di fare, salvare una foto e in più ti permette di

accedere alla galleria. Con il linguaggio nativo non sarebbe stato più complicato ma le potenzialità dei due metodi sono le stesse. La costruzione di pagine prettamente statiche è risultato semplice, su Cordova per via dell'approccio HTML + CSS, la parte grafica sicuramente sarebbe risultata più precisa e più fluida con il codice nativo. Così come più precisa e controllabile sarebbe stata la gestione delle foto con il codice nativo.

Uno sviluppatore nativo esperto può legittimamente manifestare le proprie preoccupazioni su un framework del genere rispetto alla stabilità e capacità fornite da una piattaforma nativa. L'aspettativa è che l'utilizzo di un tool del genere porti a un risparmio notevole di tempo di fronte a una già acquisita conoscenza di HTML, CSS, Javascript in accoppiamento all'uso di librerie come jQuery. La manutenzione di un codice unico porterebbe non pochi vantaggi di fronte all'esigenza di porre modifiche, ottimizzazioni e miglioramenti. Di contro la necessità di avere una versione del software per ogni piattaforma ne rallenterebbe i tempi. La possibilità di interagire con le funzionalità hardware del dispositivo (come GPS, fotocamera, back button, etc.) in maniera trasparente e stabile su tutte le piattaforme è vantaggio universalmente riconosciuto. La possibilità di incorrere in applicazioni lente, o non fluide, e la difficoltà di ottimizzazione su piattaforme diverse frenano non poco gli sviluppatori in vista di una possibile migrazione. Infine, come presumibile, lo scoglio più arduo da superare rimane il dover sopperire alla mancanza di un debugger, il miglior alleato di ogni sviluppatore. La ricerca dei bug, l'ottimizzazione del codice e i test sono fasi essenziali durante lo sviluppo di un applicativo. E' necessario quindi che tali operazioni possano essere svolte in tempi proporzionati e con strumenti adeguati. Il rimedio offerto dal framework risulta troppo parziale e approssimativo specialmente per chi è abituato all'uso dei debugger offerti da Eclipse, Firebug o X-Code.

L'obiettivo della tesi era la realizzazione di un'applicazione che riducesse i tempi di elaborazione di una procedura che l'azienda Spazio Verde utilizza. L'applicazione si può ritenere complessivamente completa e in linea con
gli obiettivi che si erano dati, raggiungendo un discreto risultato, essendo
riusciti a risolvere il problema di portabilità su più piattaforme (anche se non testato) attraverso Apache Cordova. Possibili sviluppi futuri potrebbero essere sicuramente una maggiore attenzione alla grafica e altri migliorie. Tra le richieste dell'azienda, non era

di primaria importanza prendere in considerazione particolari aspetti grafici, ma lo era invece la parte funzionale dell'applicazione.

Dopo tutte le considerazioni fatte è necessario valutare quanto sia effettivamente proponibile, in ambito industriale, un applicazione sviluppata su Apache Cordova paragonandola alle corrispettive versioni native. Dall'esperienza maturata durante le fasi di sviluppo dell'applicazione è possibile riscontrare quanto l'utilizzo di Apache Cordova risulti essere la soluzione più comoda in termini di difficoltà tempi e costi. Una realizzazione dell'applicazione con codice nativo, tolto la parte di apprendimento ha una sostanziale equivalenza dal punto di vista dei tempi di realizzazione con una realizzazione su Apache Cordova. E` palese come questa presunta parità diventi una caratteristica vincente del framework rispetto alla versione nativa, quando si considera l'applicabilità della soluzione a più piattaforme. La proporzione è forte: 7 (numero di piattaforme supportate) a 1. Considerando inizialmente una stima di tempi/costi di sviluppo equa nelle due versioni, si nota la convenienza e l'utilità offerta da parte di Apache Cordova. Ogni applicazione, in base ai propri scopi e alle funzione che metterà a disposizione può essere sottoposta a diverse esigenze e requisiti in termini di performance, consistenza, e hardware. L'applicazione sviluppata per Spazio Verde, rispecchia un profilo di utilizzo di risorse medio-basso e risponde quindi perfettamente al profilo di applicazione a cui Apache Cordova meglio si adatta. E' con questa tipologia di applicazioni che il framework risulta più efficace e si pone come soluzione valida per un deploy multipiattaforma. Ritengo che di fronte a progetti di sviluppo più complicati che richiedano elevate capacità di calcolo, che coinvolgano un'ingente quantità di dati in memoria o che possano necessitare di elaborazioni grafiche complesse, difficilmente il framework potrà soddisfare le esigenze necessarie. I tempi di sviluppo potrebbero crescere in maniera più consistente rispetto alle versioni native e, come già spiegato, si andrebbe incontro a problemi di performance e di capacità applicativa. La scelta nativa diverrebbe quindi obbligatoria. La possibilità offerta da Apache Cordova di sviluppare applicazioni con l'uso di semplice HTML, CSS e JavaScript lo porta ad essere accessibile anche da sviluppatori meno esperti o con minor familiarità con la programmazione nativa e di poter arrivare a tutti i dispositivi in breve tempo in concomitanza con un importante risparmio. Ed è questo il grande vantaggio che Apache Cordova mette a disposizione dei programmatori, la possibilità di

avvicinarsi al mondo mobile e permettere la realizzazione di applicazioni basiche (o anche più strutturate) in poco tempo o con limiti iniziali di budget.

Di fronte a questo genere di richieste Apache Cordova è in grado di soddisfare i requisiti offrendo la possibilità di proporre soluzioni che, pur non garantendo il massimo dal punto di vista di qualità e performance, possono essere adeguate alle esigenze sottoposte e consentano di ottenere applicazioni multipiattaforma in tempi brevi.

APPENDICE IMMAGINI

[Figura 1]- Tabella delle compatibilità delle Api messe a disposizione da Apache Cordova, reperibili sul sito ufficiale.

[Figura 2] - Applicazione che scatta una fotografia accedendo alla fotocamera del dispositivo, e che da la possibilità di selezionare una foto della galleria del dispositivo.

[Figura 3] - Chiamata al Server con AJAX, si possono vedere i principali parametri che servono.

[Figura 4] - Tabelle del database interno al dispositivo, con tutti i campi relativi.

[Figura 5] - Chiamata al Server utilizzata nell'applicazione finale con relativi parametri utilizzati.

[Figura 6] - Istruzione tx.executeSql con query SQL che interroga il database.

[Figura 7] - Funzione nazioni, contente query SQL che recupera informazioni dal database interno.

[Figura 8]- Funzione outputNazioni, funzione che mi permette di visualizzare a video le risposte alle interrogazioni al database.

[Figura 9] - Funzioni per la gestione delle foto, contiene le funzioni che servono per la gestione della vita di una foto, inserimento nella tabella e il metodo per salvarla nella memoria cache.

[Figura 10] - Funzioni per la cancellazione dalle tabelle presenti nel database interno, per cancellare ogni riferimento all'immagine.

[Figura 11] - Funzione per la cancellazione della foto dalla memoria cache del dispositivo in modo da non occupare memoria inutilmente.

[Figura 12] - Funzione numFoto che dice quante foto sono presenti all'interno della tabella FOTO.

[Figura 13] - Campi della funzione FileTransfer utilizzati per caricare una foto sul Server.

BIBLIOGRAFIA

[1] **Apache Cordova** - *About Apache Cordova* - https://cordova.apache.org/

[2] **Apache Cordova Plugin System,** - http://plugins.cordova.io/#/

[3] **Apache Cordova Documentation** - http://cordova.apache.org/docs/en/5.0.0/

[4]**TheCommand-LineInterface-**
http://cordova.apache.org/docs/en/5.0.0/guide_cli_index.md.html#The%20Command-Line%20Interface

[5]**Apache Cordova API Reference** -
http://cordova.apache.org/docs/en/5.0.0/cordova_plugins_pluginapis.md.html#Plugin%20APIs

[6] **HTML-Descrizione Generale** -*Wikipedia* - http://it.wikipedia.org/wiki/HTML

[7]**Introduzione ai fogli di stile, Cascading Style Sheet** -
https://it.wikipedia.org/wiki/CSS

[8] **JavaScript,** *Wikipedia* - http://it.wikipedia.org/wiki/JavaScript

[9] **AJAX,** *Wikipedia* - http://it.wikipedia.org/wiki/AJAX

[10] **jQuery,** *Wikipedia* - http://it.wikipedia.org/wiki/JQuery

[11] **JSON,** *Wikipedia* - http://it.wikipedia.org/wiki/JQuery

[12] **NodeJs,** *Wikipedia* - http://it.wikipedia.org/wiki/NodeJs

[13] **Ant,** *Wikipedia* - http://it.wikipedia.org/wiki/Ant

[14] **Git,** *Wikipedia* - http://it.wikipedia.org/wiki/Git