Indice

Introduzione

Il seguente elaborato ha lo scopo di esaminare le problematiche relative allo sviluppo di applicazioni per il web e delle modalità con cui possono essere superate limitando lo sforzo dei programmatori. La trattazione si articolerà nella presentazione di alcuni dei linguaggi per il web e di alcune delle tecniche che gli sviluppatori utilizzano nella realizzazione applicazioni adattive.

In un epoca come la nostra, in cui la maggior parte degli esseri umani ha accesso alla rete, Siti Dinamici (o Web applications) rappresentano una risorsa dal grande potenziale sotto molti punti di vista. Proprio il fatto che ad accedere al web sia una vastità di persone con una grande quanto varia quantità di dispositivi rappresenta un problema per gli sviluppatori.

Un qualsiasi programmatore, più o meno esperto, si troverà quindi di fronte alla problematica di progettare un'applicazione che deve essere in grado di interagire con l'hardware di un gran numero di *devices* e che principalmente deve avere un'interfaccia che si adegui a schermi di dimensioni variabili.

Risulta interessante e utile analizzare le varie soluzioni adottate dagli sviluppatori per il Web per riuscire nel compito, apparentemente impossibile, di adeguarsi alle svariate tecnologie e inoltre sembra lecito domandarsi se esista o meno una soluzione universale per il problema dello sviluppo di web apps cross-platform. L'analisi svolta non ha la presunzione di individuare tale soluzione, ma può essere d'aiuto nella scelta di un

determinato processo produttivo, avendo coscienza dei propri obiettivi e delle risorse che si hanno a disposizione.

Capitolo 1: Il Web multipiattaforma e le sue problematiche

1.1 Programmazione Web

Con Programmazione Web si indicano le attività e le tecniche di programmazione, che consentono la realizzazione e lo sviluppo di applicazioni per il Web.

Un Sito Web insieme di più pagine Web, collegate tra loro per mezzo di collegamenti ipertestuali, e cui si accede tramite lo stesso indirizzo. È il luogo virtuale nel quale un singolo individuo, una società, un'associazione o un'istituzione offrono contenuto informativo al pubblico dei visitatori.

La programmazione web è dipendente dagli obiettivi dell'applicazione sviluppata ed è strettamente legata alla scelta tra Web statico e Web dinamico.

Il primo approccio consiste nella stesura di documenti formattati direttamente in codice HTML, ognuno di questi rappresenta un'unica e distinta pagina del sito web. Nel caso in cui due pagine contengono porzioni identiche di contenuto, come ad esempio il menu, il codice di quest'ultimo sarà ripetuto in entrambi i file HTML. Tutto ciò comporta delle difficoltà anche per piccoli aggiornamenti che richiedono l'intervento di un programmatore esperto.

Il secondo approccio, invece prevede la possibilità di pagine web in grado di aggiornarsi in maniera coerente con le richieste dell'utente.

Il web dinamico fa uso di database e di linguaggi di scripting elaborati dal server.

La programmazione web, pertanto si suddividerà in programmazione web lato client e programmazione web lato server a seconda che ci si occupi dello sviluppo del front-end o del back-end dell'applicazione web. Spesso allo sviluppatore capita di doversi

occupare di entrambe le parti. Applicazioni web abbastanza semplici possono essere costruite anche totalmente con logica lato client (ad es. interamente in JavaScript). La creazione del database con relative tabelle e manutenzione, è spesso opera di un database Administrator (DBA).

1.2 Applicazioni Web

Una *Web Application* è la conseguente evoluzione di un Sito Web. La rete inizialmente si componeva di pagine statiche legate tra loro attraverso dei *link*, ma il potenziale offerto dall' *open source* e la volontà di soddisfare le esigenze di un pubblico sempre più vasto hanno portato gli sviluppatori a creare delle pagine dinamiche, in grado di aggiornarsi nel tempo interagendo con un *database,* e dotate di un'interfaccia intuitiva, tale da semplificarne la consultazione e renderle appetibili in un mercato molto vasto.

1.2.1 Un po' di storia

Dalla metà degli anni 1990, quella della programmazione Web è stata una delle industrie nel mondo che si è sviluppata più velocemente. Nel 1995 c'erano meno di 1.000 compagnie nel settore solo negli Stati Uniti, ma dal 2005 erano già oltre 30.000 avendo avuto un forte sviluppo negli anni 2000 con la diffusione di applicazioni Web. La crescita di questa industria è spinta dalla prospettiva del mercato di vendere prodotti e servizi ai loro clienti e di automatizzare il flusso di lavoro. Il settore della programmazione web è quindi generalmente un settore florido e in continuo sviluppo. Gli sviluppatori web mirano a realizzare applicazioni che tradizionalmente erano disponibili solo in ambiente locale. Questo ha permesso la nascita di nuovi metodi di comunicazione, la decentralizzazione dell'informazione e la distribuzione dei contenuti in rete.

Esempio della trasformazione che la programmazione web ha portato nelle comunicazioni e nel commercio è l'e-commerce (es. eBay). Altro esempio di come lo

sviluppo del software destinato al web abbia apportato nette trasformazioni nel modo di comunicare sono i blog.

Per le grandi imprese e le grandi organizzazioni, i team di web developer si compongono anche di decine di persone, mentre le organizzazioni più piccole possono necessitare di un webmaster singolo.

1.3 Cross-Platform

Lo sviluppo della rete è avvenuto in contemporanea con quello delle tecnologie in grado di accedervi. Oltre ad una varietà di software (browser), in grado di mostrare il contenuto del Web, esistono una moltitudine di dipositivi che gli utenti possono utilizzare indistintamente. Negli ultimi decenni i classici PC dotati di desktop sono stati affiancati dalla tecnologia mobile (*smartphone e tablet*), dalle TV rese smart e ultimamente dai dispositivi indossabili (come *smartwatch*).

1.4 Problematiche dello Sviluppo

L' ambiente multidevice, appena descritto, comporta, per i programmatori e i progettisti, la necessità di progettare applicazioni che siano in grado di adeguarsi alla varietà di hardware a disposizione degli utenti. Bisogna che, innanzitutto i Siti Dinamici siano in grado di adattare il proprio aspetto alla tipologia di schermo su cui sono visualizzati, inoltre risulta necessario che sia possibile sfruttare in ogni caso e nel modo migliore la tecnologia a disposizione (es. geolocalizzazione, camera etc).

I precedenti problemi comportano che lo sviluppatore debba confrontarsi con un'immensa mole di lavoro, sia in fase di progetto e soprattutto nell' obbligatoria ed enorme quantità di test. Nasce pertanto l'esigenza di trovare metodologie che

semplifichino lo sviluppo di un'applicazione web cross-platform evitando di ridurre le funzionalità del prodotto.

Capitolo 2: Linguaggi di Programmazione

2.1 Introduzione ai linguaggi

Il lato Client di un'applicazione per il Web, che vengano utilizzati o meno editor visuali, prevede come base di sviluppo la cooperazione di HTML, CSS e Javascript.

HTML è la base di ogni sito ed è necessario per definirne la *struttura* e quindi gli elementi presenti in essa.

I **CSS (Cascade Style Sheets)** hanno il compito di *razionalizzare e formattare* adeguatamente una pagina web, permettono di agire sui contenuti modificando il modo con cui questi vengono presentati. Il loro utilizzo risulta essere fondamentale per avere una web app con un design adattivo o reattivo.

Javascript viene eseguito direttamente dal Browser Web ed è indispensabile per *rendere la pagina interattiva* senza dover coinvolgere il server. Può essere utilizzato per elaborazioni più o meno complesse e oltre a lavorare sui dati è in grado di modificare anche il contenuto della pagina in ogni suo aspetto.

2.2 HTML

HTML è l'acronimo di Hyper Text Markup Language e non è un linguaggio di programmazione. Si tratta invece di un linguaggio di markup (di 'contrassegno' o 'di marcatura'), che permette di indicare come disporre gli elementi all'interno di una pagina.

Queste indicazioni vengono date attraverso degli appositi marcatori, detti tag ('etichette'), che hanno la caratteristica di essere inclusi tra parentesi angolari (ad es, <img> è il segnaposto di un'immagine).

Con HTML quindi indichiamo, attraverso i tag, quali elementi dovranno apparire su uno schermo e come essi debbano essere disposti. Tutte queste indicazioni sono contenute in un documento HTML, spesso detto "Pagina HTML". Una pagina HTML è rappresentata da un file di testo, ovvero un file che possiamo modificare con programmi come notepad e in genere hanno un nome che finisce con l'estensione .html.

Ciò significa che HTML non possiede i costrutti propri della programmazione, come i meccanismi condizionali, che consentono di reagire in modo diverso a seconda del verificarsi di una condizione, o i costrutti iterativi.

2.2.1 HTML5, Web e mobile

HTML 5 è la versione che nasce appositamente per uscire dal solo ambito Web e diventare piattaforma per la creazione di applicazioni, anche desktop e mobile.

La specifica infatti definisce:

- Una sintassi per il *markup* più efficace e adatta alle esigenze più moderne, con l'introduzione di specifici controlli per i form o degli attributi "data-" da arricchire i tag di informazioni specifiche.
- Una serie di *API* che consentono di gestire, in modo approfondito, aspetti come network, multimedia e hardware dei dispositivi.

Questo sviluppo dello standard ha dato il via alla generazione delle cosiddette App mobile ibride, che sfruttano sia HTML5, per creare apps che si possono distribuire, come quelle native, sui marketplace dei dispositivi più comuni.

2.3 CSS

L'acronimo CSS sta per *Cascading Style Sheets* (fogli di stile a cascata) e designa un linguaggio di stile per i documenti web. I CSS istruiscono un browser o un altro programma utente su come il documento debba essere presentato all'utente, per esempio definendone i

font, i colori, le immagini di sfondo, il layout, il posizionamento delle colonne o di altri elementi sulla pagina, etc.

La storia dei CSS procede su binari paralleli rispetto a quelli di HTML, di cui vuole essere l'ideale complemento. Da sempre infatti, nelle intenzioni degli uomini del W3C, HTML dovrebbe essere visto semplicemente come un linguaggio strutturale, alieno da qualunque scopo attinente la presentazione di un documento.

La specifica CSS3 ha portato l'aggiunta di proprietà tecniche e metodi tarati sulle esigenze reali di chi fa siti web.

I contesti in cui i CSS3 rivelano le loro potenzialità sono:

- I selettori.
- Proprietà e i nuovi metodi per la definizione del colore.
- Proprietà dedicate alla gestione di bordi e sfondi.
- Funzionalità legate al testo e ai font come la possibilità di usare caratteri tipografici non presenti sul computer dell'utente.
- Nuovi modi per impostare il layout (layout multi-colonna, flexible box model).
- La possibilità di servire fogli di stile ad hoc in base alle caratteristiche dei dispositivi (**Media Query**).
- Metodi e tecniche per dare dinamicità alla pagina (transizioni, trasformazioni, animazioni).

2.3.1 Media Query

Le Media Query rappresentano una delle più grandi innovazioni nell'ambito del design di siti web da quando l'utilizzo dei CSS è diventato predominante. Le *media query* permettono di applicare gli stili alle pagine riconoscendo le specifiche di ogni dispositivo su cui viene visualizzato il sito.

Una *media query* è un'espressione logica la cui veridicità indica se applicare o meno le

regole di stile (vero per applicarle o falso per ignorarle).

I parametri delle espressioni sono conosciuti come *media feature* e quello usato più di frequente riguarda le dimensioni del dispositivo o della vista.

Le *media query* estendono la sintassi dei tipi di media utilizzati in CSS 2.1 e HTML 4.01, ovvero, quella che consente di richiamare stili che dipendono dai media, per esempio il collegamento a fogli di stile esterni.

<link rel=”stylesheet” href=”foo.css” media=”screen”>

Il codice nell' esempio richiama il foglio di stile esterno foo.css solo se il dispositivo è uno schermo e non, per esempio, una stampante. La sintassi viene estesa aggiungendo la parola and seguita dalla query fra parentesi.

<link rel=”stylesheet” href=”foo.css” media=”screen and (query)”>

Il codice modificato ha due condizioni: il tipo di medium deve essere lo schermo e l'espressione logica della query deve essere vera. Se si verificano entrambe le condizioni allora si applica foo.css.

Le media query possono essere utilizzate anche per incorporare fogli di stile esterni in altri, tramite @import.

@import url(‘foo.css’) screen and (query);

Il codice presentato può essere usato sia nei tag, sia nei fogli esterni.

Infine, possono essere utilizzate anche inline quando si ha bisogno di applicare un blocco di regole in casi specifici.

@media screen and (query){…}

2.4 Javascript

JavaScript è il linguaggio di scripting più diffuso sul Web, la sua enorme diffusione è dovuta principalmente al fiorire di numerose librerie nate allo scopo di semplificare la programmazione sul browser, ma anche alla nascita di framework lato server e nel mondo mobile che lo supportano come linguaggio principale.

Mentre HTML è un linguaggio di markup, javascript è un linguaggio di programmazione

orientato agli oggetti ed agli eventi. Attraverso Javascript è possibile scrivere veri e propri "programmi" (script) attraverso i quali interagire con l'utente e con il sistema attraverso l'ausilio del browser che, dopo aver scaricato il codice sorgente del programma, ne darà esecuzione. In questo senso si parla di linguaggio di scripting lato client per distinguerlo dai linguaggi lato server dove l'interpretazione del codice è compito del server e non del client. Essendo un linguaggi di programmazione lato client, ciò comporta dei tempi di esecuzione più lunghi, poiché deve essere caricato dal browser prima di poter essere eseguito, e l'impossibilità di eseguire, in modo diretto, operazioni che coinvolgono il server, come la scrittura di dati in un Database. Il rovescio della medaglia è che eseguire programmi anche complessi non comporta alcun sovraccarico della capacità computazionale del server.

Javascript non è un vero e proprio linguaggio di programmazione in grado di creare programmi *stand-alone* perché richiede un browser per eseguire i propri script.

2.5 Esigenze del programmatore

Nel Capitolo 1 è stato scritto che la necessità principale nella realizzazione di Web Application Cross-platform è quella di semplificare il lavoro al programmatore senza dimenticare che il prodotto debba funzionare e si debba presentare in maniera adeguata sulla fascia di dispositivi desiderati.

Lavorare con HTML, CSS e Javascript, richiede numerose competenze se si vuole avere un

uon risultato, soprattutto se ad occuparsi di tutti gli aspetti è un'unica persona, pertanto, a patto di rinunciare in minima parte alla propria libertà, un programmatore web può decidere di ricorrere all' utilizzo di piattaforme web, come i CMS.

Capitolo 3: Content Management System

3.1 Le piattaforme sul Web: CMS

Un CMS, acronimo di *Content Management System,* è un insieme di metodi e tecniche focalizzate alla realizzazione, gestione e distribuzione di siti dinamici che possono accrescere e mutare il proprio contenuto continuamente.

Nei sistemi tradizionali il redattore di contenuti web deve creare tutte le sue pagine dall' inizio alla fine con un editor HTML (o scrivendo direttamente il codice HTML) su un server di sviluppo, provvedere alla pubblicazione e occupandosi in prima persona della manutenzione e dell'archiviazione.

Il Content Management System offre la possibilità di separare la gestione del contenuto del Sito dalla gestione della presentazione di quest'ultimo. Grafico e programmatore si occupano della creazione e della gestione di interfaccia e struttura del progetto, mentre il contenuto può essere modificato da terze parti.

In base a come è stato presentato lo strumento precedentemente risulta evidente che sia possibile approcciarsi in modi differenti all' utilizzo dei CMS.

Un'azienda o un privato interessato ad avere un semplice spazio sul web può sfruttare il sistema analizzato in modo da evitare intermediari nel curare la propria immagine in rete, restringendo i costi, e riuscendo a ottenere attraverso un solo strumento tutti i servizi necessari per avere un sito funzionate, che abbia un aspetto accattivante e la cui gestione risulti notevolmente semplificata attraverso l'utilizzo di un'unica interfaccia polifunzionale.

Un altro gruppo di utenti di un tale Sistema è quello costituito da coloro che sono interessati a sfruttare le potenzialità dei CMS per ciò che riguarda la distribuzione e l'ottimizzazione

dei siti per le varie finestre su cui vengono visualizzati, avendo la liberta e le capacità per accedere e modificare il codice di programmazione.

3.2 I CMS e la programmazione Cross-Platform

L' interesse di questo studio è rivolto alla ricerca del metodo per creare un prodotto funzionante che possa essere distribuito nella maniera migliore sui vari dispositivi, limitando tempi e costi. Un utente esperto di programmazione web, pur essendo conscio che le piattaforme presentate tendono a omologare il prodotto finale per garantirne la stabilità (non necessariamente la sicurezza), ritiene tutto questo un buon punto di partenza da cui muoversi a piacimento e in base alle disponibilità. Gran parte del lavoro dello sviluppatore è destinato alla modifica dei fogli di stile (CSS) che rendono possibile cambiare, in base ai desideri e a seconda delle esigenze, l'aspetto dell'applicazione. I Content Management Systems offrono dei temi standard su cui basare la propria creazione che possono essere modificati utilizzando degli editor visuali. Risulta possibile inserire testi, modificare font e dimensione del carattere, inserire elenchi puntati, inserire foto e immagini, inserire tabelle, scegliere la posizione di testi e immagini e altro. Potendo agire sul codice il programmatore ha la possibilità di correggere ciò che vuole modificandone il design e soprattutto aggiungendo script per introdurre nuove funzioni al fine di ottenere il prodotto desiderato.

3.3 Panoramica sui CMS

Il panorama sui CMS sul web è molto esteso e sicuramente non è possibile averne una visione completa. Non è possibile attuare un confronto vero e proprio tra tutti i servizi poiché alla base hanno le stesse tecnologie e tutti cercano di soddisfare i bisogni dell'utenza.

Alcuni nomi degni di nota sono Wordpress, Joomla, Drupal, Google Sites, Github.io e Magento. Tra tutti primeggia, in modo indiscusso, Wordpress che si è ritagliato un'ampia fetta di mercato fidelizzata. Delle piattaforme menzionate, la maggior parte sono rivolte principalmente al blogging e alla gestione di Siti personali mentre altre devono la loro fama perché alla base di siti di e*commerce* senza però dimenticare il fatto che l'utente può

scegliere un CMS piuttosto di un altro e servirsi dei *templates* per raggiungere le proprie finalità.

Un Sistema di Gestione può inoltre offrire il servizio di hosting o meno, il che rappresenta un'ulteriore semplificazione per il procedimento di pubblicazione di una pagina per chi non ha dimestichezza con tali processi.

3.3.1 WordPress

Nato nel 2003 come piattaforma di blogging, oggi WordPress permette di sviluppare ogni tipo di sito Web, dai blog personali, agli e-commerce più avanzati; il CMS ha infatti alla base un framework che ne ha fatto un ambiente di sviluppo per ogni sorta di applicazione Web. Le sue caratteristiche fondamentali sono la semplicità d'uso e la flessibilità della struttura a vantaggio degli sviluppatori. La complessità viene gestita dal framework, che rende disponibile un insieme di API tramite le quali è possibile estendere WordPress.

Le funzionalità core di WordPress possono essere ampliate grazie alle migliaia di estensioni gratuite e commerciali.

Istallare WordPress e pubblicare contenuti online è questione di pochi attimi. L'architettura di base dispone di post per i contenuti dinamici e pagine per i contenuti statici, ma questa impostazione può rapidamente essere estesa con tipi di contenuti personalizzati, metadati e tassonomie. È possibile infatti importare i contenuti da altre piattaforme.

Non tutti gli utenti del sito hanno gli stessi privilegi, e WordPress, come ogni CMS avanzato, permette di assegnare capacità specifiche ad ogni ruolo utente. Di base, WordPress dispone di sei ruoli predefiniti:

- Administrator.
- Editor.
- Author.
- Contributor.
- Subscriber.

Il set di ruoli non è rigido e grazie ai plugin l'utente può aggiungerne di nuovi e modificare le combinazioni di capacità assegnate ad ognuno di essi.

Il sistema di gestione di immagini e file mediali è semplice nell'utilizzo e avanzato nelle funzionalità. Per l'utente caricare media è un'operazione immediata grazie al supporto del Drag & Drop e a funzionalità di editing che permettono di modificare le immagini ridimensionandole e tagliandole tramite la comoda interfaccia grafica del pannello di amministrazione in base alle più diverse esigenze.

I temi sono le estensioni grafiche di WordPress. Esistono migliaia di temi gratuiti e a licenza commerciale, sviluppati per esigenze generiche e per obiettivi specifici.

I plugin sono invece le estensioni funzionali di WordPress. Come per i temi, esistono migliaia di plugin gratuiti e commerciali, che permettono a WordPress di gestire eventi, località geografiche, archivi di ogni tipo, negozi virtuali e quant'altro. WordPress notifica la disponibilità di nuove versioni di temi e plugin, e permette di prelevarli dalla repository senza mai abbandonare il pannello di amministrazione.

WordPress dispone di un sistema di commenti integrato. Nessun plugin è necessario per mettere in opera un sito dotato di un sistema di discussione completo ed estremamente semplice da gestire.

WordPress nasce ottimizzato per i motori di ricerca, anche se l'amministratore del sito può migliorare questo aspetto grazie ai numerosi plugin disponibili nella repository. La lingua base è l'inglese, ma WordPress è disponibile in oltre 70 lingue.

WordPress è un software Open Source rilasciato con licenza GPL la quale prevede che ogni opera derivata debba essere rilasciata con lo stesso tipo di licenza. Ciò significa che si può estendere il CMS con temi e plugin, i quali possono essere ceduti commercialmente ma devono essere distribuiti sotto GPL. Se WordPress è il CMS più diffuso al mondo lo si deve anche all'attiva comunità di utenti e sviluppatori.

Il sistema di API di WordPress presenta circa una ventina di interfacce costituite da metodi e funzioni che eseguono operazioni specifiche su database, file system, dati e UI. Il sistema

di API permette di sviluppare plugin e temi in modo indipendente aggiungendo funzionalità e servizi terzi al core del CMS.

Grazie all'Application framework di WordPress lo sviluppatore può dedicarsi esclusivamente alle funzionalità core delle proprie applicazioni, senza riprogettare il sistema di gestione degli utenti, localizzazione dell'ambiente di lavoro, richieste HTTP, accesso al database e gestione delle sessioni.

WordPress evolve di continuo, e gli ultimi sviluppi vedono il CMS diventare un framework di sviluppo.

3.3.2 Joomla

Joomla è tra i più noti CMS e nasce nel 2005 sviluppato da Team di Joomla.

Come per tutti i software open source di successo, la forza di Joomla sta nella comunità. Una community molto grande, compatta e laboriosa, pronta ad effettuare test e riportare bug, disposta a tradurre il CMS in moltissime lingue e a collaborare nella realizzazione di temi grafici o componenti aggiuntivi che potenziano questo applicativo.

Joomla è uno strumento che permette la creazione e la pubblicazione di siti Internet dinamici, in maniera semplice e veloce, ma anche con grandi potenzialità e sicurezza. Un pannello di controllo ricco di icone e con grafica accattivante ci guida nell'inserimento dei contenuti e nella configurazione delle caratteristiche del sito, fra sondaggi e notizie, gallerie fotografiche, blog e molto altro. Tutte queste operazioni possono essere effettuate senza scrivere o modificare una riga di codice.

Particolare importanza poi riveste la presenza di un editor integrato, la cui interfaccia utente, simile a quella delle popolarissime applicazioni "Office", aiuta l'utente a creare agevolmente i contenuti che intende realizzare.

3.3.3 Drupal

Drupal è una piattaforma open source, un software che aiuta utenti singoli o grandi comunità a pubblicare, amministrare e organizzare grandi quantità e varietà di contenuti in un sito web, in maniera semplice e veloce. Drupal è interamente sviluppato in PHP e permette di

realizzare una gamma molto varia di applicazioni Web:

- Portali e community di qualsiasi dimensione
- Siti Intranet
- Blog, blog multiutente
- Directory di contenuti
- e-commerce
- Siti istituzionali (è stato utilizzato anche dall'università di Harvard, dal governo del Belgio e per alcuni siti della NASA)

Drupal è tra i CMS Open Source più performanti e flessibili disponibili sul mercato. Con le ultime versioni, sono migliorate molte cose, come ad esempio l'installazione.
L' interfacci di amministrazione presenta controlli avanzati che permettono, ad esempio, di disegnare il layout dei blocchi attraverso il drag and drop.
Come per gli altri prodotti open source, la presenza di una community di sviluppatori attiva garantisce lo sviluppo continuo di nuove funzionalità per la piattaforma.
In definitiva Drupal offre stabilità, sicurezza e performance, e inoltre è molto semplice nell' utilizzo.

3.3.4 Magento

Magento è un CMS dedicato al commercio elettronico utilizzato da oltre 250'000 commercianti in tutto il mondo. È stato rilasciato la prima volta nel 2008 dalla Varien, che lo ha realizzato sfruttano Zend, un noto framework PHP open source per lo sviluppo di applicazioni web. Il rilascio era sotto licenza BSD. Magento è disponibile in due forme, la prima liberamente scaricabile, mentre la seconda forma è a pagamento, "Magento Enterprise Edition", ed è una versione per aziende di grandi dimensioni, rilasciata nell'aprile del 2009.
Questa piattaforma garantisce una serie di vantaggi poiché include potenti strumenti di marketing e di gestione del catalogo prodotti, ed è un sistema estremamente versatile e perfetto per l'ottimizzazione SEO. Le funzioni che lo differenziano da altri CMS e lo

rendono uno tra i più scaricati per il commercio elettronico sono:

- Possibilità di confronto tra i prodotti
- Inserimento di recensioni sul prodotto
- Schede con immagini multiple
- Immagini con possibilità di zoom
- Visualizzazione di report e la gestione degli ordini
- Batch di importazione e di esportazione di catalogo
- Personalizzazione degli account
- Spedizione di mail per l'ordine a più indirizzi contemporaneamente
- Inserimento di sconti, promozioni e prezzi speciali
- Possibilità di negozi online multipli
- Supporto per più valute
- Supporto per più lingue

3.3.5 Google Sites

Google Sites è il servizio che Google mette a disposizione gratuitamente a tutti gli utenti registrati e che consente di realizzare dei siti utilizzando un sistema guidato, un'interfaccia interattiva molto semplice e una serie di templates che, combinati tra loro, rendono possibile anche ai meno esperti pubblicare in pochi minuti siti Web di ottima fattura.

Tra i numerosi punti di forza di Google Sites si conta la possibilità di creare un numero infinito di siti strutturati a nostro piacimento, non solo dal punto di vista estetico ma, soprattutto, per quanto riguarda le funzionalità e le architetture di disposizione dei contenuti. Google Sites si dimostra uno strumento molto versatile con il quale realizzare progetti della più varia natura. Possiamo infatti aggiungere pagine di varia tipologia e, all'interno di queste pagine, predisporre sistemi di navigazione o interazione che vanno dalla classica presentazione di contenuti multimediali e testuali a previsioni del tempo, fogli di calcolo,

gallerie video, strumenti di collaborazione a distanza e infiniti altri elementi.

3.3.6 Github.io

Github è un social network dedicato ai programmatori. Grazie a github è possibile gestire progetti online, mantenere una copia sul server e visionarla online.

Inoltre la piattaforma permette l'hosting di un sito web statico. Una pagina GitHub permette di "hostare" direttamente da una repository di GitHub una pagina personale, di un'organizzazione o di un progetto. Questa piattaforma diversamente dalle altre piattaforme permette la gestione di un sito web statico non dotato di codice server-side come PHP.

Capitolo 4: App Ibride

4.1 Web App e App Ibride

Dopo aver analizzato alcune delle tecnologie che possono essere utilizzate per la creazione di Siti Web aperti o Web app specifiche risulta utile guardare ai processi che ci permettono di creare applicazioni che possono essere rese disponibili su store online e eventualmente installate su un dispositivo.

Bisogna, in primo luogo fare una distinzione tra le applicazioni in base al loro processo di creazione.

- Le *app native* sono create grazie a tecnologie per piattaforme specifiche non web, come OS X, iOS, Windows e Android.
- Le *web app* utilizzano i linguaggi della piattaforma web e possono includere siti ospitati online oppure all' interno di contenitori compressi installati nel device.
- Le *app ibride* utilizzano tecnologie web, ma sono inserite in contenitori nativi.

4.1.1 Applicazioni Ibride : Perche ?

Le Applicazioni Ibride sono un compromesso tra la programmazione nativa e quella multipiattaforma cercando di sfruttare i vantaggi di entrambe. I linguaggi propri della rete permettono di progettare il contenuto delle schermate delle Apps in maniera efficiente ma non garantiscono l'interazione con i dispositivi sui quali le applicazioni sono avviate. La scelta di racchiudere una web app in un contenitore nativo permette un'intermediazione con le risorse del dispositivo utilizzato per eseguirla. L'aumento dei tempi e dei costi della produzione nello sviluppo di un'app ibrida non sono certamente paragonabili a quelli della

produzione di un app nativa. Far ricorso all' approccio descritto significa, per uno sviluppatore di applicazioni, avere la possibilità di testare il proprio prodotto su più livelli. Infatti l'utilizzo di tecnologie per il web permette di mettere a disposizione del pubblico l'applicazione, quanto prima possibile, ed è, quindi, possibile valutare preventivamente un successo o un fallimento con perdite ridotte. "Impacchettare" poi il prodotto in un *wrapper* nativo permette di ottimizzarne il funzionamento e migliorarne l'accessibilità.

4.2 Design Responsive e Adaptive

Gli approcci e i metodi per pianificare come il contenuto, descritto dal codice HTML, deve essere visualizzato sui vari dispositivi sono diversi in circolazione. Le soluzioni più diffuse nella community degli sviluppatori sono quella *adaptive (adattivo)* e quella *responsive (reattivo)*.

Un sito reattivo si adatta esclusivamente alla forma e dimensione dello schermo del dispositivo mentre una *soluzione adattiva* è ottimizzata completamente per i device mobili, garantendo una migliore *user experience*.

Le differenze tra i dispositivi utilizzati per la consultazione di siti sono anche dovute alle diverse modalità di utilizzo. Risulta necessario, pertanto, non potendo creare un sito diverso per ogni singolo strumento, cercare di adattare il sito alle caratter1istiche uniche del device. Tutto ciò si può ottenere sfruttando le tecnologie HTML5, CSS3 i frameworks e le Media Query.

4.3 App Hosted o Packaged

Un'app di tipo hosted conserva tutti i file su un server web esterno, di solito accessibile attraverso un URL pubblico. L'app hosted non è installata sul dispositivo ma si crea un collegamento che avvia il browser o una vista web integrata quando l'utente decide di aprirla. Una soluzione di questo tipo comporta dei vantaggi in fase di aggiornamento per lo sviluppatore ma il rischio per la sicurezza è maggiore, app di questo genere solitamente non hanno accesso alle API dei dispositivi.

Le app di tipo packaged contengono tutto il necessario per l'esecuzione al loro interno, compresso in un file unico. Tutti i file sono installati sul dispositivo, e anche se il device può connettersi a servizi web esterni, non è sempre necessaria una connessione. Confezionare un'app in questo modo permette ai gestori dello store di controllarne e autorizzarne il contenuto, e garantire permessi extra se questo è ritenuto sicuro. Tutto ciò significa che lo sviluppatore può accedere alle API riservate, come quelle per l'accesso alla rubrica e alle funzioni di messaggistica.

Le app di tipo packaged si aprono e si chiudono più velocemente di quelle hosted, essendo i file tutti nel dispositivo, e dipendono meno dallo stato della rete. I dati devono essere però archiviati in locale e poi sincronizzati non appena si va online.

4.4 PhoneGap/ Apache Cordova

Nel caso in cui si voglia pubblicare una web app negli store dei device più popolari, bisogna creare un'app ibrida. Queste app sono simili a quelle di tipo packaged, perché le risorse sono tutte contenute in un archivio singolo, ma hanno in più una shell nativa, o wrapper, che assicura l'integrazione con il sistema operativo principale, fornendo sicurezza e performance migliori, oltre a permettere l'accesso alle API riservate. Tra i molti software per la creazione di app ibride uno dei più comuni e più semplici da usare è PhoneGap.

PhoneGap è un software di proprietà di Adobe ma è comunque un software open source gratuito,nato sulla base del progetto Apache Cordova, che permette di creare app mobile semi-native con tecnologie web.

PhoneGap è multipiattaforma e uno dei suoi punti forti è il fatto di essere un wrapper nativo che ospita il codice di una piattaforma web, consentendo l'accesso alle API di un device, che non sono sempre disponibili attraverso il browser. Per fare questo PhoneGap utilizza le proprie API, che corrispondono a quelle standard di ogni device se presenti.

Il software, per essere configurato, necessita degli SDK di ogni device target e in alcuni casi del certificato di sviluppatore. Completata la configurazione si può iniziare un nuovo

progetto che crea una struttura di cartelle con alcuni file compresi quelli per accedere alle API.

4.4.1 Plugin APIs

Cordova porta con se un set minimale di API, ed è possibile aggiungere ai progetti le API di cui hanno bisogno attraverso dei plugins. Un Registro dei plugin permette di effettuare una ricerca tra tutti quelli esistenti provenienti anche da terze parti. Di seguito è riportata una lista dei plugins di Cordova principali:

- **Battery Status**: Monitora lo stato della batteria del dispositivo.
- **Camera**: Permette di accedere alla camera del dispositivo.
- **Console**: Aggiunge capacità addizionali per il *console.log()*.
- **Contacts**: Lavora con il database dei contatti del dispositivo.
- **Device**: Raccoglie informazioni specifiche del dispositivo.
- **Device Motion/Accelerometer**: Attinge dati dai sensori di movimento del dispositivo.
- **Device Orientation/Compass**: Ottiene la direzione puntata dal dispositivo.
- **Dialogs**: Notifiche visuali del dispositivo.
- **FileSystem**: Si aggancia al File System nativo attraverso Javascript.
- **File Transfer**: Si aggancia al File System nativo attraverso Javascript.
- **Geolocation**: Informa l'applicazione sulla posizione.
- **Globalization**: Permette di ottenere informazioni sulla località in cui si trova l'utente.
- **InAppBrowser**: Carica URLs in un istanza di browser in-app.
- **Media**: Registra e riproduce file audio.
- **Media Capture**: Cattura file multimediali attraverso applicazioni specifiche del dispositivo.
- **Network Information/Connection**: Permette un rapido controllo dello stato della rete e delle informazioni sulla connessione del dispositivo.
- **Splashscreen**: Mostra e nasconde splash screen delle applicazioni.

- **Vibration**: Un API per far vibrare il dispositivo.
- **StatusBar**: Un API che permette di mostrare, nascondere e configurare una barra di stato di sfondo.

Capitolo 5: Esempi di utilizzo degli strumenti

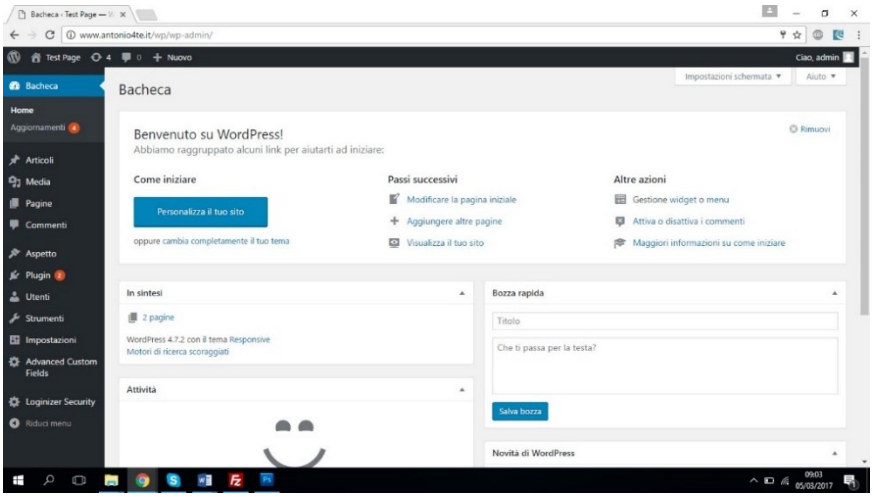
5.1 Presentazione

In questo capitolo sono presentati semplici esempi che dimostrano l'utilizzo di alcuni degli strumenti analizzati nei precedenti capitoli. Nella pratica è stato realizzato un prototipo di un'applicazione web utilizzando un Content management System e poi servendosi della piattaforma PhoneGap/Apache Cordova e del template JQuery Mobile. L' applicazione che si vuole realizzare prevede due schermate, la prima contenente un form e la seconda contenente una mappa, visualizzata per mezzo di uno script che individua, se possibile, la posizione dell'utente aggiungendo un marker sulla mappa. Lo scopo di questo piccolo progetto è quello di valutare, anche se in maniera superficiale, i diversi processi produttivi.

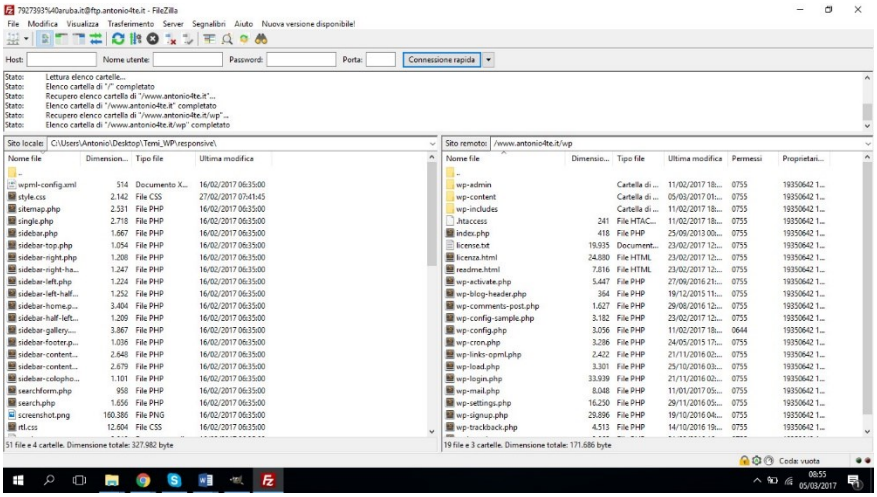
5.2 Utilizzo dei CMS

Nel caso particolare si è scelto di utilizzate la piattaforma offerta da WordPress, sia per la sua semplicità di utilizzo, sia perché attualmente rappresenta una delle più utilizzate per la gestione di Siti web.

Dopo aver installato WP nel proprio spazio web si potrà accedere attraverso un browser al pannello di controllo dell'admin del sito o si potrà lavorare in locale, trasferendo successivamente attraverso un Client FTP i file sul web.



Intefaccia di amministrazione WordPres.



Schermata principale Filezilla.

Una delle scelte fondamentali quando si utilizza WP è quella del Tema che rappresenta la maniera in cui l'utente visualizzerà il contenuto delle pagine. Nel caso in esame è stato scelto il tema Responsive 2.0, gratuito e funzionale per garantire che l'applicazione si adatti a finestre di diversi dispositivi.

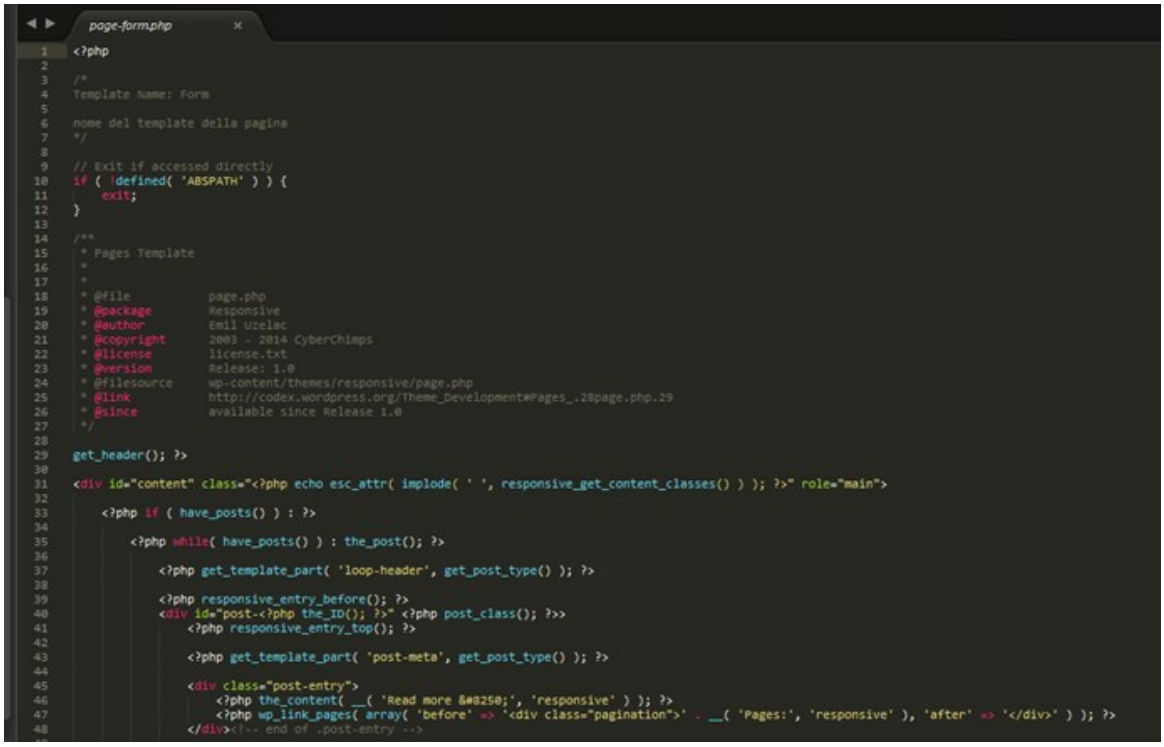
Il piccolo progetto si compone di due pagine:

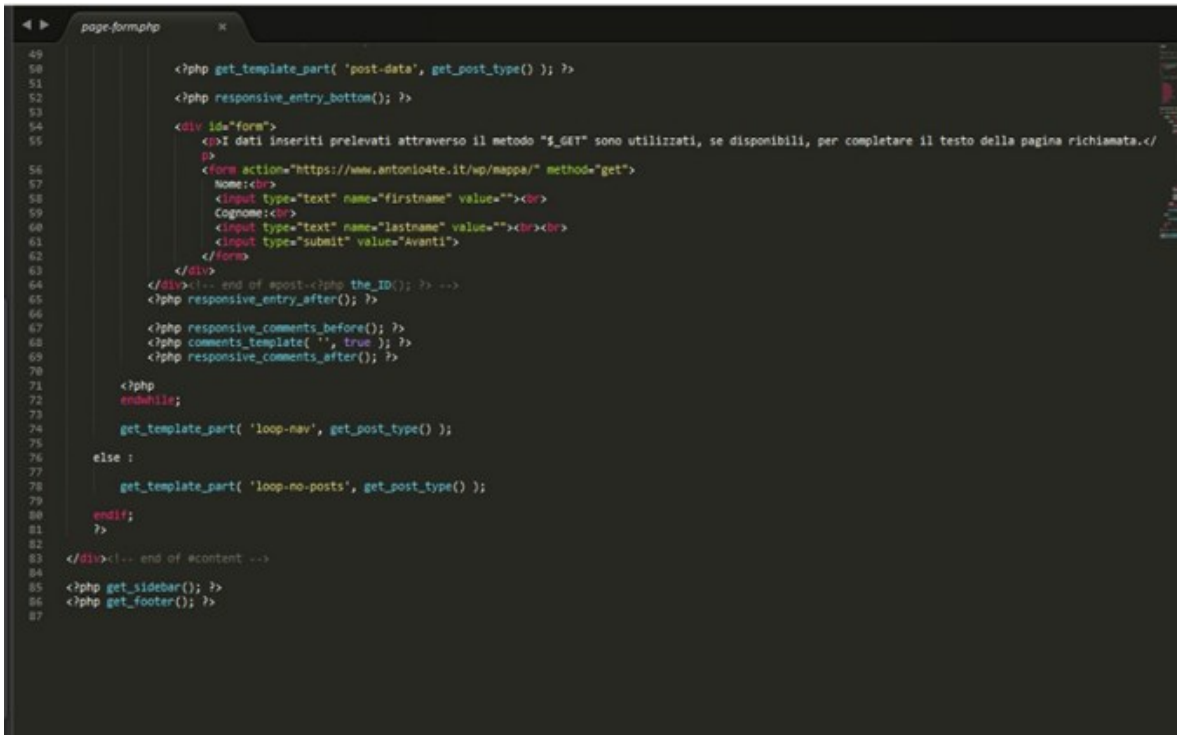
- Schermata iniziale costituita da un elemento "form".
- Pagina secondaria contenente una mappa in grado di visualizzare la posizione dell'utente.

La struttura di entrambe le pagine è stata modificata agendo direttamente, in locale sui file contenuti nella cartella del template. Una volta aperta la cartella, sono state create due copie del file *"page.php"*, rinominate come *"page-form.php"* e *"page-mappa.php"*.

Contenuto della cartella template.

Nella prima pagina è stata modificata la direttiva *"Template Name"* in modo tale che WP riconosca la differenza con la struttura di default e in seguito nell' area destinata al contenuto sono state aggiunte le righe di HTML necessarie a generare un "form" che richiede il Nome e il Cognome dell'utente con un bottone necessario per il passaggio alla pagina successiva. In questo caso i dati di input possono essere prelevati e utilizzati nella pagina successiva.





Entrambe le figure mostrano il codice del file page-form.php.

Nella seconda pagina, dopo aver apportato al modifica al *"Template Name"*, sono stati inseriti nell' area del contenuto due elementi "div". Il primo elemento contiene del testo e del codice *php* in grado di utilizzare, se disponibili, i dati prelevati attraverso il *form*, mentre il secondo elemento possiede esclusivamente l'attributo "*mappa*".

Il render della mappa è possibile attraverso l'aggiunta e l'esecuzione di uno script *javascript* che sfrutta l'API di Google Maps anch' essa aggiunta al progetto.

```
function get_position(position)
{
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;

    var miaPosizione = new google.maps.LatLng(latitude,longitude);

    var settings = {
   zoom: 15,
   center: miaPosizione,
   mapTypeControl: true,
   mapTypeControlOptions: {style: google.maps.MapTypeControlStyle.DROPDOWN_MENU},
   navigationControl: true,
   navigationControlOptions: {style: google.maps.NavigationControlStyle.SMALL},
   mapTypeId: google.maps.MapTypeId.ROADMAP};

    var map = new google.maps.Map(document.getElementById("map"), settings);

marker = new google.maps.Marker({
position: miaPosizione,
map: map,
title: "La tua posizione"
 });

}

if(navigator.geolocation)
{

   navigator.geolocation.getCurrentPosition(get_position,get_error,{'enableHighAccuracy':true,'timeout':30000,'maximumAge':0});
}
else
{
    alert('Il browser non supporta la geolocalizzazione');
}
```

Script che permette di ottenere i dati sulla posizione e di renderizzare la mappa.

Il file contente lo script e l'API necessari sono richiamati nel file "*functions.php*" attraverso la funzione "*wp_enqueue_script()*".

```
function my_theme_add_scripts() {
    wp_enqueue_script( 'google-map', 'https://maps.googleapis.com/maps/api/js?key=AIzaSyBA2XhIvR9Vw1N7Md3qtdmJfdi0CpiI3v8', array(), '3', true );
    wp_enqueue_script( 'google-map-init', get_template_directory_uri() . '/core/js/google-maps-geo.js', array('google-map', 'jquery'), '0.1', true );
}

add_action( 'wp_enqueue_scripts', 'my_theme_add_scripts' );


function my_acf_google_map_api( $api ){

    $api['key'] = 'AIzaSyBA2XhIvR9Vw1N7Md3qtdmJfdi0CpiI3v8';

    return $api;

}

add_filter('acf/fields/google_map/api', 'my_acf_google_map_api');
```

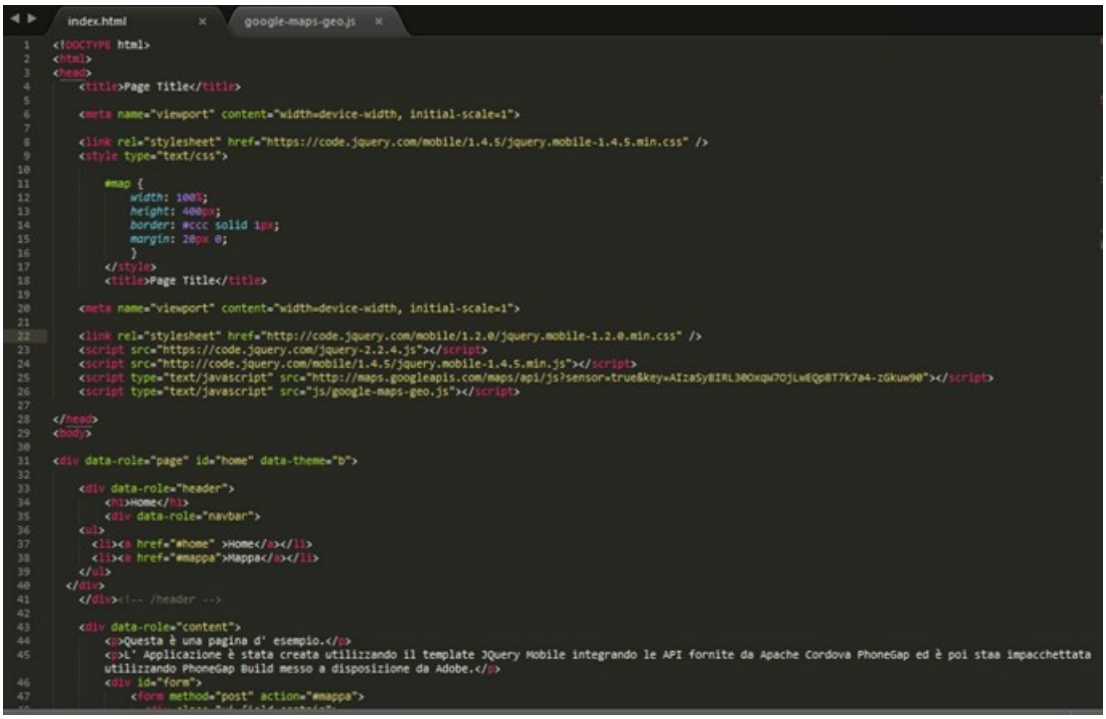
Codice aggiunto al file functions.php.

Una volta caricati i file modificati è stato sufficiente creare attraverso l'interfaccia di WordPress due nuove pagine alle quali assegnare i *Templates* modificati e rinominati. Risulta possibile ora accedere al sito da diversi dispositivi con discreti risultati a patto che si dia la possibilità al sito di accedere ai dati relativi alla posizione dell'utente.

5.3 Utilizzo di PhoneGap/ Cordova

La realizzazione dell'applicazione in questo caso deve essere preceduta da una fase di configurazione dell'ambiente di sviluppo leggermente più complessa. Risulta necessario infatti installare sul proprio PC Apache Cordova e gli strumenti di sviluppo relativi alle piattaforme alle quali è destinata l'applicazione, inoltre dovremmo registrarci su PhoneGap Build per racchiudere l'app nel contenitore nativo. Tutte le procedure possono anche essere eseguite direttamente da terminale.

Una volta creato il progetto con l'apposito comando da terminale, si ha a disposizione una cartella contenente tutti i file necessari per la realizzazione dell'applicazione (*$ cordova create path/to/myApp "com.example.app" "My App"*). All' interno del file "*index.html*" all' interno della cartella "*www*" del progetto è stato creato lo scheletro, composto da *head* e *body*, di un normale sito web. All' interno del tag *head* sono stati inseriti i link necessari ad integrare il framework di JQuery Mobile. Il framework su cui si basa l'applicazione permette di definire diverse pagine all' interno dello stesso documento html utilizzando nel modo adeguato l'attributo "*data-role*". Dopo aver definito due pagine, nella prima sono stati inseriti un *navbar* e un *form* fittizio mentre nella seconda pagina è stato inserito un *div* destinato a contenere una mappa con la posizione dell'utente. Successivamente è stato aggiunto al progetto lo script necessario a visualizzare la mappa, è stata integrata l'API di Google Maps e prima del tag di chiusura del *body* è stato integrato il file "*cordova.js*" disponibile solo dopo la compilazione del progetto.



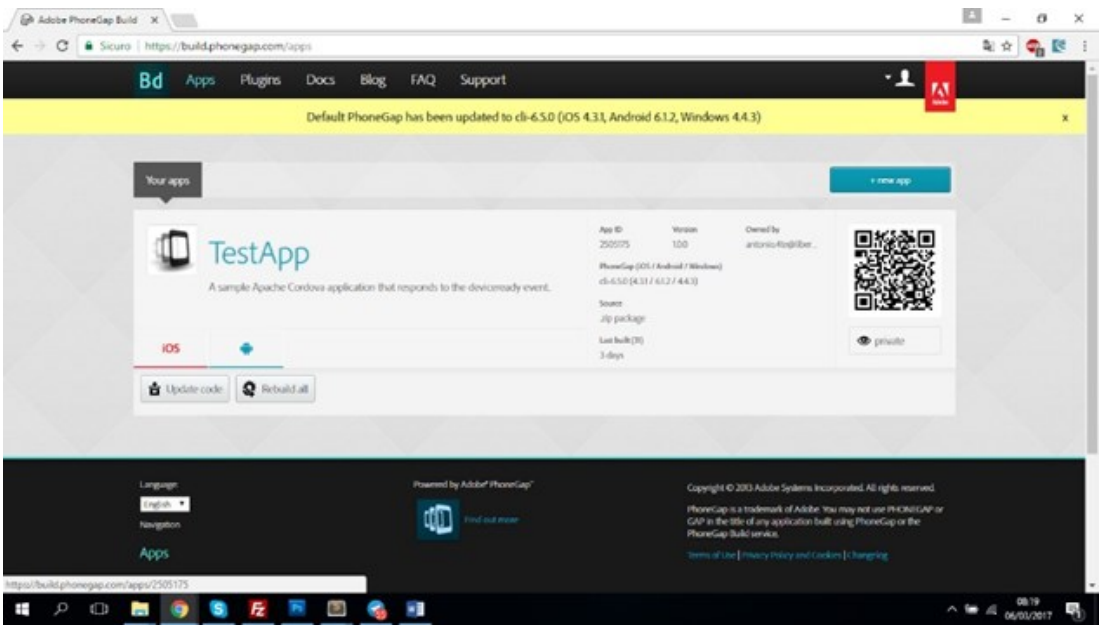
Le figure mostrano il contenuto del file index.html che stabilisce la struttura dell'applicazione.

Attraverso il terminale è stato poi aggiunto il plugin offerto da Cordova per la geolocalizzazione e il progetto è stato compilato. L' ultimo passo per la realizzazione dell'app è consistito nell' "impacchettamento" nel contenitore nativo, ottenuto caricando sul Cloud di PhoneGap build l'archivio contenente i files della cartella "*www*" con l'aggiunta del file "*config.xml*".

```xml
<?xml version='1.0' encoding='utf-8'?>
<widget id="it.antonio4te.test" version="1.0.0" xmlns="http://www.w3.org/ns/widgets" xmlns:cdv="http://cordova.apache.org/ns/1.0">
    <name>TestApp</name>
    <description>
        A sample Apache Cordova application that responds to the deviceready event.
    </description>
    <author email="dev@cordova.apache.org" href="http://cordova.io">
        Apache Cordova Team
    </author>
    <content src="index.html" />
    <plugin name="cordova-plugin-whitelist" spec="1" />
    <plugin name="cordova-plugin-geolocation"/>
    <access origin="*" />
    <allow-intent href="http://*/*" />
    <allow-intent href="https://*/*" />
    <allow-intent href="tel:*" />
    <allow-intent href="sms:*" />
    <allow-intent href="mailto:*" />
    <allow-intent href="geo:*" />
    <platform name="android">
        <allow-intent href="market:*" />
    </platform>
    <platform name="ios">
        <allow-intent href="itms:*" />
        <allow-intent href="itms-apps:*" />
    </platform>
</widget>

```

File config.xml che mostra le direttive per la build del progetto.



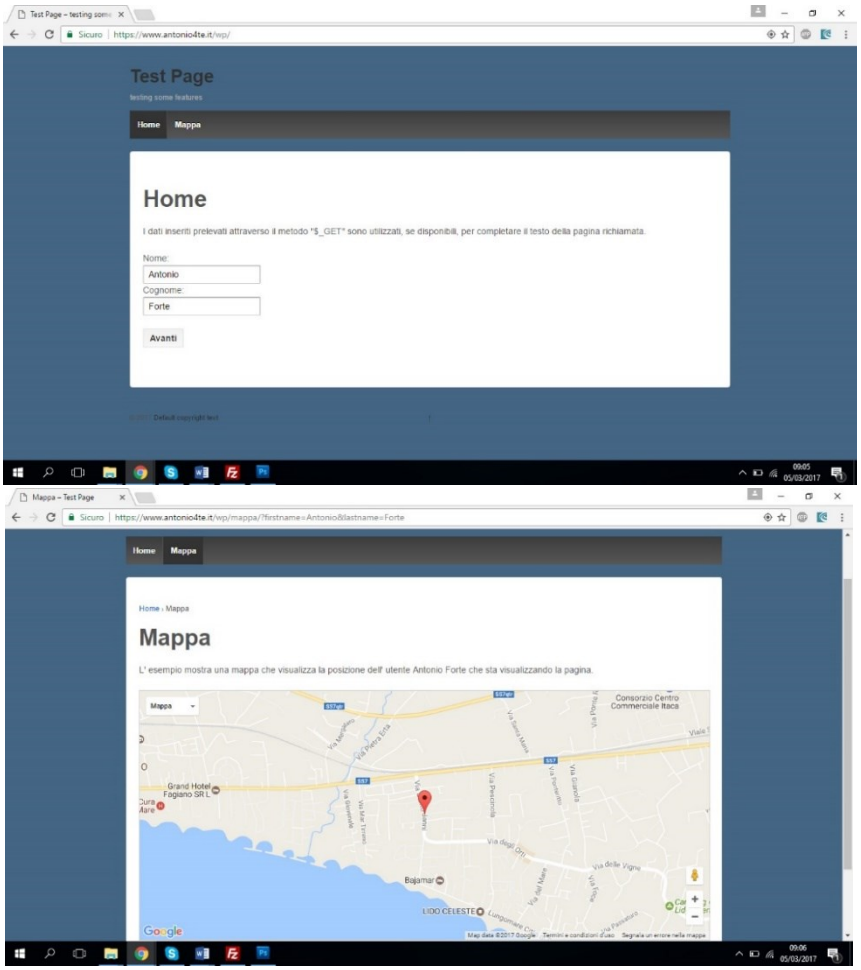
Interfaccia di PhoneGap build con cui è possibile "impacchettare" le app.

5.4 Risultati

L'applicazione degli strumenti ha portato a risultati soddisfacenti, sebbene gli esperimenti fossero molto semplici e in seguito sono riportati alcuni test su alcuni dispositivi differenti.

WordPress possiede un'interfaccia intuitiva, pertanto la creazione e la pubblicazione della web app sono molto semplici, ma non è altrettanto semplice modificare il codice a proprio piacimento evitando di compromettere il funzionamento. Un ulteriore vantaggio del CMS è la gestione integrata di un database dal quale prelevare le informazioni per riempire le pagine. Di contro l'applicazione creata con WordPress necessita di un browser per poter essere eseguita (https://www.antonio4te.it/wp).

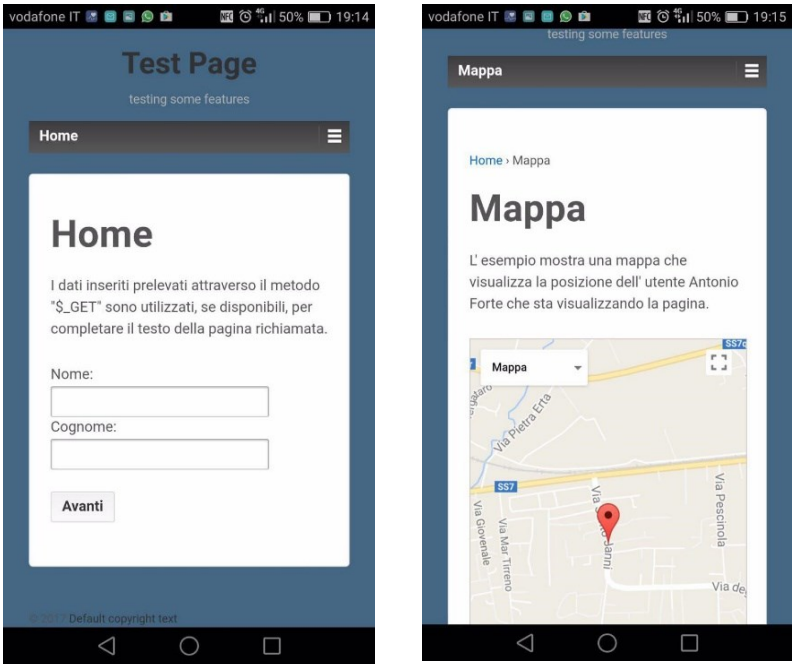
L' applicazione è stata testata su un pc da 15,6', su uno Xiaomi Note 3(5,5'), su un Iphone 5s e su un Honor(5,5').



Test su PC.

Test su Xiaomi Note 3.

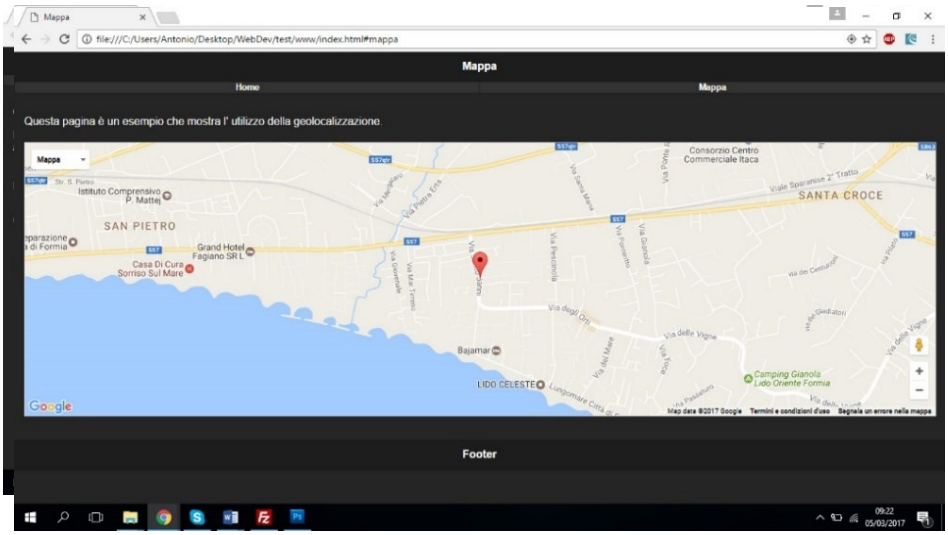
Test su Iphone 5s.



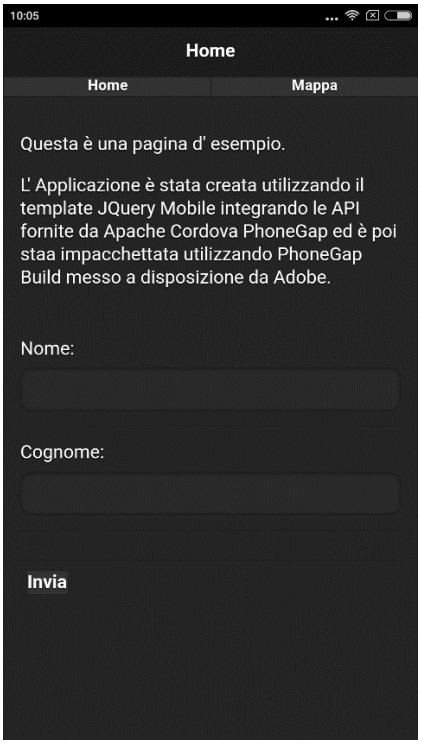
Test su Honor.

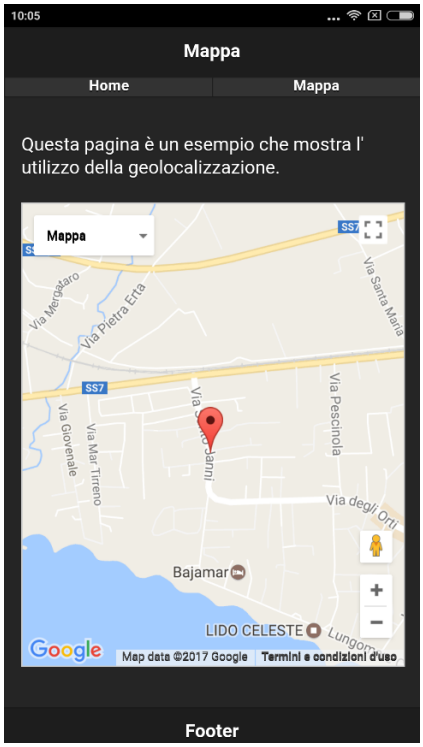
Il lavoro svolto PhoneGap/Cordova ci permette di avere pieno controllo sul codice inoltre l'applicazione creata può essere sia caricata direttamente su un web server sia racchiusa in un wrapper nativo. La gestione autonoma del codice, anche servendosi di un framework, impone allo sviluppatore di curare ogni aspetto dell'applicazione, compresa la gestione di un eventuale database o della memorizzazione di dati in locale, con un conseguente aumento dei tempi di produzione.

La Web app è stata testata su pc in locale e sullo Xiaomi Note 3(5,5') con Sistema Android, non avendo i requisiti da sviluppatore IOS.



Test in locale.





Test su Xiaomi Note 3.

5.5 Conclusioni

Come ribadito più volte nella trattazione, sembra evidente che non esista un criterio oggettivo di cui servirsi nella scelta di una metodologia di sviluppo per applicazioni web cross-platform. Procedere in una direzione o in un'altra dipende da diversi fattori, quali scadenze, costi, personale a disposizione o anche la semplice esperienza del programmatore che lo porta a preferire una strada piuttosto che un'altra. Un altro parametro di cui si deve tener conto è il parere del pubblico al quale è destinato il prodotto, che nel campo Web rappresenta uno dei principali banchi di prova dell'applicazione. L'utilizzo di framework specifici come PhoneGap rappresenta una soluzione più affidabile rispetto all' utilizzo dei CMS che, se pur facilitano il compito del programmatore, non tengono conto del sistema su cui è lanciata l'applicazione creata. Risulta importante infatti garantire che la web app sia in grado di prelevare in maniera corretta i dati dei sensori della macchina specifica su cui viene eseguita soprattutto nel caso in cui tali informazioni sono indispensabili per garantire il corretto funzionamento del servizio fornito.

Bibliografia

[1] Peter Gasston, Sviluppare applicazioni web multi-device *con HTML, CSS3 e Javascript*, APOGEO , 2013, 218.

[2] HTML.it, "http://www.html.it/development/", 14/03/2017

[3] WordPress.org, "https://codex.wordpress.org/it:Documentazione_per_sviluppatori", 14/03/2017

[4] Apache Cordova Documentation, "http://cordova.apache.org/docs/en/latest/", 14/03/2017

[5] PhoneGap Docs, "http://docs.phonegap.com/", 14/03/2017

[6] Drupal.it, "http://www.drupal.it/home-documentazione", 14/03/2017

[7] Magento Tech Resources, "https://magento.com/resources/technical", 14/03/2017

[8] Joomla! Documentation, "https://docs.joomla.org/Main_Page", 14/03/2017

[9] GitHub Help, "https://help.github.com/categories/github-pages-basics/", 14/03/2017

[10] Guida di Sites, "https://support.google.com/sites#topic=6372850", 14/03/2017

[11] 1and1, "https://www.1and1.it/digitalguide/siti-web/programmazione-del-sito-web/diversi-tipi-di-app-che-cose-una-web-app/", 14/03/2017

[12] TECNOTECA, "http://www.tecnoteca.com/it/azienda/risorse/tecnopedia/cms-content-management-system", 14/03/2017

[13] FormGet, "https://www.formget.com/phonegap/", 14/03/2017

[14] JQuery mobile, "https://demos.jquerymobile.com/1.2.0/", 14/03/2017