UNIVERSITÀ DI PISA

Corso di Laurea Triennale in Informatica Umanistica

Tesi di Laurea

UN MODELLO DI SITO RESPONSIVE PER I DIPARTIMENTI DELL'UNIVERSITÀ DI PISA

Relatore:

Prof.ssa MARIA SIMI

Laureando:

VINCENZO RIZZA

ANNO ACCADEMICO 2013-2014

Alla mia famiglia e ai miei amici

Sommario

L'argomento di questa tesi è la presentazione di un modello di sito responsive per i dipartimenti e i corsi di laurea dell'Università di Pisa.

Vengono introdotte le tecniche più comuni per la realizzazione di interfacce web fluide e viene discussa l'importanza della realizzazione di siti web multi-devices. Dopo la storia e lo stato dell'arte del responsive design, vengono illustrati gli strumenti utilizzati per raggiungere gli obiettivi fissati e per risolvere i problemi rispetto al modello di sito non-responsive, punto di partenza del lavoro. Viene successivamente presentata la struttura del layout fluido prodotto e vengono analizzate tutte le componenti dell'interfaccia e il loro comportamento alle diverse risoluzioni degli schermi dei dispositivi. Il pacchetto rilasciato contiene i file HTML e CSS, le linee guida da seguire per eventuali modifiche future e un tema Wordpress. Vengono infine presentati i risultati di alcuni test.

Indice

Capitolo 1

Introduzione

1.1 Che cos'è il responsive design

Il responsive design è una tecnica del web design che consiste nella creazione di siti web visualizzabili in maniera ottimale su schermi di diversi tipi di dispositivi. In particolare il layout del sito web deve adattarsi automaticamente al variare delle risoluzioni dello schermo, riducendo al minimo le azioni di ridimensionamento e scorrimento orizzontale, e deve favorire la navigazione da diversi tipi di interfaccia: mouse, touchpad, touchscreen.

Il termine è stato coniato, nel Maggio 2010, da Ethan Marcotte in un articolo apparso sul sito *A List Apart* e il W3C ha dedicato un progetto al mobile web intitolato *Mobile Web Iniziative*, pubblicando anche una serie di guidelines per chi intende migliorare la user experience di chi accede da dispositivi diversi dal desktop.

Il responsive design non introduce nuove tecnologie ma si basa sull'utilizzo di vecchi e nuovi strumenti, tutti già esistenti, integrandoli in maniera efficace: misure relative, device detection, CSS3, Javascript.

Come si vedrà analizzando le tecniche più comuni per realizzare un sito responsive, la definizione di *tecnica del Web Design* sembrerà riduttiva e si

potrà iniziare a parlare piuttosto di *strategia* poiché coinvolge vari aspetti del lavoro di costruzione di un sito web. Oltre al layout fluido si deve prestare attenzione ad altri importanti fattori, quali: gestione dei contenuti, ovvero a quali informazioni potranno accedere gli utenti e con quali priorità; accessibilità, attenersi agli standard e mantenere un codice pulito per avere maggiore compatibilità con vari browser; velocità delle connessioni e potenza di calcolo dei dispositivi portatili, riducendo i tempi di accesso alle pagine.

L'eterogeneità dei dispositivi e dei browser utilizzati e le reti spesso troppo poco potenti portano ad una notevole difficoltà nel trovare un'unica soluzione ottimale, valida per tutti i siti, e applicabile come standard. Nello stesso tempo, però, le statistiche dicono che, nel giro di pochi anni, gli accessi al web da dispositivi mobili supereranno quelli da PC e pertanto non si può più prescindere dallo sviluppo di siti web flessibili. È dunque il momento adatto per adottare questa strategia poiché gli utenti si adattano velocemente ai servizi offerti dalle nuove tecnologie ed è importante adeguarsi alle tendenze, per non essere tagliati fuori dal cambiamento rimanendo invece *"a prova di futuro"*.

Figura 1.1: Classico esempio di responsive design

1.2 Storia

Nel 1994 Alain Rossmann sviluppa il WAP (Wireless Application Protocol), un protocollo di connessione ad internet ideato appositamente per i telefoni cellulari. Nel 1999 il gestore Omnitel fornisce l'accesso al WAP e, nello stesso anno, viene annunciata l'uscita del Nokia 7110, primo cellulare che supporta questo protocollo. Si potevano leggere le email ma si poteva navigare solo su dei siti appositamente formattati, utilizzando il linguaggio WML (Wireless Markup Language). Le connessioni utilizzate erano il GSM e il GPRS, che aveva una velocità massima teorica di 171,2 kbit/s ma un valore realistico di 30-70 kbit/s.

Negli anni successivi, grazie soprattutto ai palmari e ai Blackberry, si comincia a sfruttare la connessione EDGE, con cui aumenta la velocità di trasmissione dei dati (150-200 kbps) e le tariffe sono proporzionate alla quantità di dati scaricati piuttosto che al tempo di connessione. Il problema però è il supporto dei browser ai fogli di stile CSS e al linguaggio Javascript, quindi si decide di escludere queste due tecnologie fornendo solo pagine di puro HTML. Per far questo si iniziano ad usare tecniche di device detection, grazie alle quali il browser riconosce il tipo di dispositivo utilizzato e fornisce o esclude alcuni file di conseguenza.

Siamo nella prima metà degli anni 2000 e nell'ambito dei PC desktop e notebook si assiste ad una standardizzazione delle risoluzioni. Abbandonata l'obsoleta risoluzione 800x600px si affermano le risoluzioni di 1024x768px e 1280x800px e i siti web si standardizzano su layout a 960px di larghezza fissa, ma vengono talvolta utilizzate tecniche di adattamento fluido, che risultavano però di difficile gestione.

Nel 2007 avviene una piccola rivoluzione, Apple annuncia l'uscita del primo iPhone, con supporto alle sole connessioni 2G (GPRS e EDGE) e, visto il boom di vendite, nel luglio 2008 viene sviluppato il secondo modello, l'iPhone 3G,

con supporto alle connessioni UMTS (384 kbit/s). La prima risposta da parte degli sviluppatori è la creazione di siti appositamente progettati per mobile che vede però i suoi limiti nella proliferazione nel mercato di netbook e smartphone, seguiti da tablet, Web-TV e dispositivi con Retina Display.

Nasce dunque l'esigenza di sviluppare un unico codice HTML, CSS e Javascript (o almeno una buona parte) che offra all'utente la possibilità di usufruire dei contenuti di cui ha bisogno, offrendo un'esperienza coerente tra i vari dispositivi, senza ancorarsi ad uno solo di questi. [4]

1.3 Implementazione

Ethan Marcotte, l'anno successivo al suo articolo su *A list Apart*, pubblica il libro intitolato *Responsive Web Design*, ponendo di fatto le basi per la costruzione di siti web responsive. Afferma che per una corretta implementazione sono necessari tre elementi:

- Media query
- Layout flessibile e *grid-based*
- Immagini e media flessibili

Il linguaggio di formattazione CSS3 gioca un ruolo fondamentale sebbene non sia ancora un vero e proprio standard. Il W3C però pubblica costantemente informazioni sul suo sviluppo e tutti i più importanti siti web del mondo ne fanno largo uso. La proprietà più interessante per lo sviluppo di questo argomento è sicuramente *media type*, già presente nelle specifiche CSS2, ma estesa e migliorata. In particolare, mentre prima era possibile selezionare un foglio di stile in base al tipo di media utilizzato dal client, ad esempio:

```
<link type="text/css" media="screen" href="style_1.css" />
<link type="text/css" media="print" href="style_2.css" />
```

adesso è possibile indagare anche le proprietà del dispositivo stesso e utilizzare operatori logici per creare query più precise e complesse.

La lista di proprietà dei dispositivi indagabili è: *width, height, device-width, device-height, orientation, aspect-ratio, device-aspect-ratio, color, color-index, monochrome, resolution, scan, grid*, mentre i media type più importanti sono: *screen, print, braille* e *tv*. [5] Ecco un codice di esempio:

```
<link rel="stylesheet" type="text/css" media="screen and
(min-width: 480px) and (max-width: 990px)" href="tablet.css" />
```

La stessa proprietà può essere usata nei fogli di stile interni:

```
@media screen and (min-width: 480px) and (max-width: 990px){
     body, div, p { font-size: 0.8em }
     // altro codice CSS
}
```

In questi due casi la media query ha valore *true* se il dispositivo utilizzato è uno schermo di un computer o un dispositivo mobile (*screen*) e se questo schermo ha una larghezza compresa tra 480px e 990px. Se il dispositivo rispetta queste regole nel primo codice verrà caricato un apposito file *tablet.css*, nel secondo invece verrà diminuita la dimensione del font del testo. I due codici rappresentano la stessa regola ed è possibile alternarli a piacimento o usarli contemporaneamente, anche se ci sono casi, come si vedrà nel prossimo paragrafo, in cui è preferibile il secondo caso.

Grazie alle media query, unite alla proprietà *float* e all'uso di misure relative è possibile creare una griglia fluida. Supponendo ad esempio di voler visualizzare una dozzina di box contenenti immagini o testo, il layout si dovrà adattare da questa visualizzazione:

1	2	3
4	5	6
7	8	9
10	11	12

A questa:

1	2
4	4
5	6
7	8
9	10
11	12

Se le media query e le griglie sono state programmate correttamente allora il layout sarà fluido. A questo punto per rendere le immagini e i media flessibili basterà applicare due semplici proprietà ed essi si adatteranno sempre ai loro *div* contenitori:

```
img,
embed,
object,
video {
     max-width: 100% ;
     height: auto ;
}
```

La proprietà *max-width: 100%* consente infatti di adattare la larghezza dell'immagine in modo relativo, stringendo l'immagine nel caso diminuisca la dimensione del contenitore, per esempio durante la rotazione di un tablet da orizzontale a verticale. Questa proprietà inoltre è più utile della proprietà *width* perché evita di deformare l'immagine, allargandola eccessivamente, nel caso in cui la dimensione del box sia superiore alla grandezza effettiva (in pixel) dell'immagine. La proprietà *height: 100%* serve a calcolare l'altezza in proporzione alla larghezza, sempre con l'intento di non deformare l'immagine. In realtà però è anche possibile omettere questa dichiarazione in quanto i browser, in mancanza di un'altezza specifica, applicano *auto* di default. È importante notare come ci si preoccupi sempre della larghezza degli elementi

e mai dell'altezza, che viene invece modificata in proporzione. Questo perché, come detto nel primo paragrafo, uno degli obiettivi del responsive design è di evitare le azioni di scorrimento orizzontale, molto scomode per l'utente.

In teoria, dunque, queste tre tecniche fin qui esposte sono sufficienti per creare un layout responsive ma non bisogna dimenticare che la realizzazione del layout è solo una parte del lavoro di front-end development e, se prima si è parlato di strategia, allora è importante illustrare quali sono gli altri elementi fondamentali che consentono di attuarla.

Una delle decisioni più importanti da prendere è la scelta dei *breakpoint*, ovvero i punti in cui, grazie alle media query, si verificano delle modifiche al layout. Anche in questo caso non si introduce nulla di nuovo ma, per capire l'importanza di questo concetto, è utile ricorrere alla differenziazione terminologica proposta da Jeremy Keith, il quale differenzia il *breakpoint* dal *tweakpoint*. Il primo è una modifica drastica della struttura della pagina, come un elemento nascosto o un adattamento al menù di navigazione, il secondo invece è l'aggiustamento di un dettaglio, come la dimensione e l'allineamento del testo. Formalmente non c'è alcuna differenza fra i due, in quanto si realizzano entrambi con una media query, ma nella fase di progettazione fissare i breakpoint è estremamente importante per poter delineare quale sarà l'interfaccia del sito web. La decisione fondamentale da prendere riguarda il numero di breakpoint da fissare ma, poiché non vi sono delle regole fisse, può essere utile partire da una suddivisione standard:

- 320px: smartphone con orientamento *potrait*
- 480px: smartphone con orientamento *landscape*
- 768px: tablet con orientamento *potrait*
- 1024px: tablet con orientamento *landscape*

Questi breakpoint sono pensati in base alle risoluzioni più comuni dei moderni dispositivi ma non si deve dimenticare che il layout deve sempre essere progettato tenendo conto dei contenuti da presentare e che alcuni siti potrebbero avere un accesso più frequente da parte di utenti che privilegiano un tipo di dispositivo piuttosto che un altro.

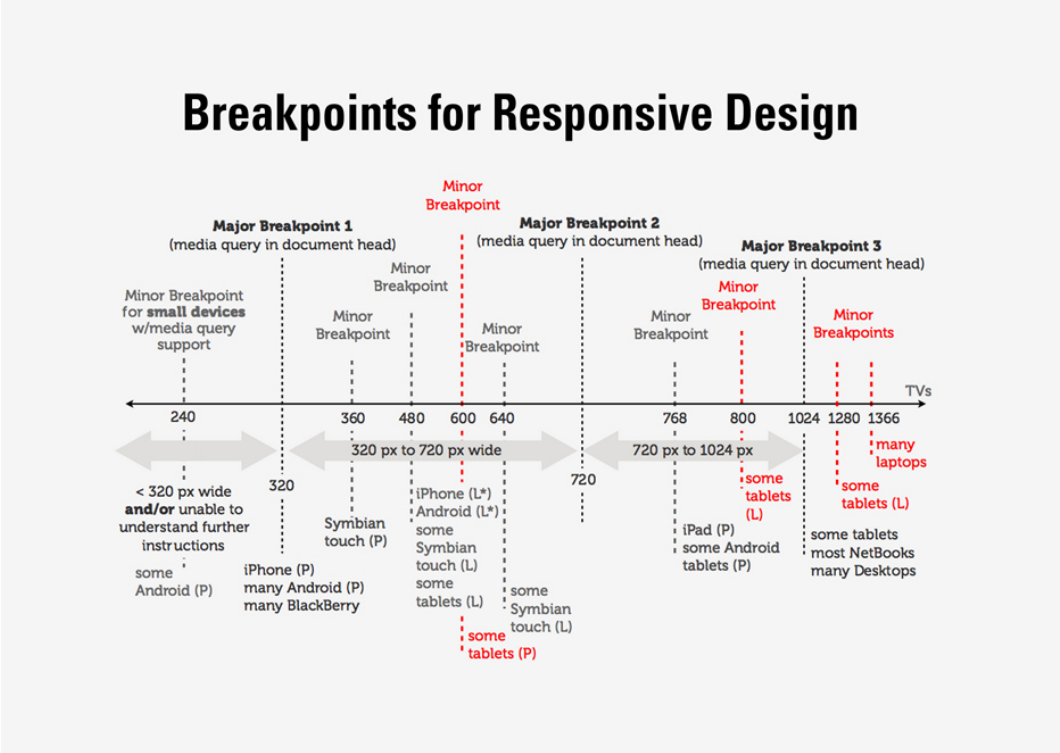

Figura 1.2: Infografica sui breakpoint più comuni associati ai dispositivi [6]

1.4 Usabilità

Un altro argomento strettamente correlato con l'implementazione di un sito responsive è l'usabilità. L'argomento è molto vasto e viene trattato spesso insieme all'*accessibilità*, altro argomento chiave del web design che però in questa tesi non viene trattato. L'usabilità è definita dall'ISO (*International Organisa-*

tion for Standardization), come *l'efficacia, l'efficienza e la soddisfazione con le quali determinati utenti raggiungono determinati obiettivi in determinati contesti.* È facile quindi capire che se lo scopo finale della strategia responsive è quello di offrire un'ottima *user-experience* allora sarà importante affrontare alcuni argomenti legati al tema dell'usabilità. Un layout scalabile infatti, da solo, è poco utile se l'esperienza complessiva dell'utente non è soddisfacente. Una strategia vincente è quella che Steve Krug, già nel 2000, aveva formalizzato con la frase *Don't Make Me Think*, ovvero non portare l'utente a dover pensare a come usufruire dei contenuti offerti ma offrirglieli con semplicità e immediatezza. Per riuscirci è necessario adottare alcune tecniche.

Innanzitutto non dimenticare che la differenza tra desktop e mobile non si esaurisce solo nella dimensione dello schermo ma si estende soprattutto alla modalità di interazione dell'utente, il quale utilizza spesso schermi *touch*. In questi casi adattare l'interfaccia restringendo i contenuti è un grosso errore poiché la precisione del click del mouse non è paragonabile alla precisione del tocco delle dita e l'utente potrebbe facilmente cliccare su un link sbagliato. Apple per esempio raccomanda una grandezza delle icone minima di circa 44x44 pixel e Microsoft suggerisce di non considerare solo la grandezza in pixel ma anche quella effettiva, in millimetri.[7][8] Un altro aspetto importante da tenere in considerazione è l'assenza di alcune funzioni come *hover*, che permette di associare un evento al passaggio del mouse su un elemento. È dunque impossibile associare funzionalità rilevanti ad effetti che non sono supportati su tutti i dispositivi poiché si infrangerebbero le regole essenziali di usabilità. Un esempio può essere il classico menù a tendina che si attiva al passaggio del mouse, il quale non sarebbe supportato da tablet e smartphone e dunque è necessaria un'importante scelta di progettazione. In generale, in tutti i siti, il menù di navigazione ricopre un ruolo cruciale e, nell'implementazione di una strategia responsive, il suo adattamento necessita inevitabilmente di un trattamento ad hoc e non può essere assolutamente escluso. Esistono

due ottime soluzioni, entrambe basate sulla possibilità di nascondere il menù quando non serve, risparmiando notevole spazio nell'interfaccia, senza cadere nell'errore precedentemente esposto di restringere eccessivamente i contenuti: *Dropdown Menu* e *Select Menu*. Il primo si costruisce applicando degli effetti di *sliding*, spesso in Javascript, facendo comparire il menù a favore dei contenuti e nascondendo inizialmente i sottomenù che saranno accessibili tramite click. Il secondo invece trasforma il menù in una *select box* che è facilmente gestita dai browser tramite una semplice finestra che appare a tutto schermo. Sempre tenendo a mente che l'utente acquisisce esperienza navigando, è importante sapere che alcune tecniche sono diventate estremamente familiari e il loro utilizzo offre la possibilità di interagire tramite meccanismi assimilati e quindi funzionanti. L'utente, inconsciamente, si aspetta di trovare meccanismi di *Hint and Reveal* (suggerisci e rivela) e riesce ad interagire con l'interfaccia se in essa sono presenti icone standard. *Hint and Reveal* è un'applicazione della tecnica di *Progressive disclosure*, utilizzata nella *human computer interaction*, che permette di mantenere l'attenzione dell'utente su aspetti più generali dell'interfaccia dando però la possibilità di approfondire scoprendo le informazioni più specifiche. Ha assunto un ruolo fondamentale nel responsive web design perché permette di *andare oltre* la dimensione del dispositivo utilizzando dei pulsanti che fanno apparire alcuni contenuti, offuscandone altri. Queste tecniche sono utili anche perché invitano l'utente a interagire e producono un senso di controllo e confidenza verso il sito web.

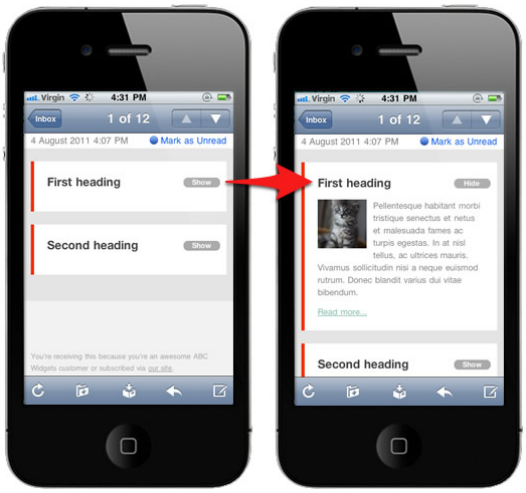

Figura 1.3: I pulsanti *Show/Hide* sono un'applicazione di *Hint and Reveal*

Un ulteriore aspetto da tenere in considerazione sono le prestazioni delle connessioni e dei dispositivi, in quanto la velocità di caricamento torna ad essere fondamentale, laddove nei PC sembrava un problema ormai superato grazie all'incremento considerevole della potenza delle connessioni rispetto agli anni passati. Se è vero che un dispositivo mobile collegato ad una Wi-fi o ad una rete 3G caricherà velocemente qualsiasi sito web, è anche vero che è frequente il caso in cui un utente possa doversi collegare da una zona poco servita persino da una rete 2G. Da tenere in considerazione anche che le attuali tariffe telefoniche sono calcolate in base al traffico dati e quindi servire pagine leggere sarà senz'altro a favore degli utenti. Esistono varie tecniche per migliorare le prestazioni di un sito web e, almeno per quanto riguarda la programmazione front-end, possono essere riassunte nel seguente elenco:

- Caricare il codice Javascript sempre alla fine del documento in modo da far caricare prima il codice HTML e CSS, che contengono informazioni e contenuti, e dopo gli script, che si solito servono a manipolare l'interfaccia o interagire con l'utente.

- Caricare le informazioni secondarie in maniera asincrona
- Utilizzare file Javascript e CSS minimizzati e markup non ridondante
- Utilizzare sempre il testo e mai le immagini per le scritte
- Servire varie risoluzioni della stessa immagine e caricare solo la più appropriata
- Usare librerie esterne perché potrebbero risiedere nella cache del client
- Ottimizzare le chiamate HTTP, diminuendo le connessioni tra il client e i server, tenendo a mente che i browser accettano al massimo due connessioni contemporaneamente

È importante citare un altro approccio, tendenza del momento, che prevede la progettazione della versione mobile di un sito web prima di quella desktop. L'approccio è chiamato *Mobile First* e l'idea di base è che può essere più facile progettare inizialmente un'interfaccia mobile che conterrà sicuramente tutte le informazioni più importanti del sito e, solo successivamente, aggiungere le informazioni accessorie, al passaggio su desktop. Questa strategia è un esempio di *Progressive Enhancement*, tecnica che consiste nel realizzare modelli semplici e successivamente aggiungere elementi accessori per creare modelli più complicati. Nell'adattamento di un sito desktop invece si adotta una tecnica di *Graceful Degradation*, in cui l'interfaccia possiede tutte le caratteristiche avanzate e subirà una semplificazione nella versione mobile. Se si progetta un sito da zero è preferibile adottare un *Progressive Enhancement*, per avere subito un test sulla versione mobile e tablet.

L'ultima accortezza da considerare, nella fase di progettazione, è quella di non dimenticare mai di offrire all'utente un'esperienza il più possibile coerente tra i vari dispositivi, poiché nella maggior parte dei casi esso accederà al sito sia da mobile che da desktop.

1.5 Strategie alternative

Le strategie alternative al responsive design sono due: *mobile site* e *mobile app*. Nel mobile site si crea una seconda versione del sito assegnata ad un altro dominio, al quale si reindirizza l'utente in seguito a un device detection (di solito *m.dominio.com*). Il vantaggio è quello di non alterare l'eventuale sito desktop già presente, lo svantaggio è quello di dover mantenere e gestire due siti diversi. Kiran Prasad, direttore tecnico per i servizi mobile di Linkedin, afferma che il mobile site è attualmente la scelta migliore per network o grandi applicazioni web. [11] La mobile app è invece un software installabile sul sistema operativo del device utilizzato. Si utilizza soprattutto se bisogna interagire spesso con l'utente o i sensori del dispositivo ma richiede un grosso investimento perché il software deve essere prodotto per tutti i sistemi operativi (*Android*, *iOS*, *Windows* ma anche altri). [12]

1.6 Statistiche

Alcune statistiche possono evidenziare l'importanza dello sviluppo cross-device. In base agli ultimi dati pubblicati da Audiweb, organismo che rileva i dati di audience di internet in Italia, il 38% del campione composto da 11-74enni dichiara di poter accedere a internet da smartphone e il 7,8% da tablet. [13] [Figura 1.4]

Figura 1.4: Statistiche Audiweb marzo 2013

1.7 Scopo della tesi

Lo scopo della tesi è quello di creare un template di sito responsive che potrà essere adottato dai dipartimenti e dai corsi di laurea dell'Università di Pisa e che sia conforme alle linee guida date dall'Ateneo. Si è cercato di utilizzare il più possibile strumenti standard, evitando elementi superflui nel codice e nell'interfaccia, poiché i poli SID (*Sistema Informatico Dipartimentale*) non utilizzano un unico CMS (*Content Management System*) e quindi si dovranno effettuare leggeri adattamenti nella fase di implementazione effettiva nei server.

Il template viene presentato con i contenuti e i colori del Dipartimenti di Filologia, Letteratura e Linguistica. Prima di rilasciare il template è stato fatto un importante test, generando un tema per Wordpress, CMS utilizzato dal SID Polo 4 - Area Umanistica.

Per meglio comprendere la struttura del template e le scelte fatte in fase di progettazione, nel prossimo capitolo vengono presentati gli strumenti, le librerie e le tecnologie utilizzate.

Capitolo 2

Strumenti e linguaggi

2.1 HTML5 e CSS3

Sebbene non siano ancora standard definiti ma specifiche in fase di definizione, si è scelto di utilizzare le ultime versioni di HTML e CSS perché si allineano perfettamente allo scopo di innovazione del progetto e si adattano ad essere la base perfetta per il framework Bootstrap che verrà descritto nel paragrafo successivo. Nonostante il loro supporto sia carente nei browser di vecchia generazione, le proprietà principali sono accettate da tutti i moderni browser e soprattutto dai browser dei dispositivi mobili. CSS3 è indispensabile per adottare la tecnica delle media query, fondamentali per il responsive design. HTML5 invece, oltre ad avere un markup più semplice e semantico, utile per l'accessibilità, possiede tutta una serie di caratteristiche pensate appositamente per migliorare l'esperienza degli utenti che accedono da dispositivi mobili. Funziona per esempio già bene la funzione di geolocalizzazione e sono in fase di sperimentazione altre API per interagire con fotocamera o rubrica. Sarà presto possibile superare il tag *<img>* con un tag che sfrutta le media query di CSS per caricare immagini diverse in base ai dispositivi e sarà anche possibile salvare alcuni file direttamente sul browser per diminuire i tempi di

caricamento. [9]

2.2 Bootstrap

Bootstrap è un front-end framework (CSS + Javascript) sviluppato da Mark Otto e Jacob Thornton per l'interfaccia di Twitter e in seguito rilasciato con licenza open source. È composto da un file CSS che rappresenta una configurazione iniziale di un sito web in cui elementi tipografici, form, bottoni, menù di navigazione e altri componenti hanno uno stile predefinito e già inizializzato. In questo modo, utilizzando il markup HTML previsto dalla documentazione, è possibile creare un'interfaccia web con estrema semplicità ed immediatezza. Oltre al file CSS principale, in realtà, sono presenti altri file CSS e Javascript che si comportano come estensioni, aggiungendo nuovi elementi o effetti particolari. Il 19 Agosto 2013 è stata rilasciata la versione 3, che ha segnato un importante passo avanti nel settore dei framework pensati per lo sviluppo di siti e applicazioni Web responsive. È stata introdotta la possibilità di adottare la strategia Mobile First e il supporto al responsive design non è basato su un foglio aggiuntivo, come nella versione precedente, ma è integrato. È importante notare che, utilizzando Bootstrap, non si è vincolati ad alcuna tecnologia in particolare, perché il file viene letto dai browser o dai CMS come un normale file CSS creato dallo sviluppatore ed è quindi integrabile in qualsiasi progetto.

Dopo aver scaricato ed eventualmente personalizzato il file CSS principale, lo si include nella pagina HTML. È possibile sovrascrivere tutte le proprietà con ulteriori file CSS caricati in seguito e, infatti, di solito il file principale è incluso prima di tutti gli altri. Un semplice esempio di implementazione è il seguente:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Esempio Implementazione Bootstrap</title>
    <link href="bootstrap.css" rel="stylesheet">
    <link href="stile_aggiuntivo.css" rel="stylesheet">
  </head>
  <body>

    //codice HTML

    <!-- Opzionale: jQuery e Bootstrap JavaScript plugins-->
    <script src="jquery.min.js"></script>
    <script src="bootstrap.min.js"></script>
  </body>
</html>
```

Questo esempio crea una pagina bianca ma tutto il codice HTML che verrà inserito, in particolare titoli, paragrafi, bottoni, form, avranno già uno stile preimpostato.

Per quanto riguarda il contenuto delle pagine, Bootstrap si basa su un sistema *grid based*, creando una serie di righe e colonne con cui è possibile strutturare il layout. Inizialmente il sistema potrebbe apparire complesso e poco intuitivo ma permette di creare un layout solido e robusto, facile da gestire e da personalizzare. Per gestire le griglie si associano delle particolari classi ai *div* e si usano delle regole che possono essere così sintetizzate:

- Righe e colonne devono essere sempre contenute dalla classe *.container*
- Le righe rappresentano gruppi di colonne e sono associate alla classe *.row*
- La classe *.row* crea un *div* che si adatta alla larghezza massima del suo contenitore e può contenere un massimo di dodici colonne
- Alle colonne è associato un numero che rappresenta la loro larghezza rispetto al *.row* che li contiene e può variare da 0 a 12

- La somma dei numeri associati alle colonne per ogni *.row* deve sempre essere uguale a 12
- Le colonne sono create con la classe *.col-x-y* dove y rappresenta il numero precedentemente descritto e x è un valore tra *xs*, *sm*, *md*, *lg* che rappresenta il comportamento della colonna in base al dispositivo di accesso alla pagina web
- Ogni colonna può a sua volta contenere un *.row* e altre colonne, seguendo sempre le stesse regole

È importante capire a fondo il funzionamento della classe *.col-* perché rappresenta la chiave del meccanismo che consente di adattare il layout ad ogni dispositivo. Ogni colonna, infatti, dovrà subire un adattamento con il variare della larghezza dello schermo e potrebbe, per esempio, affiancarsi ad un'altra colonna in uno schermo grande ma ricoprire tutto il display in uno schermo piccolo. I valori *xs*, *sm*, *md*, *lg* rappresentano rispettivamente *Phones* ($<$ 768px), *Tablets* ($\geq$768px), *Small Desktops* ($\geq$992px), *Large Desktops* ($\geq$1200px) e funzionano grazie a delle normali media query. Da notare che essi rappresentano di fatto i *breakpoint* del template e sono fissati in base agli standard descritti nel capitolo precedente.

Un esempio basterà a chiarire il concetto:

```
<div class="row">
  <div class="col-lg-6 col-xs-12"></div>
  <div class="col-lg-6 col-xs-12"></div>
</div>
```

In questo caso ogni *div* ha due proprietà. Con la proprietà *col-lg-6*, in un *Large Desktop*, i due *div* ricopriranno metà dello spazio occupato da *.row* e si affiancheranno l'uno all'altro; con la proprietà *col-xs-12*, in un *Phone*, i *div* si allargheranno occupando il 100% dello schermo e disponendosi uno sotto l'altro. Naturalmente per ottenere un risultato più preciso è possibile utilizzare

tutte e quattro le proprietà contemporaneamente. È possibile inoltre utilizzare altre proprietà per gestire alcuni casi particolari.

La proprietà *-offset* permette di creare eventuali spazi vuoti, nel caso in cui la somma dei numeri associati alle colonne non arrivi a 12. In pratica viene ampliato il margine sinistro della colonna stessa così da compensare lo spazio libero. Si utilizza in maniera analoga alla classe *.col-* ma è priva del valore *xs* in quanto, in un'interfaccia stretta, è consigliato riempire sempre tutto lo schermo:

```
<div class="row">
  <div class="col-lg-6 col-lg-offset-3 col-md-10 col-md-offset-1">
</div>
```

In quest'ultimo esempio è presente un unico *div* che non copre tutta la larghezza del *.row* ma lascia tre spazi a sinistra nella visualizzazione *-lg* e uno spazio nella visualizzazione *-md*.

Un'altra proprietà interessante è quella che consente di invertire l'ordine delle colonne. Prima di descrivere il suo funzionamento è bene ricordare che le colonne si affiancano l'una alla destra dell'altra e quindi, in ogni riga, la prima colonna è sempre quella più a sinistra. Diminuendo le dimensioni dello schermo, le colonne più a destra scivoleranno in basso fino ad arrivare ad una situazione in cui si presenteranno tutte una sotto l'altra. Questa tecnica dunque risulta molto utile, ad esempio, nel caso in cui il contenuto principale della pagina è posto a destra di un menù verticale ma lo si vuole far apparire in alto nella visualizzazione mobile. Si realizza inserendo i valori *push* o *pull* all'interno della classe *.col-*:

```
<div class="row">
  <div class="col-sm-12 col-lg-6 col-lg-push-6"> Div1 </div>
  <div class="col-sm-12 col-lg-6 col-lg-pull-6"> Div2 </div>
</div>
```

In questo esempio, nella visualizzazione *sm* le colonne riempiranno lo scher-

mo e si posizioneranno nell'ordine in cui sono state scritte, una sotto l'altra. Nella visualizzazione *lg*, invece, il *Div1* si sposterà a destra di 6 blocchi (*col-lg-push-6*) e il *Div2* si sposterà a sinistra (*col-lg-pull-6*) e le colonne risulteranno invertite.

Un caso molto comune potrebbe anche essere quello di voler nascondere alcuni contenuti. Questo è possibile grazie alle classi *.visible* e *hidden* che si uniscono ai valori *xs*, *sm*, *md*, *lg* con l'aggiunta di un ulteriore valore *print*. Il meccanismo è sempre lo stesso:

```
<div class="row">
  <div class="hidden-xs hidden-sm visible-print"></div>
</div>
```

A questo punto si potrebbe pensare che il framework offra un meccanismo eccessivamente rigido, ma è bene precisare che il developer è libero di modificare tutte le proprietà fin qui esposte. Si può modificare il file CSS principale durante il download dal sito ufficiale alla pagina Customize usufruendo di un'interfaccia grafica. Alternativamente si può utilizzare *LESS*, un linguaggio di stylesheet dinamico, per modificare i *breakpoints* o il numero di colonne massime e la loro grandezza.

Le proprietà fin qui esposte rappresentano solo quelle fondamentali per poter comprendere il funzionamento del layout *grid-based*. Il framework offre molte altre proprietà ma verranno discusse, nel prossimo capitolo, solo quelle effettivamente utilizzate nella produzione del template proposto.

2.3 jQuery

jQuery è un framework, cross-browser, che permette di semplificare la programmazione lato client degli script Javascript delle pagine HTML. Negli ultimi anni il linguaggio Javascript è stato rivalutato, sia grazie alla possibilità di produrre effetti grafici dinamici, alternativi all'ormai tramontata tecnologia

Flash, sia grazie alla diffusione della tecnica AJAX (*Asynchronous JavaScript and XML*) che permette di modificare alcune parti dell'interfaccia, scaricando file dal server, senza dover ricaricare la pagina intera. La natura cross-browser è particolarmente importante perché neutralizza la differenza di interpretazione del codice Javascript da parte dei diversi browser, i quali posseggono differenti interpreti/compilatori. Lo scopo principale di questo framework è di rendere il codice più sintetico e di limitare l'estensione di oggetti globali per ottenere massima compatibilità con librerie esterne. È stato scelto per questo progetto proprio perché, nell'ambito della progettazione responsive, è fondamentale la piena compatibilità con tutti i browser e la brevità del codice, che permette di generare script meno pesanti e quindi più veloci da scaricare sul client. Sviluppando inoltre un template che dovesse essere implementato su differenti CMS e in cui verranno aggiunte varie funzionalità, potrebbe ricoprire un ruolo fondamentale la funzione *jQuery.noConflict()* che permetterà, qualora ci fosse bisogno, di usare contemporaneamente altri framework Javascript. È possibile includere anche una serie di librerie aggiuntive che fanno parte del progetto *jQuery UI* che forniscono interazioni, animazioni e oggetti avanzati. Una delle librerie più importanti è *jQuery Mobile* che ottimizza le interazioni per dispositivi touch e ha la possibilità di interagire direttamente con i più importanti sistemi operativi mobile.

La funzione principale è *$()*, grazie alla quale è possibile richiamare le funzioni che sono associate ai selettori. Le funzioni inoltre sono concatenabili per poter ottenere un codice più leggero e leggibile.

L'implementazione si ottiene tramite file locale:

```
<script language="javascript" type="text/javascript" src="jquery
    -1.10.2.min.js"></script>
```

Oppure tramite file remoto:

```
<script language="javascript" type="text/javascript" src="//code.
    jquery.com/jquery-1.10.2.min.js"></script>
```

Una terza soluzione, non ufficiale ma molto utilizzata, è quella di includere una copia del framework presente sui server di Google:

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.
    min.js"></script>
```

Quest'ultima soluzione è particolarmente indicata soprattutto se si pensa al suggerimento, dato nel capitolo precedente, riguardo le performance delle reti e le prestazioni dei dispositivi mobili. Le ragioni sono almeno tre: [**?**]

- Caching: è molto probabile che l'utente, navigando sui server di Google, abbia già scaricato il file e ne possegga una copia in cache. Da considerare anche le eventuali copie scaricate dall'utente navigando su altri siti importanti come Twitter, Foursquare, Slideshare e StackOverflow
- Parallelismo: i browser limitano gli accessi paralleli allo stesso hostname quindi è preferibile eliminare una delle richieste al proprio sito
- Diminuzione latenza: all'interno della rete esistono dei nodi *CDN* (Content Delivery Network) che si preoccupano di instradare le richieste ai server fisicamente più vicini. È molto probabile che una delle copie del file jQuery di Google sia più vicina all'utente rispetto al file nel server del proprio sito

2.4 Wordpress

Wordpress è un *content management system* (CMS) che consente la creazione di siti facilmente gestibili e aggiornabili. È stato creato da Matt Mullenweg nel 2003 ed è stato rilasciato sotto licenza GNU General Public License. Possiede una comunità molto attiva e numerosa e dà la possibilità di usufruire di template preimpostati, chiamati "temi", e di crearne di propri. Offre anche la possibilità di implementare o sviluppare dei plugin, ovvero funzionalità

aggiuntive che permettono un miglioramento notevole nella programmazione del sito. Crea URL permanenti alle pagine, migliorando l'ottimizzazione per i motori di ricerca e distingue le normali pagine web dai *post*, i quali supportano nativamente un sistema di suddivisione in categorie, tramite tag. Gestisce in maniera estremamente semplice i permessi di accesso dei vari utenti a un sito e possiede un editor *WYSIWYG* (What You See Is What You Get) per la formattazione dei testi. Queste caratteristiche lo rendono particolarmente adatto alla gestione di un sito di un corso di laurea o di un dipartimento ed è infatti utilizzato attualmente dal SID Polo 4.

2.5 Altre librerie

Sono state utilizzate inoltre altre librerie:

- Sidr.js: un plugin jQuery per la creazione di menu responsive
- Touchwipe.js: un plugin jQuery che implementa alcune *gesture* utilizzate dai dispositivi mobili
- Awesomefont.css: una libreria di icone vettoriali scalabili pensate appositamente per estendere il pacchetto di icone di Bootstrap. Le *font icon* sono più veloci da caricare rispetto alle immagini ed essendo scalabili sono compatibili anche con i Retina Displays. Non necessitano di Javascript e sono compatibili con gli screen reader
- Respond.j: un plugin Javascript che abilita l'uso delle media query anche sui vecchi browser
- Normalize.css: un piccolo file CSS che allinea lo stile di default di alcuni elementi HTML nei diversi browser. È un'alternativa moderna al tradizionale CSS reset

Capitolo 3

Modello

3.1 Punto di partenza

Il punto di partenza del lavoro è stato il template rilasciato dall'ufficio comunicazione nel dicembre 2012, in cui sono esposte le nuove linee guida per lo sviluppo del layout dei futuri siti dei dipartimenti. La struttura dell'intestazione contiene i nomi dei corsi di laurea o dei dipartimenti e sono previste due possibilità:

- Denominazione struttura o Dipartimento / Università di Pisa
- Corso di laurea / Dipartimento / Università di Pisa

Il menu di navigazione è pensato per rimanere uguale per tutte le pagine e la funzionalità di ricerca per essere locale al sito, il footer invece riporta informazioni utili per i contatti come numeri di telefono, fax, posta elettronica del Dipartimento o corso di laurea. Ad ogni area è stato inoltre assegnato un colore caratterizzante, utilizzato per alcuni dettagli grafici.

Il layout del corpo della pagina è diviso in quattro zone: zona in alto con contenuti multimediali per rafforzare l'identità del sito, colonna sinistra con

menù di navigazione, corpo della pagina, colonna a destra per contenuti extra. Questa struttura è stata dichiarata flessibile e sintetizzata in questo modo:

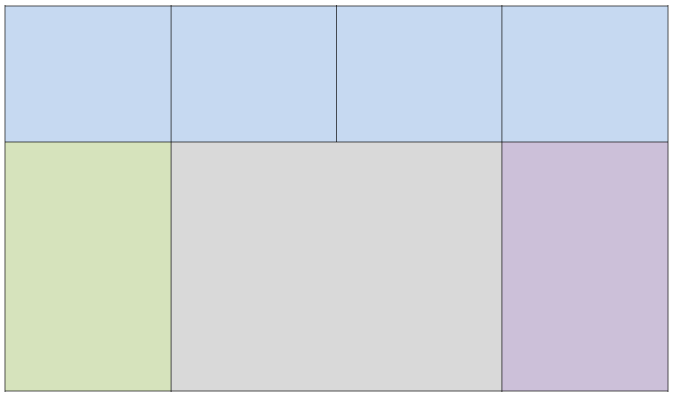

Figura 3.1: Griglia di base

Questa griglia è particolarmente utile per le homepage perché contengono molte informazioni ma può essere snellita soprattutto nelle pagine secondarie di contenuto.

Il template iniziale di esempio aveva questa interfaccia:

Figura 3.2: Template iniziale

All'inizio si è provato a lasciare inalterate le classi CSS più importanti e ad adattare la struttura HTML integrandola a Bootstrap. Vista però la difficoltà di questo adattamento, dovuta alla grossa differenza tra l'articolazione richiesta dal framework e quella iniziale, è stato preferito infine ristrutturare da zero un nuovo template.

3.2 Layout grid-based

Il primo passo nella costruzione del nuovo modello è stato quello di ricostruire lo stesso layout del template di partenza seguendo però le linee guida sulla costruzione di layout *grid-based* descritta nel precedente capitolo. Il risultato, visibile in Figura 3.3, è soddisfacente e prestando attenzione alle classi utilizzate si possono già notare alcune scelte di adattamento multi-device effettuate, come ad esempio il *div#subcontent* che si allarga quando la colonna a destra sparisce. Una modifica sostanziale è stata quella di passare da una larghezza massima di 916px a 1200px, per sfruttare l'ampiezza degli attuali monitor desktop e allineando il template allo standard adottato da Bootstrap. L'altra modifica rilevante è stata quella di dare maggiore importanza alla colonna di sinistra spostandola in alto, subito sotto il *nav*. La conseguenza di questa scelta è stata quella di restringere lo spazio riservato ai contenuti multimediali.

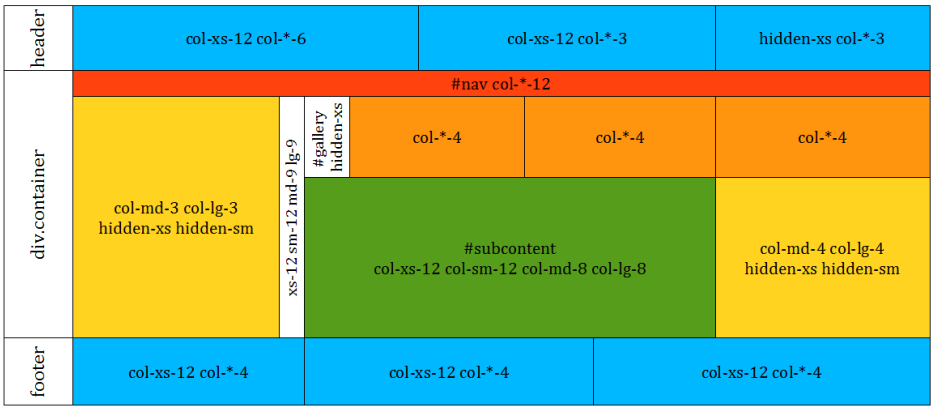

Figura 3.3: Layout grid-based

All'interno di questa struttura ogni componente ha subito diverse modifiche ed è utile elencare, punto per punto, quali sono gli elementi visualizzati nell'interfaccia e come essi sono stati costruiti.

3.3 Descrizione template

L'header ha una griglia molto semplice, identica a quella del modello di partenza, ma nasconde una complicazione dovuta alla molteplicità di variazioni che subisce al variare del dispositivo. C'è inoltre da tener presente che la lunghezza dei nomi dei corsi di laurea o dei dipartimenti può variare molto ed è dunque difficile adottare un'unica soluzione ottimale per un elemento che di solito nei siti web rimane invariato. È stato necessario utilizzare quattro *tweakpoint* in cui dalla versione estesa:

Corso di laurea / Dipartimento / Logo

si riduce la dimensione del testo e dei margini nei dispositivi con risoluzione inferiore a 1200px e spariscono le barre inclinate. Negli smartphone viene nascosto anche il logo e i due titoli ricoprono tutta la larghezza del dispositivo disponendosi uno sotto l'altro. È stato utilizzato inoltre il font Swift (Gerard

Unger, 1989) con proprietà *small-caps* (maiuscoletto), lo stesso utilizzato per il logo ufficiale dell'Università di Pisa.

Il menu di navigazione principale è l'elemento che subisce maggiori modifiche. All'interno del div *#nav* ci sono due differenti menù: *#menu_ desktop* a cui sono assegnati le classi *visible-md visible-lg* e *#menu_ mobile* a cui sono assegnati *hidden-md hidden-lg*. Il primo è composto dalla lista di elementi standard *ul > li*, in cui le uniche particolarità sono la classe *.current* che serve a sottolineare la voce di menù selezionata con il colore del dipartimento e la classe *dropdown* utilizzata da Bootstrap per creare i sottomenu. Sulla destra è presente la barra di ricerca e un'icona per visualizzare la versione inglese del sito.

Figura 3.4: Header e menu desktop

Il secondo menù, quello visualizzabile da smartphone e tablet verticale, è inizialmente privo di voci ed è formato solo da un piccolo logo dell'Università a sinistra e da due icone a destra. Il logo serve perché a questa risoluzione è già scomparso il logo nell'header, le icone invece servono rispettivamente per fare una ricerca nel sito e per aprire un menù laterale a comparsa. Durante la navigazione, grazie ad un effetto Javascript, il menù resta fisso in alto così da avere sempre accesso ai pulsanti senza dover scorrere ripetutamente la pagina. È stato lasciato appositamente un po' di spazio al centro di questo menù per permettere l'aggiunta di un eventuale secondo logo. Il menù a comparsa è stato implementato grazie alla libreria *Sidr* che permette di aggiungere alla pagina un menù simile a quelli utilizzati dalla App mobile, in cui i contenuti vengono spostati verso uno dei due lati dello schermo con un effetto di *sli-*

de. Questa tecnica si rivela particolarmente utile sia perché non è intrusiva e permette di usufruire di tutto lo spazio nei dispositivi con schermi piccoli, sia perché l'utente ha familiarità con il suo utilizzo. La larghezza del menù è di 260px, abbastanza largo per ospitare le voci di menù senza però mai coprire l'intero schermo. È possibile chiudere il menù sia premendo lo stesso pulsante di apertura sia con una *Swipe gesture*, letteralmente *gesto di trascinamento*, da sinistra verso destra. Questo meccanismo, supportato da tutti i dispositivi touch, è gestito tramite un altro script esterno chiamato *touchwipe*, estensione di jQuery. Anche in questo caso è stato fatto uso dell'effetto *dropdown*, indispensabile per riuscire a visualizzare tanti contenuti in un piccolo dispositivo, unito anche all'effetto jQuery *toggle* con lo scopo di tenere aperto solo un sottomenù per volta e chiudendo automaticamente gli altri.

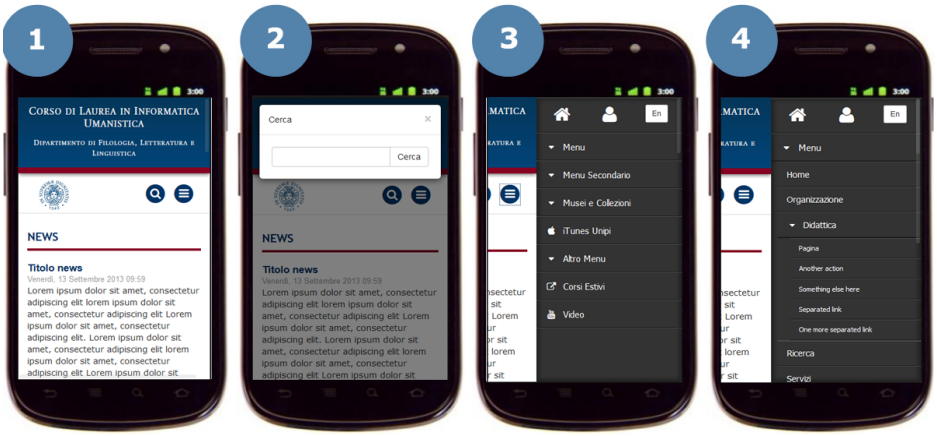

Figura 3.5: Header e menu laterale per smartphone

Grazie a questo menù è possibile alleggerire i contenuti della pagina eliminando le colonne laterali e trasformando le informazioni in esse contenute in voci di menù. Le icone sono utili per compiere azioni frequenti come tornare alla homepage o effettuare un login, l'altro pulsante serve a sostituire l'icona della bandierina che in questo menù risultava graficamente poco coerente.

La visualizzazione da tablet è un misto delle due visualizzazioni viste fin'ora e in particolare la versione *landscape* risulta identica a quella di un monitor desktop ma con i contenuti ristretti e leggermente più allungati. Nella versione *portrait* i menù laterali scompaiono ma l'header rimane uguale alla versione desktop/landscape.

Un discorso a parte meritano i contenuti centrali della pagina, ovvero i contenuti multimediali e lo spazio per le News o gli articoli. Questi ultimi saranno sempre visualizzati in qualsiasi dispositivo perché rappresentano il nucleo delle informazioni che le pagine offrono all'utente e occuperanno l'intera larghezza del dispositivo quando scompaiono le colonne. I contenuti multimediali invece possono variare offrendo la libertà di scegliere se visualizzare tre o quattro immagini, oppure uno slideshow. Anche la decisione sul tipo di adattamento è libera, si può scegliere se nascondere i contenuti o visualizzarne solo una parte. Nel modello base è stato scelto di visualizzare tre immagini e di nasconderle nella visualizzazione da smartphone.

I menù laterali hanno subito un'evidente restyle grafico in quanto è stato evidenziata un'eccessiva vicinanza tra i link del menù di partenza ed è stato progettato un menù in cui è possibile cliccare su tutta la riga corrispondente alla voce. Questo faciliterà sicuramente la navigazione da tablet, evitando errori nei click o *zoom gesture*. È stato inserito inoltre un intestazione con bordi arrotondati e una griglia che contiene le voci.

Figura 3.6: Vecchio menu (a sinistra) e nuovo menu (a destra)

Il footer infine è stato notevolmente cambiato inserendo, oltre ai contatti, una colonna con i link utili e i link ai social network e una colonna contenente la mappa di Google, utile per indicare all'utente la posizione delle strutture dei dipartimento. Una striscia finale contiene invece Sitemap e altre eventuali informazioni.

Figura 3.7: Footer

Per dare un margine di flessibilità sono previste anche delle soluzioni alternative per alcune parti del template. Oltre alle varie soluzioni possibili tra l'adozione di tre, quattro immagini o uno slideshow, è possibile ad esempio rinunciare al secondo titolo dell'header oppure adottare i vecchi menù laterali. Il primo caso è stato preso in considerazione poiché potrebbero verificarsi casi di

titoli eccessivamente lunghi e difficili da gestire. Il caso dei menù invece è utile se, in fase di implementazione, si vuole adottare una migrazione dal vecchio al nuovo template più graduale e meno incisiva verso l'utente. Il pacchetto finale è dunque formato dai seguenti file:

- index.html
- main.css
- colore.css
- Swift-04-Regular.otf
- altri file html per le soluzioni alternative
- immagini
- readme.pdf con le linee guida

In generale la filosofia alla base della progettazione del template è stata quella di semplificare il più possibile il codice così da renderlo facilmente accessibile ai tecnici dei poli SID per eventuali adattamenti futuri e quindi le soluzioni proposte servono a dare un esempio concreto che segua le linee guida imposte dall'Ateneo. Tutte le modifiche che non infrangano le linee guida sono quindi possibili.

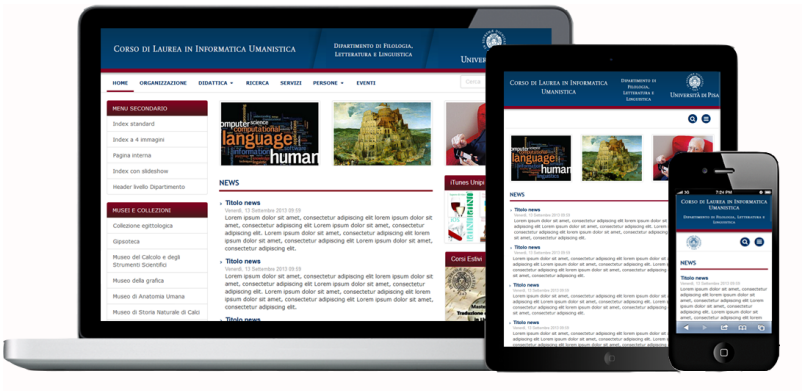

Figura 3.8: Risultato finale

3.4 Tema per Wordpress

I Poli SID utilizzano CMS diversi, ed è fondamentale capire che, se ognuno si servisse di template preimpostati, adattandoli alle linee guida, i risultati ottenuti sarebbero sempre troppo differenti tra loro. La scelta che invece viene proposta con questo lavoro è quella di far generare ai CMS lo stesso codice del template HTML e, in questo modo, i siti prodotti avranno tutti la stessa interfaccia. Un margine di differenza ci potrà sempre essere, considerando anche la diversità di contenuti presentati dai diversi corsi di laurea, ma si manterrà sicuramente una buona coerenza generale.

Partendo dunque dal pacchetto di file precedentemente esposti è stato prodotto un tema Wordpress. Gli obiettivi erano due: capire se il codice era interamente riproducibile da un CMS e dare ai tecnici dell'area umanistica un prodotto pronto per un'implementazione immediata. Il risultato è stato ottimo perché, seguendo il codice HTML già prodotto e avendo a disposizione i file CSS associati, la produzione del tema personalizzato è stata rapida. La pagina *index.html* del pacchetto è stata trasformata in una pagina PHP, incorporando le funzioni richieste da Wordpress, con il seguente risultato:

```
<?php get_header(); ?>
<div class="container">
  <div class="row">
    <?php get_template_part( 'navbar' ); ?>
    <?php get_sidebar(left); ?>
    <div class="col-xs-12 col-sm-12 col-md-9 col-lg-9">
      <?php get_template_part( 'gallery'); ?>
      <div class="row">
        <?php get_template_part( 'subcontent'); ?>
        <?php get_sidebar('right'); ?>
      </div>
    </div>
  </div>
</div>
<?php get_footer(); ?>
```

Da notare che la suddivisione in porzioni di codice, buona pratica nella programmazione PHP e nell'utilizzo dei CMS, rispecchia anche la suddivisione della griglia fluida che è stata progettata. [Figura 3.3]

I * * con le relative classi creano l'*impalcatura*, le funzioni PHP invece includono le diverse porzioni dell'interfaccia. In questo modo si può lavorare su singole parti di codice senza preoccuparsi della griglia che è già configurata. Naturalmente si dovrà continuare a seguire le regole di Bootstrap, stando attenti a non sovrascrivere le classi già utilizzate, altrimenti si verificheranno degli errori. Sono stati aggiunti solo due plugin esterni: WP-PageNavi e Breadcrumb NavXT, entrambi molto utilizzati dagli sviluppatori per creare menu di navigazione con strutture differenti rispetto allo standard del CMS. L'intento era appunto quello di cambiare il markup di dafault proposto dal CMS e *forzarlo* a produrre quello imposto dal template responsive.

È stato inoltre predisposto il tema per la navigazione tra gli articoli e sono state create alcune pagine di prova ma non è stato effettuato nessun lavoro riguardo la migrazione dei contenuti poiché l'obiettivo del progetto era la creazione dell'infrastruttura, che è indipendente dalla popolazione del sito.

3.5 Test

Sono stati effettuati alcuni test di compatibilità e accessibilità.

Data l'impossibilità di testare materialmente il template su tutti i più comuni dispositivi, è stato utilizzato il servizio online Browserstack.com. Esso realizza delle simulazioni fornendo come risultato degli screenshot su come viene visualizzata una determinata pagina web in ogni dispositivo. I risultati sono stati ottimi ma si deve considerare che non si tratta di un test completo o attendibile al 100% perché esclude, per esempio, la possibilità di testare la reattività dei componenti che appaiono dinamicamente in Javascript, come il menù di navigazione laterale o gli effetti *dropdown*.

L'ultimo test effettuato riguardava la velocità di accesso al sito ed è stato utilizzato il servizio Gtmetrix.com che esprime dei giudizi sulla velocità di caricamento di una singola pagina. Il test è stato condotto sulla pagina *index* e il risultato è stato ottimo, tenendo soprattutto conto che non sono state adottate tutte le tecniche di ottimizzazione lato server ma solo quelle relative alla pulizia del codice e all'uso di caricamenti asincroni per i contenuti meno importanti. Il test ha evidenziato questi risultati:

- Total page size: 812KB
- Page load time: 3.77s
- Total number of requests: 31
- Page Speed Grade: A
- YSlow Grade: B

Il peso totale della pagina è molto basso, considerando anche che in media la homepage ha più contenuti del resto delle pagine. Il tempo di caricamento è inferiore a 4 secondi ed è quindi considerato un ottimo risultato, il numero

di richieste è un po' alto ma non ci sono richieste inutili e, come dimostrato da alcuni test condotti da Akamai, a volte è meglio sfruttare il parallelismo delle chiamate http dei browser piuttosto che salvare tutti i file sullo stesso server. Gli altri due valori sono invece un riassunto di molte proprietà che influiscono sui tempi di risposta tra client e server e possono essere analizzate nel file di report generato e incluso nel pacchetto. I tecnici che si occuperanno di implementare i siti nei server potranno così avere un termine di paragone quando caricheranno i contenuti e le informazioni cercando di non appesantire troppo le pagine.

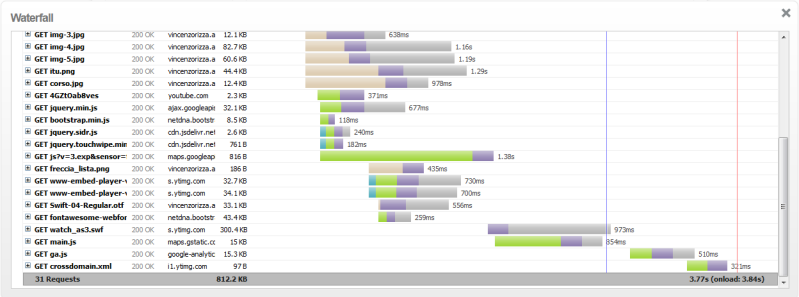

Figura 3.9: Waterfall chart con timeline caricamento file

Capitolo 4

Conclusioni

Il lavoro proposto si inserisce in contesto più ampio di ristrutturazione dei siti web che l'Ateneo sta portando avanti per offrire agli utenti un'esperienza più soddisfacente e performante, cercando di uniformare il più possibile le informazioni e i contenuti e riducendo l'eterogeneità di siti e servizi sorti negli ultimi anni. Il modello presentato ha quindi il duplice obiettivo di proporre un'interfaccia abbastanza semplice da poter essere adottata da tutti i dipartimenti senza troppi problemi di implementazione e di proporre soluzioni graficamente innovative senza però stravolgere la linea grafica adottata da tutti gli altri siti dell'Ateneo. La vera problematica è stata infatti quella di sperimentare senza stravolgere e di non cadere nell'errore di adottare soluzioni poco versatili e quindi inefficaci. Il risultato ottenuto è in linea con gli obiettivi fissati in partenza ed è stato ampiamente visionato e discusso da alcuni tecnici informatici e dall'Ufficio Comunicazione, che ha anche espresso suggerimenti utili riguardo le soluzioni adottate.

A questo punto dovrebbe essere chiaro che il responsive design non è semplicemente una tecnica CSS *cool* ma una strategia profonda, una soluzione a problemi reali, un'impostazione che può reggere alla velocità di cambiamento del Web e alle sue modalità di accesso. Proprio per questo motivo è necessario

fin da subito attrezzarsi per riuscire a rimanere al passo coi tempi e, anche se non esistono ancora degli standard pienamente accettati o dei workflow da seguire, è meglio non rimanere ancorati a delle soluzioni che già adesso presentano delle problematicità e che, nel futuro, ne presenteranno sicuramente delle altre.

Il modello può essere considerato una versione 1.0 stabile ma avrà certamente bisogno di numerosi aggiornamenti. Sarebbe utile un test di implementazione su tutti i CMS, un test di accessibilità e un test utente, senza dimenticare le ottimizzazioni lato server per migliorare ulteriormente la velocità di caricamento delle pagine. Il lavoro sarebbe dunque ancora lungo ma questo modello rappresenta sicuramente un primo passo verso una user experience nettamente migliorata.

Bibliografia

[1] Marcotte, Ethan. *Responsive Web Design*. A Book Apart; 1st edition (2011).

[2] Marcotte, Ethan. Sito di A List Apart, articolo *Fluid images*, 7 giugno 2011.

http://alistapart.com/article/fluid-images

[3] Wroblewski, Luke. *Mobile First*. A Book Apart, 18 ottobre 2011.

[4] Saporiti, Marco. *La storia della telefonia in Italia. Da Marconi e Meucci ai giorni nostri*. Cerebro Editore

[5] Sito del W3C, pagina *Media Queries*

http://www.w3.org/TR/css3-mediaqueries/

[6] Sito di Steve Roper, articolo *Setting Responsive Design Breakpoints*. 11 Aprile 2013

http://stephenroper.com/blog/?p=2357

[7] Apple Inc, *Icon and Image Sizes* in *iPhone Human Interface Guidelines*, 22 Ottobre 2013.

https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/MobileHIG.pdf

[8] Sito di Windows Phone Dev Center, pagina *Interactions and usability with Windows Phone*

http://msdn.microsoft.com/en-us/library/windowsphone/design/hh20288-9%28v=vs.105%29.aspx#BKMK_Minimumtextsize

[9] Sito del W3C, pagina *HTML5*

http://www.w3.org/html/wg/drafts/html/master/

[10] Sito di Bootstrap 3

http://getbootstrap.com/

[11] Yiannis Konstantakopoulos. Slideshare.com, presentazione *Responsive Web Design and the state of the Web* pag. 74.

http://www.slideshare.net/ykonstantakopoulos/responsive-web-design-the-state-of-the-web

[12] Mashable.com, pagina *Should You Build a Responsive Site or a Native Mobile App?*

http://mashable.com/2013/08/06/responsive-vs-native-app/

[13] Audiweb.it, comunicato stampa del 06-05-2013.

http://www.audiweb.it/cms/view.php?id=4&cms_pk=285