

**I sistemi di gestione del contenuto
(CMS) in ottica web 2.0: un progetto
reale, Capri 2.0.**

Renato Salzano

11 luglio 2008

Indice

1	Introduzione	5
2	Introduzione ai sistemi di gestione di contenuti	13
2.1	Cosa sono i sistemi di gestione di contenuti?	13
2.2	Le funzionalità chiave dei CMS	14
2.3	Un breve storia dei CMS, e in particolare dei WCMS.	17
2.4	La situazione attuale del mercato dei CMS	18
2.5	L'importanza dei CMS nell'ambito "web2.0"	20
3	Il progetto Capri 2.0	21
3.1	Perché Capri 2.0?	21
3.2	Analisi dei requisiti	22
3.3	Uso di un CMS per Capri 2.0	25
3.3.1	Perché usare un CMS?	26
3.3.2	Rapido confronto fra CMS esistenti sul mercato	28
3.3.3	Perché abbiamo usato Wordpress per Capri 2.0	33
3.4	Studio dell'Implementazione di Capri 2.0 usando Wordpress	35
3.4.1	Funzionalità richieste dal cliente e implementate da Wordpress	36
3.4.2	Funzionalità richieste da implementare estendendo Wordpress	37
3.5	Implementazione delle funzionalità particolari richieste da Capri 2.0	38
3.5.1	Come estendere le funzionalità di Wordpress?	38

Indice

3.5.2	Struttura dei plugin di Wordpress	39
3.5.3	Plugin per la funzionalità statistica sugli articoli	42
3.5.4	Plug-in per la funzionalità di spedizione di e-mail a più utenti	45
3.5.5	Localizzazione del plug-in	46
3.5.6	Plug-in per la funzionalità di segnalazione dei nuovi articoli	48
3.5.7	Funzionalità implementate modificando Wordpress (codehack)	49
3.6	Progettazione e implementazione del front-end verso l'utente di Capri 2.0	50
3.6.1	Meccanismo generale di Wordpress per modificare i front-end utenti: Themes	50
3.6.2	Design iniziale del Theme per Capri 2.0	51
3.6.3	Implementazione del Theme per Capri 2.0 in CSS e XHTML: Layout generale.	54
3.6.4	Implementazione dell'intestazione di Capri 2.0	56
3.6.5	Implementazione delle colonne laterali	58
3.6.6	Implementazione della zona con il contenuto specifico della pagina	60
3.6.7	Implementazione del piede delle pagine	62
3.6.8	Finalizzazione del front-end verso l'utente di Capri 2.0	62
3.7	Verifica dell'usabilità del front-end utente di Capri 2.0	69
3.8	Implementazione del front-end amministrativo di Capri 2.0	69
3.8.1	Funzionalità da implementare nella parte amministrativa di Capri 2.0	70
3.8.2	Implementazione del front-end amministrativo di Capri 2.0 usando il plug-in.	72
4	Spunti dal progetto Capri 2.0	74
4.1	La rivoluzione "Web 2.0" nell'ambito giornalistico	74
4.1.1	Cos'è il "Web2.0"?	74
4.1.2	Il giornalismo "dal basso": I blog e la blogosfera	76

Indice

4.1.3	I social networks - i successori dei blog?	79
4.2	Organizzazione dei contenuti con gerarchie piatte e possibili evoluzioni	81
4.2.1	Gerarchie piatte - un nuovo modo di organizzare i dati	81
4.2.2	TaggedNotes: un semplice esempio di uso delle gerarchie piatte in ambito personale	84
4.2.3	Folksonomies: organizzazione dei contenuti partendo “dal basso”	85
5	Conclusioni	90
	Bibliografia	97

1 Introduzione

In questo lavoro vedremo cosa sono i sistemi di gestione del contenuto (o CMS, dal loro acronimo inglese Content Management System), in particolare quelli orientati all'uso sul web, i cosiddetti WCMS, e un esempio di loro uso in un progetto reale, Capri 2.0, un servizio/comunità nascente dal web e che è incentrato sulle problematiche dell'isola di Capri.

I sistemi di gestione dei contenuti o CMS, sono una classe di applicazioni web che stanno affermandosi negli ultimi anni, il cui compito principale è quello di permettere a un utente non esperto, in maniera semplice, di inserire, gestire e recuperare del contenuto, solitamente un blocco multimediale rappresentabile su una pagina web. L'altra capacità che i CMS solitamente offrono è il controllo della pubblicazione del contenuto, ovvero dell'accesso pubblico al contenuto che gestiscono, creando una interfaccia pubblica che è assimilabile a un sito dinamico costruito usando il contenuto gestito.

Vedremo anche le relazioni fra i CMS e il cosiddetto "Web 2.0", analizzando innanzitutto cosa sia il "Web 2.0" e le innovazioni che esso ha portato, e poi come i CMS entrano in gioco come componente chiave di quasi tutte le più importanti innovazioni da esso portate. In particolare ci soffermeremo su alcune novità che emergono dai CMS e dal "Web 2.0" nel campo della strutturazione e classificazione dei dati, in particolare sulle gerarchie piatte (Flat Hierarchies) e le folksonomie.

Nel secondo capitolo vedremo nel dettaglio cosa sono i CMS, partendo dalla problematica

1 Introduzione

alla quale cercano di rispondere. Quindi vedremo la problematica della gestione dei contenuti, problematica che nasce in ambiti che non sono quelli attuali del web, e che fa da “ponte” fra le problematiche da un lato della memorizzazione dei dati (database), e dall'altra di quelle del recupero delle informazioni (information retrieval). Vedremo come questa problematica impatta domini di svariato tipo, da quello aziendale a quello personale, e come nel tempo si è provati a rispondere a tale problematica.

Vedremo poi di definire con maggiore precisione cosa sia un sistema di gestione dei contenuti e quali funzionalità deve offrire per risolvere la problematica collegata, e come nel tempo queste funzionalità siano cambiate ed evolute. Successivamente illustrerò una breve storia dei sistemi di gestione del contenuto, partendo dai suoi progenitori di metà degli anni '70, come Atex, che offrivano alcune delle funzionalità chiave, fino ad arrivare ai primi sistemi che offrano tutte le funzionalità definenti un CMS, come StoryServer nel 1996, fino ad arrivare, vedendo che funzionalità nuove i sistemi di gestione del contenuto hanno offerto, ai giorni d'oggi.

Infine, alla fine del secondo capitolo, vedremo una panoramica dei sistemi di gestione del contenuto presenti sul mercato, le funzionalità che offrono, cosa li accomuna e cosa li distingue fra loro, tentando di effettuare una classificazione di quello che attualmente si trova sul mercato. Evidenzieremo il legame fra i CMS e il cosiddetto “Web 2.0”, evidenziando come i CMS si leghino ad alcune caratteristiche chiave di tale movimento, in particolare al nuovo ruolo attivo degli utenti e alla presenza quindi del cosiddetto “contenuto utente”.

Nel terzo capitolo affronteremo nel dettaglio il progetto Capri 2.0, come esempio di uso di un sistema di gestione dei contenuti in un progetto reale, iniziando dal come nasce il progetto, come punto di aggregazione e di informazione sull'isola di Capri, e dalle richieste che mi sono state inizialmente state richieste per la parte riguardante il suo sito web, sia come funzionalità, sia come prime indicazioni sull'interfaccia con l'utente.

1 Introduzione

Successivamente vedremo perché abbiamo scelto di usare un CMS per svolgere il progetto, e quali funzionalità venivano richieste al CMS che doveva gestire il progetto. Cercheremo anche di rispondere alla domanda di quando sia conveniente usare un CMS e quando invece sia preferibile usare altre soluzioni, e perché nel caso specifico di Capri 2.0, l'uso di un sistema di gestione del contenuto fosse vantaggioso. Per poi scegliere quale prodotto CMS usare, faremo una panoramica di alcuni candidati che erano fattibili viste alcune richieste tecniche fatte, in particolare il fatto che il sito doveva essere basato su un sistema usante PHP e come base dati MySQL, e in particolare vedremo in dettaglio cinque sistemi di gestione del contenuto candidati: Docebo, Drupal, Joomla, PHP-Fusion e Wordpress. Ci concentreremo su questi cinque candidati, facendo una approfondita analisi di ognuno, evidenziando le loro caratteristiche peculiari, i punti di forza e le funzionalità che offrono, cercando di dare un quadro generale delle capacità di ognuno, ed evidenziando anche gli eventuali difetti o funzionalità mancanti. Vedremo in particolare in dettaglio come organizzano il contenuto che gestiscono, come viene amministrato lo stesso, quali funzionalità offrono allo sviluppatore web che li usa, e quanto sia facile sia adattarne l'interfaccia con l'utente che estenderne le funzionalità. Nel confronto poi non trascureremo la parte di quanto la documentazione relativa ad ognuno sia chiara e funzionale, parte che nell'affrontare per la prima volta un progetto che li usi assume una importanza non trascurabile, e quanto supporto sia presente sia in termini di comunità che in termini di estensioni presenti e reperibili in maniera gratuita.

Al termine di questo confronto, vedremo come i cinque candidati rispondevano in particolare alle esigenze del progetto, valutando quale fra essi era quello più adatto all'implementazione del sito di Capri 2.0. Analizzeremo il perché alla fine si sia scelto di usare Wordpress, e quali vantaggi esso aveva, nell'ottica del progetto, rispetto alle altre soluzioni, non trascurando comunque di segnalare quali sarebbero stati i vantaggi e le difficoltà ulteriori da affrontare se si fosse usata un'altra soluzione.

Una volta scelto Wordpress come sistema base di CMS da usare, delineremo come imple-

1 Introduzione

mentare su di esso il sito di Capri 2.0, partendo dal valutare quali funzionalità Wordpress implementava nativamente e quali interventi invece erano necessari, sia come funzionalità da aggiungere, sia per costruire l'interfaccia verso l'utente. Vedremo come in particolare solo poche funzionalità ulteriori erano richieste, in particolare l'implementazione di un sistema di spedizione di e-mail a una parte degli utenti e alcune funzionalità statistiche del sito, e come in generale doveva essere implementata l'interfaccia utente sfruttando alcune caratteristiche di Wordpress.

Illustreremo in generale i meccanismi che Wordpress offre per estendere le proprie capacità, ovvero il meccanismo dei cosiddetti "plug-in" di Wordpress, e come funzionino tali meccanismi cercando di riportare un quadro che pur passando per l'implementazione del progetto, illustri in generale tali meccanismi. In particolare vedremo che tali meccanismi sono principalmente assimilabili a delle API PHP verso Wordpress, e vedremo nel dettaglio i tre meccanismi con cui Wordpress richiama delle funzioni PHP definite nei plug-in: azioni, filtri e template tags.

Dettaglierò poi l'implementazione effettuata tramite tali meccanismi delle funzionalità statistiche sugli articoli, in particolare l'estrazione dell'elenco degli ultimi articoli più letti e commentati, vedendo nel dettaglio le eventuali nuove strutture dati che ho dovuto aggiungere, e come queste siano state implementate sfruttando il database relazionale sottostante, a cui Wordpress permette l'accesso usando apposite interfacce (WPDB), e il come il codice riesca a estrarre l'elenco degli articoli più visti di recente. Vedremo anche l'implementazione della funzionalità di spedizione delle e-mail a un gruppo di utenti, vedendo come Wordpress permetta di estendere la sua interfaccia verso l'amministratore del sito tramite l'aggiunta nuove opzioni e pagine, e i meccanismi con cui vengono implementati. Vedremo nel dettaglio il codice che aggiunge una pagina in cui l'utente inserisce i dati della e-mail da spedire, il come il plug-in faccia alcuni controlli base di sicurezza tramite un meccanismo incorporato in Wordpress, e come poi mandi la e-mail estraendo le informazioni che gli servono riguardo i destinatari usando le API di Wordpress. Infine

1 Introduzione

vedremo le meccaniche di Wordpress per la localizzazione dei plug-in, che ho sfruttato per rendere facilmente portabile in altri linguaggi queste funzionalità, basate su un meccanismo standard di Unix, GNU readline, e sul come si effettua la costruzione dei file appositi per tradurre i plug-in in altre lingue, usando strumenti standard disponibili.

Vedremo poi i meccanismi che Wordpress offre per la creazione dell'interfaccia web verso l'utente, i template o temi, e come funzionino questi meccanismi, sfruttanti delle pagine PHP che hanno la massima libertà nella struttura, e come questi meccanismi siano stati sfruttati, utilizzando le API apposite di Wordpress, dette template tags. Illustrerò le varie fasi della progettazione dell'interfaccia usata nel progetto, partendo dalle specifiche richieste iniziali, e vedendo come tramite alcuni prototipi si è passati, tramite un ciclo di iterazioni progettuali, alla progettazione dell'interfaccia utente finale.

Descriverò poi come questa interfaccia sia stata implementata usando XHTML e CSS, illustrando come l'interfaccia sia stata innanzitutto divisa in blocchi, e come poi sia stato implementato ogni blocco. Evidenzierò come siano state usate alcune tecniche CSS non banali, come le "sliding doors", che hanno permesso di mantenere l'interfaccia web fluida, ovvero che si adatti dinamicamente alla risoluzione dello schermo dell'utente, pur mantenendo un'aspetto grafico sofisticato, e "Holy Grail" che permette di avere uno schema a tre colonne con pedice e intestazione, sempre fluido, e che funzioni sotto la maggioranza dei browser, descrivendo rapidamente il funzionamento di tali tecniche. Farò alcuni accenni al perché sono state prese alcune scelte riguardo l'interfaccia web, come la scelta di non usare tabelle e di mantenere il layout fluido, che si adatti dinamicamente alla risoluzione dello schermo, in un'ottica dell'accessibilità del sito anche da parte di utenti diversamente abili. In questa parte cercherò anche in generale di evidenziare come si sia cercato di tenere conto di alcuni principi generali dell'interazione uomo-macchina, e di webdesign in generale.

Passeremo poi a come è stata progettata e implementata l'interfaccia amministrativa del sistema, e quali meccaniche di Wordpress sono state usate per implementarla. Vedremo

1 Introduzione

come si possa modificare tramite un plug-in l'interfaccia amministrativa di Wordpress, e che modifiche erano richieste dalle specifiche del progetto. Vedremo anche come queste specifiche siano cambiate parzialmente in corso di progetto, e come infine si sia implementato le varie modifiche cercando di usare principalmente sempre CSS. Mostrerò anche come siano state implementate con un'altra tecnica (codehack, ovvero modifica diretta del codice di Wordpress) alcune funzionalità, e perché sia stato necessario per esse ricorrere alla modifica diretta e non al meccanismo più standard di usare un CSS. Evidenzierò anche che svantaggi ha portato usare quest'ultima tecnica.

Successivamente, a chiusura del terzo capitolo, vedremo le verifiche di usabilità che sono in corso, mostrando con quali metodologie siano fatte.

Nel quarto capitolo, risposteremo la nostra attenzione dal progetto particolare di Capri 2.0, a una visione più generale, tornando a riprendere il tema del legame fra CMS e il movimento "Web 2.0", partendo dal progetto per vedere più in dettaglio alcuni aspetti generali delle innovazioni portate dai CMS e dal "Web 2.0". Approfondiremo il concetto di "Web 2.0", specificando meglio cosa significhi, e quale sia la sua storia, e vedremo poi i principali concetti che sono associati ad esso, dettagliandoli rapidamente.

Poi, partendo dal valore giornalistico di Capri 2.0, vedremo le innovazioni che il Web 2.0 ha portato in ambito giornalistico, e in particolare focalizzeremo la nostra attenzione sul fenomeno dei blog e della blogosfera. Vedremo cosa sono, quando sono nati e che le principali novità che hanno portato. Vedremo anche dal punto di vista tecnico lo stretto legame che hanno con i CMS, e come le loro storie siano intrecciate. Passeremo poi al concetto di "giornalismo dal basso" o "giornalismo partecipativo", e vedremo perché i blog sono diventati un fenomeno rilevante, con un valore rilevante in ambito giornalistico. Infine faremo alcune considerazioni sul loro impatto su ambiti come quello politico o sociale.

Successivamente vedremo un'altro tipo di applicazione recente che si evolve da alcuni

1 Introduzione

concetti del blog e del “Web 2.0”, i social-network. Vedremo cosa sono, e come possono essere definiti, e che novità portano. Anche in questo caso, cercheremo il loro legame con i CMS, e come abbiano posto nuove problematiche nel campo della gestione dei contenuti, e in particolare quali problematiche, e come Capri 2.0 è legata in parte a loro.

Guardando i CMS, ci focalizzeremo su una particolare struttura dei contenuti che di recente sta emergendo, le gerarchie piatte. Vedremo come anche Capri 2.0 ne usa una, e che vantaggi e novità portino. Vedremo dove altro sono usate, e in dettaglio cosa sono e come sono implementate, vedendo anche una semplice applicazione che le usa, TaggedNotes. Evidenzieremo anche i loro limiti, legati alla presenza di un “rumore di tag” peculiare delle gerarchie piatte, e che tecniche possono essere usate per superare tali limiti. Ricorderemo anche qualche esempio di servizio che le usa.

Infine, alla fine del quarto capitolo, vedremo come l’unione fra le gerarchie piatte e i CMS abbiano generato una nuova impostazione possibile nella classificazione e strutturazione dei dati, le folksonomie, unendosi ad alcuni concetti del “Web 2.0”, e in particolare lo sfruttare la “mente collettiva” degli utenti. Vedremo cosa sono in dettaglio, illustrandone il loro funzionamento e la loro struttura. Vedremo la loro storia, partendo dal primo servizio che le ha usate, del.icio.us, e come possano rispondere a delle problematiche finora irrisolte, come il loro valore per superare un problema, quello di assegnare valore semantico al contenuto, per l’implementazione reale del Web Semantico. Vedremo infine i limiti delle folksonomie, le tecniche che si stanno studiando per superarle, come i suggerimenti attivi e i raggruppamenti (bundle) di tag.

Nell’ultimo capitolo vedremo infine le evoluzioni future, partendo dall’impatto che i CMS hanno oggi in vari settori, non solo strettamente informatici, per poi vedere nei dettagli che evoluzioni possono avere in quattro aree chiave: il dominio applicativo, la strutturazione dei contenuti gestiti, il loro ruolo come “motore” o “ambiente” posto fra lo stoccaggio dei dati veri e proprio (base dati) e l’interfaccia con l’utente, e l’interfaccia vera e propria con l’utente. Vedremo quali problematiche esistono oggi in questi quattro ambiti chiave,

1 Introduzione

e che soluzioni sono attualmente in studio per superarle, dallo spostamento del loro uso dall'ambito internet a quello intranet, all'uso di nuove strutture dati come le Topic Map, fino all'uso di nuovi editor per l'inserimento del contenuto che non siano del tutto visuali.

Successivamente vedremo che evoluzione può avere il progetto Capri 2.0 nel futuro, delineando le aree future di intervento e di interesse, e le eventuali funzionalità che in futuro potrebbero diventare di interesse.

Infine descriverò la mia personale esperienza che ho avuto nel lavorare su Capri 2.0, e come questo abbia arricchito le mie conoscenze, ponendomi di fronte a problematiche che non avevo affrontato prima, e ad esperienze anche sperimentali, come il telelavoro.

2 Introduzione ai sistemi di gestione di contenuti

2.1 Cosa sono i sistemi di gestione di contenuti?

I sistemi di gestione di contenuti, conosciuti con l'acronimo CMS dall'inglese Content Management System, acronimo che useremo per indicarli da qui in avanti, sono un insieme di attrezzature, programmi e altri mezzi finalizzati a risolvere il problema della gestione di contenuti, problema che è definibile come:

“Content management is getting the right content to the right person at the right time at the right cost”¹ - Gerry McGovern

Ovvero il dare il giusto contenuto alla giusta persona nel giusto tempo e al giusto costo. Ma questo pone la questione di cosa significhi “contenuto”, e cosa significhi “gestire” il contenuto.

Il contenuto è definibile come un insieme di dati aggregati fra loro di interesse per gli utenti del CMS, dati spesso anche di tipo diverso, e che sono appunto aggregati in “contenuti”. Una pagina web, un testo o una raccolta di immagini di un certo avvenimento sono esempi di contenuti.

¹Gerry McGovern, “*CM Professionals mission statement*”, 2004

Il “gestire il contenuto” significa permettere al giusto costo a persone di vario tipo di ricevere, modificare e aggiungere i contenuti da loro richiesti in una forma a loro gradita.

Il problema da risolvere quindi ha stretti legami con i problemi dell’Information Retrieval da un lato e della memorizzazione dei dati e delle Basi Dati dall’altro. In effetti i CMS sono il sistema che si pone in mezzo fra le basi dati che memorizzano i dati usati per generare i contenuti e gli utenti con le loro richieste.

Nella realtà il termine CMS viene usato comunemente per indicare i sistemi di gestione di contenuti web, solitamente pagine multimediali contenenti tutti i tipi di dati gestibili dal web, ed essi stessi sono programmi scritti con la logica delle applicazioni web, ovvero che si integrano con il server web usando linguaggi cosiddetti di scripting per generare il codice delle pagine web servite agli utenti.

Questi CMS dovrebbero essere più correttamente chiamati WCMS (Web Content Management System, sistemi di gestione di contenuti web) proprio per indicare il tipo di contenuto che gestiscono, ma è ormai usuale usare il termine CMS per indicare questo tipo di applicazione. Faremo anche in questa tesi lo stesso assunto, indicando con il termine CMS i WCMS, ove non indicato altrimenti.

Quindi usualmente con il termine CMS si indica una applicazione web che usa i dati contenuti in una base dati esterna per generare delle pagine web che sono offerte agli utenti, e che permettono a una parte degli utenti stessi di aggiungere, modificare, ricercare o cancellare una parte dei dati stessi.

2.2 Le funzionalità chiave dei CMS

Si può definire un insieme di funzionalità chiave che definisce i CMS (cfr. [1]). Esse sono:

- Memorizzazione dei dati gestiti in un deposito (Repository) esterno, solitamente una base dati o un insieme di file.

2 Introduzione ai sistemi di gestione di contenuti

- Separazione dei contenuti memorizzati dalla presentazione degli stessi, spesso usando un sistema cosiddetto di “templating”, ovvero un sistema che usi uno “schema” di base con indicato in che punti posizionare i dati.
- Sistemi semplici e visuali (WYSIWYG) di modifica e creazione dei contenuti gestiti.
- Implementazione di un “flusso di lavoro” (workflow), ovvero di una sequenza ben definita di passi per inserire nuovo contenuto, che può prevedere una serie di approvazioni da parte di una o più classi di utenti.
- Gestione di eventuali conflitti sulla modifica di un contenuto gestito, ovvero il caso in cui contemporaneamente due utenti abilitati alla modifica cerchino di modificare uno stesso contenuto, solitamente risolto impedendo l’accesso per modifica di un contenuto se quel contenuto è in fase di modifica da parte di un’altro utente (“check-in/check-out”). Inoltre spesso solo l’utente che ha inizialmente aggiunto il contenuto (detto solitamente proprietario) e una classe di utenti (amministratori o redattori) può modificare un contenuto. Altri CMS permettono a tutti di modificare ogni contenuto gestito, e sono una classe particolare di CMS detti wiki.
- Gestione di una serie di collegamenti a pagine web esterne al CMS, spesso raccolte in categorie, e che sono presentate nelle pagine prodotte.

Altre caratteristiche che i CMS hanno nel tempo offerto sono:

- Controllo delle versioni di un contenuto, spesso memorizzando le precedenti revisioni del contenuto in un archivio storico, e con la possibilità spesso di confrontare, evidenziando le differenze, due versioni dello stesso contenuto. Questa funzionalità è quasi sempre offerta dai CMS che permettono a tutti gli utenti di modificare i contenuti, i cosiddetti wiki.
- Associazione di metadati ai contenuti, sia per effettuare ricerche sugli stessi, sia per migliorare la capacità degli utenti di arrivare ai contenuti che desiderano, sia per organizzare i contenuti stessi. I metadati sono solitamente etichette (o tag) associati

2 Introduzione ai sistemi di gestione di contenuti

ai contenuti e in alcuni casi il CMS può usare una organizzazione a gerarchia piatta dei dati, che sono distinti solo dalle etichette (o tag) associati.

- Riutilizzo dei dati, con la possibilità di associare un contenuto a un'altro contenuto presente sul web o su un'altro CMS, distribuendo ed estendendo così le fonti usate per recuperare i dati usati per generare le pagine finali. Questo può avvenire solo se ci sono formati e protocolli comuni che permettono questo scambio di dati fra diversi CMS. SCORM è un esempio di protocollo simile, orientato a dati usati per l'insegnamento a distanza (Learning Management System o LMS).
- Distribuzione finale dei contenuti su più canali (web, palmari, video, stampa). Questo implica che uno stesso contenuto è presentato usando formati dati finali diversi (ad esempio una pagina web e un documento pdf), solitamente se i dati sono memorizzati usando XML si usa una trasformazione XSLT per generare i diversi formati finali. Un caso semplice è l'uso di diversi fogli di stile su uno stesso documento web per generare versioni adatte al video, a un palmare e alla stampa.
- Adattabilità e adattività delle pagine web. Le pagine web generate dal contenuto possono essere diverse secondo l'utente che ne fruisce o secondo la classe dell'utente che ne fruisce, secondo indicazioni date dall'utente stesso o create automaticamente dal CMS.
- Localizzazione della presentazione dei contenuti, della gestione dei contenuti o dei contenuti stessi. Quest'ultimo caso il CMS deve prevedere diversi dati che però rappresentano lo stesso contenuto, e che si differenziano appunto nella lingua.
- Eventuale separazione della gestione dei contenuti vera e propria dalla loro presentazione, anche come applicazioni usate.

2.3 Un breve storia dei CMS, e in particolare dei WCMS.

La storia dei CMS è molto antica, come è antico il problema che gestiscono. Se ci concentriamo sui CMS informatici e in particolare sui WCMS, [1] prova a tracciarne la storia, che qui proverò ad ampliare.

Per tracciare una storia dei CMS bisogna innanzitutto tracciare quando le loro caratteristiche chiave sono state introdotte per la prima volta.

Il concetto di base dati nasce intorno agli anni '70, e non approfondiremo qui la storia delle basi dati, che è approfondita in numerosi altri documenti.

Il primo esempio di separazione fra contenuti e presentazione è associabile al linguaggio Atex, di metà degli anni '70.

Il concetto di modifica e inserimento visuale di dati è associabile a quello di interfaccia grafica, e quindi al laboratorio Xerox di Palo Alto, vedi [2], in particolare il progetto Parc Star, datato 1982. Il concetto di interfaccia grafica sarà poi diffuso e commercializzato al pubblico dalla Apple con l'Apple Lisa e successivamente con l'Apple Macintosh.

I concetti di business-logic e di flusso di lavoro sono commercializzati e implementati per la prima volta con il sistema Workflo della FileNet² nel 1985.

Il concetto di web e di pubblicazione web è legato alla storia del web e di internet stessa. Il web in particolare nasce nei laboratori del CERN intorno agli inizi degli anni '90, a opera di Tim Berners-Lee e altri[3].

Il primo sistema che distribuisce contenuti su più canali diversi è Cyberleaf della Interleaf³, del 1995.

²vedi la storia della FileNet come ritrovabile su <http://wiki.ittoolbox.com/index.php/Filenet>

³Interleaf è poi stata acquisita dalla BroadVision, e una versione moderna di Cyberleaf è Quicksilver, disponibile su <http://www.broadvision.com>

2 Introduzione ai sistemi di gestione di contenuti

Il termine CMS è stato usato per la prima volta dalla Vignette, per il loro sistema StoryServer⁴, che è effettivamente il primo vero Web CMS della storia, datato 1996.

Se però scaviamo più a fondo scopriamo che StoryServer era basato su un'altro sistema di produzione di contenuto dinamico, chiamato PRISM, vedi [4], e prodotto dalla CNET nel 1995, che era stato dato in licenza alla Vignette. PRISM è effettivamente il primo sistema di generazione dinamica di contenuto web partendo da un deposito dati.

Il primo uso di XML in un CMS è associato a eDB della Dynabase, nel 1996.

Il primo CMS open-source è Typo3, progetto che inizia nel 1998⁵, e che aprirà la strada alla moltitudine di CMS open-source attuali.

Il primo CMS con gestione delle versioni dei contenuti e con la possibilità da parte di tutti gli utenti di modificare i contenuti è un progetto open-source, TikiWiki⁶, progenitore di tutti i CMS di tipo wiki moderni, nel 2002.

2.4 La situazione attuale del mercato dei CMS

Con il passare del tempo, i CMS nel loro complesso si stanno suddividendo in classi di CMS che rispondono a problemi specifici legati al significato semantico dei dati che gestiscono.

Abbiamo così i CMS specializzati per gestire contenuti per l'insegnamento a distanza, detti sistemi di gestione dell'insegnamento (Learning Management System o LMS), che oltre alla gestione dei contenuti veri e propri (lezioni e fonti didattiche) gestiscono anche interazioni con l'utente di altro tipo (esami a distanza, questionari e simili).

Ci sono CMS specializzati per gestire i catalogi dei venditori via internet di prodotti (E-commerce Management System o EMS), che a una gestione più semplice dei contenuti

⁴prodotto non privo di controversie, cfr. Philip Greenspun, "Vignette", <http://philip.greenspun.com/wtr/vignette.html>

⁵cfr. Typo3, "Typo3 History", <http://typo3.com/History.1268.0.html>

⁶progetto ancora oggi in pieno sviluppo, e reperibile su <http://info.tikiwiki.org>

2 Introduzione ai sistemi di gestione di contenuti

aggiungono tutte le funzionalità richieste per il commercio elettronico, quali la gestione dei pagamenti elettronici e la possibilità per gli utenti di gestire un carrello spese.

Importanti sono quelli specializzati in insiemi di testi interconnessi e modificabili da tutti gli utenti, solitamente per enciclopedie elettroniche (wiki). Questi CMS sono quelli più concentrati sulla vera e propria gestione dei contenuti, e offrono funzionalità di gestione di varie versioni cronologiche di uno stesso contenuto, con la possibilità per ogni utente di vedere la storia passata del contenuto e di confrontarne versioni diverse. Inoltre hanno spesso funzionalità orientate all'uso come enciclopedie on-line, quali una maggiore attenzione al collegamento fra i contenuti.

Si sono poi molto diffusi negli ultimi anni i CMS orientati a creare blog personali, ovvero diari on-line in cui gli utenti possono commentare ogni testo (detto post) scritto dal gestore del blog. Questi CMS nel tempo hanno aumentato le funzionalità offerte, integrando molte funzionalità dei CMS più potenti, rendendoli una scelta molto diffusa da chi voglia un CMS più semplice da gestire.

Questa specializzazione è legata ai diversi ambiti semantici che gestiscono, che spesso pongono dei problemi peculiari che richiedono quindi dei prodotti più specializzati.

Inoltre è attualmente molto forte la presenza di sistemi open-source, in particolare legati alla piattaforma LAMP, ovvero a sistemi web basati su linux come sistema operativo sottostante, usanti Apache come server web per distribuire le pagine web, Mysql come base dati in cui memorizzare i dati e PHP o Perl o Python come linguaggio di scripting in cui scrivere le applicazioni web. La maggioranza dei CMS più diffusi è legato in effetti a questa piattaforma.

2.5 L'importanza dei CMS nell'ambito "web2.0"

Uno dei principi solitamente associati al nebuloso termine "web2.0" è il passaggio da siti in cui il contenuto era deciso completamente dal gestore del sito a siti che offrono agli utenti la possibilità di aggiungere del contenuto al sito, il cosiddetto "contenuto utente".

Quest'ultima cosa mette subito in luce l'importanza dei CMS. Infatti la presenza di contenuto aggiunto da utenti implica il dover gestire questo contenuto ponendo anche delle limitazioni a quale materiale accettare. Questo problema è esattamente risolto dai CMS come illustrato precedentemente.

E guardando alcuni dei siti più importanti del web, si nota come la maggioranza abbia un sistema di CMS (o a volte più sistemi CMS fra loro integrati) proprio per gestire il contenuto utente e/o il contenuto stesso che compone il sito.

3 Il progetto Capri 2.0

3.1 Perché Capri 2.0?

Il progetto che mi è stato assegnato, Capri 2.0, nasce dall'idea di creare una comunità virtuale legata all'isola di Capri. L'idea era quella di un sito che raccogliesse e attirasse gli abitanti e le figure di spicco di capri, dando uno spazio in cui si potesse parlare dei problemi locali di Capri.

Come da dichiarazione iniziale dei propositi del progetto:

“Capri 2.0 non è quindi una pubblicazione o un quotidiano online ma uno “Strumento di Partecipazione”. Una piattaforma (come è più giusto dire) che permetta ad ognuno di partecipare, ma soprattutto “contribuire”, alla vita sociale del paese, non attraverso grandi atti o imprese eroiche, ma semplicemente esprimendo la propria opinione, in modo chiaro e non anonimo, su un qualsiasi argomento di interesse generale. Una piattaforma che ci permetta di giungere ad una “seconda versione” della nostra isola, ponendo le basi per un futuro diverso da quello che viviamo ora. Capri 2.0 intende coinvolgere nella discussione i politici e i giornalisti locali ma anche il resto della popolazione. Raggiungere non solo il Centro ma anche quella che viene definita "The Long Tail", la lunga coda, la Periferia e, in un certo senso, avvalersi dell'intelligenza collettiva, mettendo a frutto tutte le opportunità che tale tipo di Brainstorming offre”

3 Il progetto Capri 2.0

Un sito di informazione e soprattutto dialogo strettamente legato alla realtà locale di Capri, insomma, e che se da un lato informasse, dall'altro permettesse la costituzione di una comunità di utenti che diano il loro contributo sia al sito che alla realtà locale dell'isola, sia sociale che politica.

Il progetto in particolare che ho svolto riguarda la parte del sito, che doveva essere strutturato come un giornale on-line con la possibilità di aggiungere commenti alle notizie, di gestire gli articoli dividendoli sia in categorie che in argomenti, e con la possibilità agli utenti di cercare notizie velocemente.

3.2 Analisi dei requisiti

I requisiti che mi erano stati inizialmente richiesti per il sito erano i seguenti:

- Definizione di un insieme di giornalisti, che devono essere gli unici a poter aggiungere articoli al sito.
- Gli articoli devono essere divisi in categorie, e l'utente deve poter vedere tutti gli articoli di una certa categoria.
- Il sito deve usare modrewrite di Apache in maniera che gli indirizzi degli articoli siano chiari e comprensibili, tipo "http://www.capri20.it/2008/01/04/musica/concerto_di_branduardi.p
- Possibilità agli utenti di iscriversi al sito. Gli utenti iscritti riceveranno settimanalmente una e-mail con le notizie più importanti della settimana.
- Gestione di un sistema di banner pubblicitari sulla prima pagina del sito.
- Possibilità di caricare video sul sito, associandoli agli articoli.
- Gli articoli hanno associati una serie di argomenti (tags), e deve essere possibile effettuare ricerche per argomenti o vedere gli articoli associati a un certo argomento.
- Presenza di feed RSS per le notizie e per le notizie di una certa categoria.

3 Il progetto Capri 2.0

La prima pagina in particolare doveva visualizzare:

- Un elenco degli ultimi articoli del sito.
- Un elenco delle categorie degli articoli.
- I banner o spazi pubblicitari
- Un link per raggiungere l'elenco di tutti i giornalisti del sito.
- Gli articoli prescelti di prima pagina.
- Una casella per autenticarsi con username e password.

Per ogni articolo inoltre doveva essere possibile effettuare le seguenti azioni:

- Vedere gli eventuali commenti associati all'articolo, se si è uno dei giornalisti.
- Aggiungere un commento all'articolo, se si è uno dei giornalisti.
- La possibilità di segnalare via e-mail l'articolo.
- La possibilità di stampare l'articolo.

Una successiva analisi evidenziava altre richieste:

- La possibilità in futuro di modificare il workflow degli articoli prevedendo un “caporedattore” che approvi o meno gli articoli eventualmente mandati dai giornalisti
- la presenza di una interfaccia per gestire tutti gli aspetti del sito che sia amichevole e facilmente usabile, e che permetta almeno di gestire quali utenti sono giornalisti abilitati a scrivere articoli, oltre che a permettere ai giornalisti di scrivere gli articoli.
- La presenza di un sistema per mandare una e-mail a tutti i giornalisti o a tutti gli utenti iscritti o ad entrambi, che sia facilmente usabile.
- La gestione di un insieme di banner pubblicitari.

Dai requisiti quindi si evince che esistano quattro figure di utenti che usano il sito:

- Gli utenti semplici, che devono poter fare le seguenti funzioni:

3 Il progetto Capri 2.0

- Vedere quali articoli sono in prima pagina
 - Vedere quali articoli sono i più visti della settimana
 - Vedere quali articoli sono presenti in una certa categoria
 - Vedere l'elenco di quali categorie esistano
 - Leggere un articolo preciso
 - Cercare fra gli articoli quali abbiano presenti alcune parole chiave
 - Vedere quali articoli parlano di un certo argomento
 - Vedere quali argomenti sono i più trattati nell'ultima settimana
 - Vedere l'elenco dei giornalisti presenti
 - Iscrivere al sito
- I giornalisti che devono poter effettuare tutte le azioni degli utenti, più le seguenti azioni:
 - Scrivere un articolo nuovo, indicando la categoria a cui appartiene e che argomenti tratta.
 - Vedere i commenti presenti associati a un certo articolo
 - Scrivere un nuovo commento ad un articolo
 - L'amministratore del sito che oltre alle azioni permesse ai giornalisti, può effettuare le seguenti azioni:
 - Aggiungere uno degli utenti iscritti all'elenco dei giornalisti
 - Rimuovere uno dei giornalisti
 - Mandare una e-mail a tutti i giornalisti del sito, o a tutti gli utenti iscritti o ad entrambi.
 - Cancellare uno degli articoli presenti
 - Cambiare di categoria uno degli articoli presenti
 - Vedere l'elenco degli utenti iscritti

3 Il progetto Capri 2.0

Inoltre in futuro deve esserci la possibilità di aggiungere una nuova figura, il caporedattore, che ha la possibilità di effettuare le seguenti azioni:

- Vedere l'elenco degli articoli scritti e non ancora controllati
- Approvare per la pubblicazione sul sito un articolo
- Negare la pubblicazione sul sito ad un articolo
- Modificare un articolo
- Vedere l'elenco dei giornalisti del sito

3.3 Uso di un CMS per Capri 2.0

Nelle specifiche del progetto era consigliato indagare sull'eventuale uso di un CMS per il sito. E visto che il sito principalmente non fa altro che gestire un certo contenuto, gli articoli, che non sono altro che blocchi di testo, il problema ricade perfettamente nell'area dei problemi risolti dai CMS.

L'uso di un CMS inoltre permetteva sia di diminuire i tempi di sviluppo del progetto, sia di avere una base solida ed estendibile in futuro da cui partire. Inoltre l'esperienza di uso di un CMS avrebbe aumentato il know-how complessivo della ditta, anche per eventuali progetti simili futuri.

Inizialmente mi è stato suggerito un certo CMS (Wordpress), ma prima di usarlo ho preso in esame e confrontato i principali CMS in mercato, anche se alla fine ho scelto proprio Wordpress che si è rivelato il più adatto allo scopo.

3.3.1 Perché usare un CMS?

I motivi per usare un CMS per progettare un sito sono molteplici, così come sono altrettanti i motivi per non usare un CMS per un progetto¹.

Il motivo principale per usarli è che per una certa classe di siti, i CMS offrono un framework già pronto che è una base efficiente da cui partire per sviluppare il sito finale. Questo permette di sviluppare più rapidamente e economicamente il sito per una ampia gamma di siti tipici.

L'altro motivo per cui è consigliato l'uso di un CMS è il fatto che non richiede specifiche conoscenze di programmazione web per un uso "base", se si sceglie di usare template e moduli aggiuntivi open-source o gratuiti, componendoli insieme per raggiungere le funzionalità e l'aspetto che si desidera. Questo permette di sviluppare soluzioni a una buona classe di esigenze senza dover avere persone che abbiano capacità programmatiche.

L'uso di un CMS inoltre permettono di concentrarsi su certi aspetti del progetto, principalmente l'aspetto grafico del front-end o sito vero e proprio, e l'aggiunta delle eventuali funzionalità non offerte dal CMS, relegando una gran parte del problemi al CMS stesso (la creazione dell'interfaccia amministrativa, la gestione vera e propria dei contenuti e della loro organizzazione, parte della business logic), in analogia maniera per cui l'uso di sistemi di RDBMS si è sviluppato negli ultimi anni: suddividere in parti separate un sistema, in cui ogni parte si specializza nel risolvere solo un aspetto specifico del problema. Questo permette di occuparsi solo delle parti specifiche del problema particolare, relegando i problemi generalmente già risolti a prodotti già esistenti, spesso open-source o gratuiti.

Inoltre una buona piattaforma CMS permette di estendere facilmente in futuro le fun-

¹cfr. i due articoli di Claudio Garu, "*Perché utilizzare un CMS*" e "*Perché non utilizzare un CMS*", sul blog onecms.it.

3 Il progetto Capri 2.0

zionalità del sito, tramite aggiunte di moduli di estensione, reperibili esternamente anche gratuitamente, cosa che permette di estendere a basso costo il progetto.

Parlando degli svantaggi di usare un CMS per sviluppare un sito, il problema principale è se la struttura aziendale esistente si adatti all'uso con un CMS, ovvero se la struttura dei documenti da gestire sia effettivamente gestibile da un CMS esistente. Spesso è così, ma in alcuni casi particolari si potrebbero avere workflow o tipi di documenti che richiedono attenzioni specifiche che non sono coperte dai CMS esistenti.

Altro problema è che magari la dimensione del progetto è tale che nessuno dei CMS è adeguato, sia perché magari il sito è risolvibile più semplicemente che usando un CMS complicato, sia perché i CMS "light" esistenti non sono adatti a risolvere il problema per loro mancanze di funzionalità richieste. Insomma, può capitare che il proprio progetto ricada in una zona di mezzo in cui i CMS evoluti sono troppo complessi da usare, ma quelli semplici e leggeri sono troppo limitati da poter essere usati. In questi casi conviene scrivere una soluzione ad-hoc.

Un'altro fattore è che quando si estende e costruisce un template ad-hoc da usare per il progetto partendo da un CMS, c'è una parte iniziale di preparazione e di apprendimento delle strutture e API del CMS che magari diventa più dispendioso che dover fare una soluzione completamente ad-hoc. Questo può capitare nel caso di progetti molto semplici, o se non ci sono disponibili persone con già queste conoscenze, ma magari con capacità di sviluppo ad-hoc di applicazioni web.

Ultima considerazione è che usare un CMS significa affidarsi a un'entità esterna per una parte del proprio progetto, cosa che ha implicazioni sia sull'assistenza che sulla sicurezza del progetto. Non sempre affidarsi a qualcosa scritto da altri è una scelta accettabile, e nel qual caso l'uso di un CMS esistente è ovviamente da escludere.

3.3.2 Rapido confronto fra CMS esistenti sul mercato

Guardando il mercato, fra i CMS che già esistono e che rispondevano ai requisiti richiesti (basati su PHP + MySQL, open source o comunque a costo contenuto, orientati a un uso giornalistico, adeguatamente diffusi e conosciuti) abbiamo selezionato come candidati possibili cinque sistemi: Docebo, Drupal, Joomla, PHP-Fusion, Wordpress. Ho successivamente poi provato ogni sistema, analizzandone la documentazione, il sistema generale di funzionamento e quando rispondeva alle esigenze del progetto.

Docebo è principalmente un sistema orientato all'insegnamento a distanza che ha un modulo CMS interno. Il modulo è abbastanza recente, e soffre di alcune pecche imputabili probabilmente proprio all'essere ancora "acerbo". E' diviso in moduli che offrono le funzionalità (LMS, per l'insegnamento a distanza; CMS per la gestione dei contenuti; ECMS per la gestione di una parte di e-commerce; un sistema di collaborazione fra utenti su progetti) e un nucleo centrale (core) che gestisce gli utenti che possono accedere ai moduli funzionali e la loro organizzazione. La parte CMS propriamente detta organizza il sito in una serie di lingue, in maniera da poter gestire siti localizzati in più lingue, che all'interno comprendono una serie di "macroaree" che raggruppano le pagine vere e proprie. Le macroaree a loro volta sono organizzate come alberi n-ari, in cui ogni macroarea può contenere altre macroaree.

Le pagine vere e proprie sono create scegliendo un layout generale (scelti fra quelli inclusi con docebo, che sono i più comuni a 2 o 3 colonne, o uno definito "personalizzato" che permette di agganciare layout ad hoc) e poi aggiungendo nella pagina una serie di "moduli" che sono il contenuto vero e proprio della pagina. I moduli offrono una ampia gamma di possibilità, da moduli che creano forum interattivi, a moduli che visualizzano una lista di documenti o file scaricabili, a moduli che semplicemente rappresentano un testo scritto tramite un comodo editor AJAX, per finire con i classici moduli per visualizzare un menù di navigazione per raggiungere altre pagine, sia della stessa "macroarea",

3 Il progetto Capri 2.0

sia di “macroaree” superiori o inferiori di livello.

Per ogni modulo poi si può scegliere uno “stile” fra un elenco di stili (di cui 3 offerti con Docebo) che seleziona l’aspetto grafico finale del modulo (colori, caratteri usati, allineamento).

Docebo è considerato uno dei migliori sistemi per LMS/CMS, e ha una solida base di utilizzatori e sviluppatori alle spalle.

Il secondo CMS analizzato è Drupal, uno dei più diffusi CMS usanti PHP. Drupal nasce come CMS generico, basandosi su una grande modularità e flessibilità. E’ un CMS molto solido e flessibile.

Drupal divide i siti in “nodi”, ognuno contenente una informazione precisa, di un “tipo”. I “tipi” di informazioni di Drupal non sono altro che una serie di filtri, che non sono altro che funzioni PHP rispondenti a una certa API, che preso il testo dell’informazione memorizzata, lo elaborano in qualche maniera, eventualmente passandolo (piping) a un’altro filtro. Il risultato finale dei filtri collegati al tipo di informazione è il codice HTML che sarà mandato al browser.

Inoltre ai tipi di dato sono associati una serie di informazioni ausiliarie definibili dall’utente a piacere, utili per definire ad esempio una vetrina di prodotti (creando un tipo di dato “articolo” con associati come campi ad esempio il prezzo e la disponibilità). Inoltre è possibile abilitare un elenco di commenti associati a ogni nodo di un certo tipo.

Drupal permette di aggiungere facilmente nuovi tipi di nodi indicando la catena di filtri associata a quel tipo di nodo, e di aggiungere nuovi filtri definendo delle funzioni PHP secondo una certa API.

I dati finali dei vari nodi sono infine “in scatolati” in una struttura base della pagina, detta “tema”, che definisce la struttura base di tutte le pagine del sito. Un tema di Drupal è un insieme di “zone” in cui può essere messo del contenuto fisso e definito a livello di sito intero (detti blocchi) o il contenuto specifico di quel nodo. I blocchi sono la parte dove

3 Il progetto Capri 2.0

la struttura di navigazione del sito è definita, e vengono aggiunti indicando in quale zona del tema vanno inseriti. I blocchi offerti di base da Drupal sono menù di link sia a nodi specifici sia a pagine web esterne, elenchi degli ultimi utenti iscritti, ultimi nodi inseriti, e molti altri.

I temi veri e propri sono definiti usando un motore di templating e una serie di template. Il motore di base di Drupal è PHPTemplate, ma ogni tema può definirne uno proprio liberamente indicandone il codice in un file PHP. il PHPTemplate di base di drupal permette a un tema di definire un template generale per la pagina, un file CSS che sarà usato in tutte le pagine, un template per indicare i blocchi come saranno generati, per indicare il contenuto di specifici tipi di dati o di certi nodi precisi come deve essere generato e altre possibilità secondo una struttura ben definita nella documentazione stessa di Drupal². La possibilità di definire liberamente il motore di templating (che deve seguire una API ben definita) permette ai temi di Drupal di personalizzare in maniera molto precisa l'aspetto finale, e di generare anche grafiche molto complesse.

Joomla è un'altro CMS basato su PHP molto diffuso e con ottima nomea. I contenuti che gestisce sono definiti "articoli" e sono contenuto web generale (limitato però a testo formattato e immagini), dotato di titolo. La struttura dati del contenuto è una semplice struttura gerarchica a due livelli: "sezioni" che contengono un insieme di "categorie", quest'ultime contenenti un insieme di "articoli". Questa struttura, almeno nella versione a disposizione al momento dell'inizio del progetto, non è modificabile in alcuna maniera.

Le pagine web sono descritte da una struttura base (template), dal contenuto previsto per quella pagina e da un insieme di "moduli" che sono parti del sito messe in posizioni definite dalla struttura base, e che comprendono anche l'elemento base di navigazione nel sito gestito da Joomla, i menù.

Le voci dei menù possono poi collegare sia a sezioni, categorie e articoli specifici che ad

²cfr. Drupal Guide, "*Anatomy of a Drupal Theme*", <http://drupal.org/node/171194>

3 Il progetto Capri 2.0

altri elementi gestiti da quelli che Joomla definisce “componenti”. I componenti sono entità che creano appunto nuovi tipi di contenuto specifico. Joomla prevede che l’utente possa aggiungere sia “componenti” per definire nuovi tipi di contenuto gestito che “moduli” per aggiungere nuove funzionalità generiche alla pagina.

Il prossimo CMS preso in analisi è PHP-Fusion. Nasce come CMS che sia minimale e compatto, e che permetta a tutti velocemente di gestire un sito. In questa ottica le sue caratteristiche principali sono la compattezza (la dimensione complessiva del codice è di soli 2 mega, circa la metà degli altri) e l’essere ready-to-go praticamente subito. Di base gestisce articoli, che sono codice html limitato a testi e immagini; news, che sono semplici blocchi testuali con una data di inizio e di fine validità; un insieme a piacere ampio di forum con tanto di gestione di moderatori anche separati per ogni forum; un insieme di file scaricabili classificati per categorie; un insieme di album di foto con tanto di galleria e infine un insieme di web links, ordinati per categorie.

I limiti di PHP-Fusion sono derivati dalla sua impostazione. La struttura dei contenuti è fissa, e non cambiabile: articoli, news e link sono organizzati in maniera gerarchica in categorie, dove ogni articolo appartiene solo a una categoria; i file scaricabili hanno un insieme fisso di attributi, e anch’essi sono organizzati gerarchicamente in sezioni, dove ogni file appartiene a una sezione che comprende più file; e le foto sono organizzate anch’esse gerarchicamente in album. Inoltre tutte le gerarchie sono a un solo livello, cosa che in molti casi può essere insufficiente.

La navigazione nel sito è guidata da un menù principale, che è creato da PHP-Fusion secondo i contenuti che sono stati aggiunti al sito. Inoltre, è presente una serie di “pannelli”, che sono delle funzioni PHP che generano del codice, e che sono posizionabili a piacere dell’utente in una delle due colonne (sinistra e destra) del sito. PHP-Fusion arriva con una serie di “pannelli” pre-esistenti, che coprono il menù principale di navigazione - modificabile aggiungendo o togliendo voci composte da un nome e dall’URL a cui puntano, e che arriva inizialmente con una serie di voci già stabilite - e tutta un’altra serie

3 Il progetto Capri 2.0

di informazioni utili (elenco degli ultimi utenti iscritti, elenco degli utenti attualmente connessi al sito, casella con i sondaggi attualmente attivi, e altri).

Per quanto riguarda la personalizzazione della grafica, PHP-Fusion usa un sistema a template, dove il sito ha associato un template globale. I template di PHP-Fusion però possono solo modificare l'aspetto e la struttura di come sono visualizzati i vari tipi di dati (news, web links, ecc.), definendo una serie di funzioni PHP secondo una API definita, ma non la struttura base (layout) del sito, che è una struttura fissa con intestazione, tre colonne e piede. I template possono al più specificare il colore di sfondo globale e le dimensioni delle colonne. Inoltre il layout base, fisso, è strutturato usando le tabelle, cosa che è inaccettabile per un sito moderno ed accessibile, come detto in [5].

PHP-Fusion infine offre la possibilità di estendere i tipi di dati che gestisce con dei moduli PHP che sono detti "infusions", e che permettono di aggiungere nuove funzionalità e tipi di dato trattati.

Wordpress è un CMS che nasce come "motore" per blog, e infatti il contenuto che gestisce è diviso in due classi: post e pagine. I primi sono blocchi di codice HTML, solitamente inserito tramite un editor visuale, dotate di un titolo e di una data, a cui possono essere aggiunti dei commenti. I post inoltre hanno associati dei metadati, di base una serie di tag che organizza i post in una gerarchia piatta e una categoria, che li organizza invece secondo una struttura gerarchica. I secondi sono pagine PHP richiamabili, con un titolo.

Wordpress ha dei meccanismi per estendere le proprie funzionalità, detti "plug-in", che non sono altro che codice PHP organizzato in funzioni, e a cui Wordpress offre delle API per interagire con i contenuti che gestisce in varie maniere.

Per quanto riguarda la personalizzazione dell'interfaccia con l'utente, Wordpress prevede dei "template" o "temi" che non sono altro che pagine PHP organizzate secondo una certa struttura, e che Wordpress richiama per visualizzare i dati. Per accedere ai contenuti,

3 Il progetto Capri 2.0

Wordpress offre delle API precise a queste pagine PHP, dette “template tags”, ma per il resto il progettista del template ha la più piena libertà.

Pur essendo nato come gestore di blog, Wordpress sta iniziando a offrire funzionalità che lo rendono un CMS per un uso anche più ampio, quali le “pages” per gestire contenuto che non sia strettamente un post di un blog, e una sofisticata organizzazione degli utenti in una serie di classi aventi delle capacità, che permette di regolare in maniera molto precisa cosa possono o non possono fare ogni classe di utenti.

L’altro punto di forza di Wordpress è la sua documentazione, estesa e organizzata ottimamente, liberamente consultabile dal web, e l’ampia comunità di utilizzatori, che ha costruito una notevole quantità di risorse liberamente usufruibili, sia come template/temi sia come plug-in per aggiungere funzionalità di svariati tipi.

3.3.3 Perché abbiamo usato Wordpress per Capri 2.0

Analizzando i vari sistemi, il sistema che ho scelto alla fine è stato Wordpress. La scelta è stata dettata da una serie di fattori, che illustrerò in questa sezione.

Iniziando da Docebo, è sicuramente un’ottimo sistema nell’ottica dell’uso per cui è nato, ovvero per realizzare un sistema di insegnamento a distanza in maniera rapida, ma robusta e con tutte le funzionalità che possano essere necessarie. Ma ho trovato la documentazione sulla parte CMS non adeguata. In particolare non è chiaro, nè illustrato sufficientemente, come creare layout ad hoc per le pagine, cosa che il progetto necessitava, nè come aggiungere “stili” per i moduli, nè come far ereditare gli stili e i moduli alle pagine. Inoltre la gestione a moduli delle pagine, che se da un lato è perfetta nell’ottica di chi voglia sviluppare rapidamente un sito agganciato a un sistema di insegnamento a distanza senza particolari esigenze di grafica, mi è risultato intralciante per sviluppare un layout e una grafica di una certa complessità come quella richiesta dal progetto. Questa serie di motivi mi ha fatto escludere Docebo come scelta finale.

3 Il progetto Capri 2.0

Drupal, il secondo CMS preso in considerazione, rispondeva in quasi tutte le caratteristiche. E' sicuramente un CMS molto flessibile e ottimale in molte situazioni, ma mancava in una delle richieste del cliente (almeno nella versione disponibile quando si è realizzato il progetto) ovvero la possibilità di avere indirizzi di accesso al sito chiari usando il modulo di riscrittura dell'url di Apache (modrewrite). Inoltre l'editor che usa per l'inserimento delle pagine è uno dei peggiori rispetto a quello usato dagli altri, non usando un editor AJAX ma una semplice casella di inserimento del testo.

Joomla è un buon CMS, ma i limiti di flessibilità, soprattutto per struttura del contenuto gestito, non mi ha fatto scegliere lui come CMS da usare, visto che alternative migliori sono emerse successivamente nel confronto. Inoltre manca di alcune funzionalità richieste - in particolare la gestione degli articoli con una gerarchia piatta, non erano disponibili nativamente al momento dell'inizio del progetto. Queste lacune, di scarsa flessibilità e di mancanza di alcune funzionalità, mi hanno fatto preferire altri CMS a Joomla.

PHP-Fusion, nell'attica di un CMS compatto e ready-to-go, riesce efficacemente nelle sue funzioni. In effetti fra tutti i CMS analizzati è quello che con meno configurazione può arrivare a risultati anche gradevoli. I suoi limiti di struttura però lo rendono inutilizzabile per il progetto che mi era stato assegnato, in particolare la mancanza di un workflow degli utenti sugli articoli (impossibilità di moderazione degli articoli), la mancanza della possibilità di creare una gerarchia piatta degli articoli tramite tag, e infine il layout fisso che era in questo caso, vista la richiesta di un aspetto grafico non banale, più di intralcio che di aiuto.

Escludendo gli altri sistemi, Wordpress si è rivelato alla fine quello più adatto. Infatti la maggioranza dei requisiti è gestita nativamente: workflow degli articoli con moderazione, utenti non registrati e registrati con la possibilità di scegliere cosa possono accedere, gestione di una gerarchia piatta degli articoli (tramite tag), divisione in categorie degli articoli, articoli con commenti eventualmente accedibili solo a certi utenti, un sistema di amministrazione che permette con semplicità di aggiungere nuovi articoli.

3 Il progetto Capri 2.0

Inoltre Wordpress ha un sistema di templating che è quello che permette maggiore personalizzazione in assoluto e che cerca di ostacolare il meno possibile lo sviluppatore, dandogli piena libertà sul layout e sulla grafica del sito. Questo è l'aspetto che mi ha definitivamente convinto a usare Wordpress: lo scrivere il template è stato semplicemente scrivere le pagine php usando le funzioni che il framework di Wordpress mi offriva, ma potendo in ogni istante usare codice php ad-hoc per aggiungere funzionalità particolari. Il non avere delle strutture fisse mi ha permesso più liberamente di usare il CSS per formattare e costruire un layout finale non banale ma molto sofisticato, senza sentirmi ostacolato dal CMS e da eventuali limiti da rispettare.

E per le funzionalità che non erano gestite nativamente, Wordpress offriva un meccanismo di estensione (plugin) che era molto bene documentato, e che mi ha permesso in pochi giorni di aggiungere le funzionalità che servivano, in maniera semplice e sempre avendo un'ottima assistenza dalla comunità che ruota intorno a Wordpress.

Per queste ragioni, alla fine ho scelto come CMS con cui sviluppare il progetto proprio quello che mi era stato suggerito inizialmente, Wordpress.

3.4 Studio dell'Implementazione di Capri 2.0 usando

Wordpress

Una volta scelto Wordpress come CMS, ho iniziato a organizzare cosa andava scritto per completare il progetto usandolo come base.

Lasciando a Wordpress tutta la sezione di gestione degli articoli, degli utenti e della parte amministrativa, mi sono concentrato inizialmente sulla parte di design del sito come grafica e layout, scegliendo fin da subito di sviluppare un sito che fosse tableless e fluido (quest'ultima scelta alla fine l'ho limitata per richiesta del cliente) e rispettante

il più possibile gli ultimi standard in fatto di siti web (CSS e XHTML 1.1 stretto come codice finale prodotto).

3.4.1 Funzionalità richieste dal cliente e implementate da Wordpress

Guardando Wordpress, ho iniziato a vedere quali funzionalità erano già offerte da Wordpress fra quelle presenti nelle specifiche. Le funzionalità che ho rilevato erano già offerte da Wordpress erano le seguenti:

- Definizione di un insieme di giornalisti, che devono essere gli unici a poter aggiungere articoli al sito. Wordpress permette di assegnare ogni utente a un “ruolo”, e i ruoli predefiniti coprono anche un ruolo che permette all’utente di scrivere articoli direttamente pubblicati sul sito.
- Gli articoli devono essere divisi in categorie, e l’utente deve poter vedere tutti gli articoli di una certa categoria.
- Il sito deve usare modrewrite di Apache in maniera che gli indirizzi degli articoli siano chiari e comprensibili, tipo “http://www.capri20.it/2008/01/04/musica/concerto_di_branduardi.p”. Wordpress permette di definire vari schemi dell’URL finale, più la possibilità di crearne uno ad-hoc.
- Possibilità agli utenti di iscriversi al sito.
- Gli articoli hanno associati una serie di argomenti (tags), e deve essere possibile effettuare ricerche per argomenti o vedere gli articoli associati a un certo argomento.
- Per gli utenti semplici, le seguenti funzionalità erano già risolte:
 - Vedere quali articoli sono in prima pagina
 - Vedere quali articoli sono presenti in una certa categoria
 - Vedere l’elenco di quali categorie esistano
 - Leggere un articolo preciso

3 Il progetto Capri 2.0

- Cercare fra gli articoli quali abbiano presenti alcune parole chiave
 - Vedere quali articoli parlano di un certo argomento
 - Vedere quali argomenti sono i più trattati nell'ultima settimana
 - Vedere l'elenco dei giornalisti presenti
 - Iscrivere al sito
- Per i giornalisti, le seguenti funzionalità erano già presenti in Wordpress:
 - Scrivere un articolo nuovo, indicando la categoria a cui appartiene e che argomenti tratta.
 - Vedere i commenti presenti associati a un certo articolo
 - Scrivere un nuovo commento ad un articolo
 - Infine per l'Amministratore del sito, le seguenti funzionalità erano già pronte:
 - Aggiungere uno degli utenti iscritti all'elenco dei giornalisti
 - Rimuovere uno dei giornalisti
 - Cancellare uno degli articoli presenti
 - Cambiare di categoria uno degli articoli presenti
 - Vedere l'elenco degli utenti iscritti

Come si vede, una gran parte delle funzionalità richieste è stata immediatamente coperta da Wordpress.

3.4.2 Funzionalità richieste da implementare estendendo Wordpress

Le funzionalità rimanenti da implementare erano quindi:

- La possibilità dell'amministratore di mandare tramite una interfaccia semplice una e-mail a tutti gli utenti iscritti al sito, o a tutti i giornalisti, o ad entrambi.

- L'averne un elenco degli articoli più letti in un certo arco temporale (ultimi 5 giorni, ultima settimana, ultimo mese).
- L'averne un elenco degli articoli più commentati in un certo arco temporale.
- L'averne degli articoli "in evidenza", che non vengano visualizzati nella prima pagina normalmente, ma siano messi una certa zona della prima pagina.
- La possibilità di segnalare via e-mail un articolo.

Inoltre Wordpress aveva aggiunte delle funzionalità che non erano desiderate, e che complessivamente erano solo complicazioni aggiuntive all'interfaccia utente, soprattutto per la parte amministrativa. Queste funzionalità sono state nascoste o rimosse in maniera opportuna.

3.5 Implementazione delle funzionalità particolari richieste da Capri 2.0

Ho dovuto quindi aggiungere le funzionalità mancanti a Wordpress, ma richieste dal progetto Capri 2.0, evidenziate in 3.4.2 nella pagina precedente. Ho cercato di farlo sfruttando i naturali meccanismi previsti da Wordpress per questo, ovvero i template e i plugins. Quest'ultimi li approfondiremo in questa sezione, mentre i primi li vedremo nel capitolo successivo.

3.5.1 Come estendere le funzionalità di Wordpress?

Wordpress come ogni CMS moderno prevede dei meccanismi appositi per estendere le sue funzionalità. Uno di questi meccanismi sono i cosiddetti plugin.

I plugin di Wordpress sono definiti come:

3 Il progetto Capri 2.0

“un programma, o un insieme di una o più funzioni, scritte nel linguaggio di script PHP, che aggiunge un insieme specifico di funzionalità o servizi a Wordpress, che possono essere integrati completamente nel motore di Wordpress usando i punti di accesso e i metodi specificati nell’interfaccia per le applicazioni plugin di Wordpress (WP-API)” [6].

I plugin sono gestiti da Wordpress stesso, con possibilità di attivarli, disattivarli e aggiornarli facilmente dalla interfaccia amministrativa di Wordpress.

L’altro modo di estendere funzionalità di Wordpress è la modifica del codice di Wordpress stesso (codehack). La cosa è fondamentalmente sconsigliata, ma è stata necessaria in certi casi specifici, quasi sempre per togliere delle funzionalità che non erano necessarie e inutili.

Per le altre funzionalità è stato costruito un plugin che comprende tutte le funzionalità extra che erano state richieste, ma non implementate nativamente da Wordpress.

3.5.2 Struttura dei plugin di Wordpress

La struttura base di un plugin di Wordpress, descritta in [6], è molto semplice: una cartella dello stesso nome del plugin, un file PHP con il codice del plugin vero e proprio e con nome uguale a quello del plugin, ed eventuali altri file accessori usati dal plugin.

La cosa fondamentale è che all’inizio del file PHP ci sia una intestazione secondo una struttura specifica, che indica i dati fondamentali del plugin. L’intestazione del nostro plugin è la seguente:

```
<?php
/*
Plugin Name: Capri20
Plugin URI: http://www.digitalsparks.it
Description: Plugin con le funzionalità ausiliarie
```

3 Il progetto Capri 2.0

richieste per il progetto Capri 2.0

Version: 1.1

Author: Renato Salzano

Author URI: <http://caretaker.altervista.org>

*/

?>

Anche per i plugin Wordpress mette poche strutture fisse, e lascia la più ampia libertà agli sviluppatori. I plugin possono poi usare il framework di Wordpress per accedere al database MySQL e ai dati del contenuto gestito da Wordpress tramite apposite funzioni PHP descritte nella WP-API di Wordpress³. I plugin poi si “agganciano” al motore di Wordpress in tre possibili modalità, come descritto in [7]: come azioni (actions), come filtri (filters) e come nuovi tag per temi (template tag).

Le azioni sono eventi che avvengono in specifiche punti di Wordpress. Un plugin può registrare una sua funzione come da attivare quando una specifica azione avviene. Ad esempio, un plugin può registrare una sua funzione come da chiamare ogni volta che un nuovo articolo viene pubblicato. Oppure quando viene aggiunto un nuovo commento ad un articolo. O anche quando viene chiamata una pagina amministrativa dall'utente. A seconda dell'azione alla quale la funzione si è associata, al momento del richiamo verranno passati determinati parametri, legati all'evento. Ad esempio una funzione che è associata all'azione di un nuovo articolo, riceve come parametro in ingresso i vari dati del nuovo articolo (il suo identificativo, il testo, l'autore, ecc.). In [7] è presente un elenco completo delle azioni presenti alle quali i plugin possono associarsi. Una funzione di un plugin per associarsi a un'azione chiama l'apposita funzione `add_action`, indicando a quale azione associarsi.

I filtri sono funzioni che sono richiamate per elaborare certi dati che Wordpress gestisce,

³Ben documentata, anche se ancora non completamente, sul Wordpress Codex, in particolare in http://codex.wordpress.org/Function_Reference.

3 Il progetto Capri 2.0

in momenti precisi, ad esempio prima di essere messo nel database o prima di essere visualizzato. Le funzioni che fanno da filtro ricevono quasi sempre un testo come argomento in ingresso e Wordpress si aspetta che ritornino un testo, che sarà quello usato alla fine da Wordpress. Praticamente ogni dato ricevuto o emesso da Wordpress nel suo funzionamento normale passa per almeno un filtro. I filtri permettono ad esempio ai plugin di modificare il testo degli articoli (magari per eliminare certi contenuti indesiderati), o dei commenti. Una funzione di un plugin si registra come filtro tramite la funzione `add_filter`, indicando il tipo di filtro a cui vuole essere associato. L'elenco dei tipi di filtro supportati da Wordpress è in [7].

L'ultima possibilità di un plugin per interagire con Wordpress è quella di definire una funzione come un nuovo tag per template. Per fare questo, il plugin definisce semplicemente la funzione PHP, e la documenta sul sito associato al plugin. Toccherà poi agli sviluppatori dei template richiamare la funzione dai template stessi, come vedremo in dettaglio in 3.6.1 nella pagina 50.

Inoltre un plugin ha anche la possibilità di aggiungere delle nuove pagine nella parte amministrativa di Wordpress, per definire le proprie opzioni o altre interazioni con l'utente amministratore. Questo avviene definendo delle funzioni che emettono il codice delle pagine web da aggiungere nella parte amministrativa, e poi collegando queste funzioni ai menù della parte amministrativa di Wordpress, chiamando due funzioni descritte in [8], `add_menu_page` per aggiungere una nuova voce nel menù principale della parte amministrativa e `add_submenu_page` per aggiungere invece una voce nuova come sottomenù di una delle voci principali della parte amministrativa di Wordpress. Al resto pensa Wordpress, che aggiungerà i link per richiamare le pagine aggiunte automaticamente, e richiamerà le funzioni associate quando l'utente le visiterà.

Questo però evidenzia un piccolo svantaggio: a ogni pagina si può associare solo una funzione, che quindi nel caso di pagine che ricevono dati dall'utente, dovrà fare attenzione a vedere se è stata chiamata dopo che l'utente ha inserito dei dati o se è la prima

volta che è stata chiamata, distinguendo i due casi guardando se ci sono dei dati che sono stati passati a PHP tramite metodi POST o GET, usando i due vettori `_POST` e `_GET` e regolandosi di conseguenza. Wordpress offre anche un meccanismo di sicurezza per assicurarsi che la funzione sia stata chiamata proprio da Wordpress, conosciuto come “nonce”. In pratica all’eventuale form presente sulla pagina viene aggiunto un campo nascosto, creato dalla funzione `wp_nonce_field`, che ha un valore che dovrebbe essere unico per quel plugin. Un’altra funzione, `check_admin_referer` controlla innanzitutto tramite gli header HTTP referer che la provenienza sia quella giusta, poi controlla se è stato ricevuto il campo nascosto e che il valore combaci. In caso contrario, esce dalla funzione, e richiama una pagina apposita di Wordpress che segnala il tentativo di intrusione.

3.5.3 Plugin per la funzionalità statistica sugli articoli

Era richiesta dal progetto l’implementazione di una funzionalità statistica, e in particolare l’elenco degli articoli più letti di recente, e l’elenco degli articoli più commentati di recente. Per implementare queste due funzionalità, abbiamo creato due funzioni nel plugin, e semplicemente poi le abbiamo richiamate dal template costruito appositamente nel progetto.

Per gli articoli più commentati, i dati necessari (commenti associati agli articoli, data di aggiunta di ogni commento) erano già nella base dati creata da Wordpress. In particolare Wordpress crea una tabella contenente i dati degli articoli, con una chiave primaria che è un identificatore dell’articolo, e una tabella con i dati dei commenti, collegati agli articoli tramite una chiave secondaria “comment_post_ID”. Il plugin quindi fa una query sulla base dati, passando l’apposito oggetto `$wpdb` documentato in [9] che Wordpress fornisce ai plugin per accedere alla base dati, facendo una join fra le due tabelle e estraendo i dati degli articoli con più commenti che abbiano la data di pubblicazione oltre la data attuale meno un certo numero di giorni (che è uno dei parametri passati alla funzione).

3 Il progetto Capri 2.0

Questa query non è stata banale da costruire, e usa una colonna ausiliaria calcolata tramite una sottoquery che usa la funzione COUNT di SQL. In pratica effettuando la query:

```
SELECT count(*) FROM wp_comments WHERE id_post_comment=1
```

Otteniamo quanti commenti in complessivo ha associato l'articolo con identificativo 1. Limitando con un'apposita clausola WHERE solo ai commenti che siano stati postati negli ultimi ad esempio 5 giorni, tramite la query:

```
SELECT count(*) FROM wp_comments  
WHERE (id_post_comment=1) AND (datediff(now(), comment_date)<5)
```

Abbiamo il numero di commenti “recenti” associati all'articolo con identificativo 1. Inserendo questa sottoquery dentro una query che esplori tutti gli articoli abbiamo il numero di commenti negli ultimi 5 giorni associato a ogni articolo, tramite la query:

```
SELECT ID, post_title, (SELECT count(*) FROM wp_comments  
WHERE (id_post_comment=ID) AND (datediff(now(), comment_date)<5))  
AS conto
```

A questo punto è bastato riordinare la query in ordine decrescente sulla colonna “conto” e limitare la query a un certo numero di articoli. La funzione aveva come parametri in ingresso proprio il numero massimo di articoli da estrarre e in numero di giorni da usare per determinare se il commento era recente, e usando la query illustrata ricava i dati degli articoli più commentati e li restituisce come una lista non ordinata XHTML (usando i tag UL e LI).

Per quanto invece riguarda l'elenco degli articoli recenti più visti, si è aggiunta una tabella ausiliaria creata dal plugin alla base dati di Wordpress, come indicato in [10], tabella usata per contare il numero di visualizzazioni di ogni articolo. La tabella è stata creata cercando di minimizzare il suo numero di righe. Quindi è stato deciso di contare

3 Il progetto Capri 2.0

quante visualizzazioni ogni articolo aveva avuto per ogni giornata, usando come colonne quindi una data (`view_date`), l'identificativo dell'articolo (`date_post_ID`) e il numero di visitatori (`view_hits`). Inoltre è stato aggiunto un identificativo unico per ogni riga, per facilitare MySQL (`view_ID`).

Poi il plugin aggiunge due funzioni, una richiamata dal template quando viene visualizzato un articolo, e una che restituisce l'elenco di articoli più letti negli ultimi n giorni. La prima delle due funzioni non fa altro che fare una query sul database cercando se c'è già una riga con la data di oggi e l'ID dell'articolo letto, e nel qual caso aggiunge 1 al numero di visite. Altrimenti crea la nuova riga, con la data di oggi e l'ID dell'articolo letto, e con settato a 1 il numero di visite.

La seconda delle funzioni fa una query sulla tabella creata, anche in questo caso usando una sottoquery per avere il totale delle visualizzazioni. In pratica effettuano la query SQL:

```
SELECT sum(view_hits) FROM views
WHERE view_post_ID=1 AND datediff(now(), view_date)<5
GROUP BY view_post_ID
```

Otteniamo il numero di visualizzazioni effettuate negli ultimi 5 giorni dell'articolo con identificativo 1. Usandola come sottoquery di una query che scorra tutta la tabella degli articoli, otteniamo la seguente query:

```
SELECT ID, post_title, (SELECT sum(view_hits)
FROM views
WHERE view_post_ID=ID AND datediff(now(), view_date)<5
GROUP BY view_post_ID)
AS conto
FROM views, wp_post
WHERE view_post_ID=ID
```

3 Il progetto Capri 2.0

Otteniamo quante visualizzazioni negli ultimi 5 giorni hanno avuto ogni articolo. Riordinando la query sul numero di visualizzazioni in maniera decrescente e limitando ai primi N risultati, abbiamo quindi i primi N articoli più visti negli ultimi 5 giorni.

La funzione riceve come parametri il numero di giorni e quanti risultati estrarre, e tramite la query illustrata estrae i dati degli articoli e li restituisce come una lista non ordinata XHTML (usando il tag UL e LI).

Queste tre funzioni sono poi chiamate nei punti appositi dal template scritto appositamente, come illustrato in 3.6.1 nella pagina 50.

3.5.4 Plug-in per la funzionalità di spedizione di e-mail a più utenti

Per aggiungere questa funzionalità si è dovuto aggiungere una pagina alla sezione amministrativa, seguendo il metodo precedentemente illustrato nella pagina 41. Il plugin per farlo quindi ha una funzione che emette il codice XHTML della pagina aggiunta.

La pagina aggiunta ha un form XHTML che chiede all'utente i dati della e-mail da spedire, titolo e corpo, usando un editor javascript open-source diffuso, TinyMCE⁴, per il corpo, in maniera da avere una interfaccia più semplice e che permetta di inserire del testo formattato XHTML. Nel form poi viene chiesto all'utente di scegliere quali destinatari devono ricevere la e-mail scegliendo una o più classi di utenti di Wordpress (si sono usate quelle standard di base di Wordpress⁵), e che indirizzo e-mail usare come mittente fra tre scelte possibili: l'indirizzo e-mail dell'utente che in questo momento sta usando la pagina (solitamente l'amministratore) come registrato al momento dell'iscrizione al sito, un indirizzo fittizio "noreply" sul dominio attuale o un indirizzo completamente fittizio "noreplay@example.invalid"⁶. Il form ha infine due campi nascosti, uno "action"

⁴Vedi il suo sito ufficiale, <http://tinymce.moxiecode.com/>

⁵Amministratore, Autori, Editori, Contributori e Utenti, come illustrate in [11]

⁶Non scelto casualmente, ma seguendo le indicazioni apposite sui domini invalidi o fittizi contenute nella RFC 2606.

3 Il progetto Capri 2.0

con valore “sendmail” e uno aggiunto dalla funzione `wp_nonce_field`, per controllo di sicurezza, come illustrato precedentemente nella pagina 42.

La funzione che genera la pagina controlla se ha ricevuto dati controllando il vettore `_POST`, descritto in [12], e se fra i campi ricevuti c'è il campo “action” con valore “sendmail”, allora fa una verifica di sicurezza con `check_admin_referer`, e preleva i vari campi del form, costruendo la mail.

Per estrarre i riceventi fra delle apposite query sulla tabella degli utenti di wordpress, estraendo gli indirizzi e-mail di tutti gli utenti che siano nelle classi selezionate, e aggiungendole al campo “bcc:” della mail, per non diffonderli a chi non ne ha diritto, visto che gli indirizzi e-mail sono dati protetti dalla legislazione sulla privacy italiana⁷.

Come mittente della mail viene settato come scelto nel form (eventualmente prelevando l'indirizzo dell'utente corrente dalla variabile `$user_email`, descritta in [9]), e come destinatario nel campo “to:” della e-mail è settato stesso il mittente. Poi la email viene spedita usando la funzione “mail” di PHP.

3.5.5 Localizzazione del plug-in

Per localizzazione, intendiamo la capacità di un programma di dialogare con l'utente in una lingua a scelta dell'utente, solitamente quella nativa dell'utente stesso. Questa capacità, sempre più essenziale per un programma moderno, deve essere prevista fin dall'inizio, adottando appositi meccanismi e tecniche. Nel caso del progetto Capri 2.0, era necessaria una localizzazione almeno in italiano, e la possibilità in futuro di aggiungere nuove lingue.

Wordpress ha un meccanismo di localizzazione (o internazionalizzazione) che viene offerto a tutti i plugin, descritto in [6], che è basato su un meccanismo standard di loca-

⁷Legge n. 675 del 1996, e ribadito dal garante della privacy Rodotà nel ricorso del 25 giugno 2002 reperibile a <http://www.garanteprivacy.it/garante/doc.jsp?ID=29864>

3 Il progetto Capri 2.0

lizzazione che è GNU gettext⁸. Il meccanismo offre una funzione che emette una stringa identificata da un codice prelevandolo da un file apposito, in un formato apposito standard (POT), che definisce un insieme di coppie chiave (detto messaggio originale) - stringa nella lingua scelta (detto messaggio tradotto). Di questi file ce ne possono essere vari, ognuno associato a una lingua specifica e a un certo “dominio” che identifica il programma a cui sono associati, e il file preciso da usare è scelto da Wordpress attraverso una delle sue opzioni di configurazione (la lingua nativa appunto da usare).

In pratica si tratta nel codice del plugin di chiamare una delle due funzioni:

```
__($messaggio, $dominio)
_e($messaggio, $dominio)
```

In cui in entrambe \$messaggio indica quale chiave usare nel file (ovvero qual’è la stringa da tradurre) e \$dominio indica il dominio testuale da usare - in pratica è una stringa unicamente associata a questo plugin. La differenza fra le due funzioni è che nel primo caso la stringa tradotta è passata come valore di ritorno, nel secondo caso è emessa a video (o nello stream di uscita predefinito).

Il plugin infine deve, prima dell’esecuzione delle due funzioni precedenti, chiamare una apposita funzione, che carica il file POT da usare:

```
load_theme_textdomain('dominio');
```

Dove il dominio da passare è lo stesso che poi sarà usato nella chiamata delle funzioni precedenti.

La localizzazione, per come illustrato, preferisce quindi che si usano un insieme preciso e ben definito di messaggi, non troppo ampio, in maniera da ridurre l’insieme di coppie da fornire nei file POT. Nel caso dei nostri plugin l’elenco dei messaggi è stato principalmente all’unica funzionalità aggiunta che abbia una vera interazione con l’utente, ovvero la

⁸Sito ufficiale: <http://www.gnu.org/software/gettext/>

3 Il progetto Capri 2.0

spedizione di una e-mail a tutti gli utenti di un insieme di categorie. I messaggi selezionati, originalmente etichettati in inglese, sono:

- Il nome del menù da aggiungere nella parte amministrativa
- I due messaggi che indicano se la mail è stata spedita con successo o meno
- Le intestazioni del form per mandare la e-mail: “e-mail data”, “Destinations”, “Mail address”, ecc.
- I valori di alcuni campi a scelta del form precedente: “All Administrators”, “Current user e-mail”, ecc.

Una volta selezionato questi messaggi, si è sostituito nel codice del plugin i riferimenti assoluti a queste stringhe cambiandoli con le giuste chiamate alle due funzioni illustrate precedentemente, e per finire si è creato il file POT, con gli appositi strumenti standard⁹ forniti dal sistema operativo in cui ho scritto il codice, con la traduzione delle stringhe in lingua italiana. Al resto ci ha pensato la struttura di WordPress.

Aggiungere altre lingue consiste semplicemente nell’aggiungere nuovi file POT, con le traduzioni dei termini.

3.5.6 Plug-in per la funzionalità di segnalazione dei nuovi articoli

L’ultima funzionalità che mi è stata richiesta e che era da implementare era la possibilità di generare automaticamente delle e-mail che riassumessero gli articoli inseriti nel corso dell’ultima settimana, e-mail che poi dovevano essere automaticamente spedite agli utenti iscritti con cadenza settimanale.

Per implementare questa funzionalità, prima di scrivere codice ad hoc, ho controllato se esisteva già un plug-in per Wordpress che svolgesse la funzione. Esaminando la lista

⁹nel mio caso ho usato “poedit”, ma ci sono moltissimi programmi per creare i file POT, per vari sistemi operativi.

dei plug-in disponibili per Wordpress¹⁰, ho trovato un plugin, Subscribe2¹¹, che svolgeva esattamente le funzionalità richieste.

Dopo aver verificato la licenza d'uso del plugin, ho semplicemente preso il pacchetto del plugin e installato dentro Wordpress. Il plugin non solo copre la funzionalità richiesta, ma permette agli utenti moltissime altre possibilità, come il ricevere solo notifica degli articoli appartenenti a certe categorie specifiche, e di scegliersi con che cadenza le e-mail vengono spedite.

3.5.7 Funzionalità implementate modificando Wordpress (codehack)

In alcuni rari casi, nè i plugin, nè il template mi permetteva di aggiungere alcune funzionalità che erano state richieste. In particolare, la semplificazione dell'interfaccia amministrativa ha richiesto di modificare Wordpress stesso, per aggiungere classi CSS ad alcuni elementi dell'interfaccia amministrativa, o per cancellare alcune funzionalità non richieste dal cliente.

Alcuni di questi interventi, che ho cercato comunque di ridurre all'essenziale, sono state cancellare alcuni pulsanti dall'editor usato da Wordpress per inserire gli articoli (il pulsante per vedere e modificare direttamente il codice HTML finale dell'articolo ad esempio), che ha richiesto una modifica¹² per cancellare alcuni parametri passati all'editor (che è un editor scritto in javascript e che usa le DOM per modificare gli oggetti visualizzati dal browser, TinyMCE¹³), e per differenziare come classe CSS due pulsanti, di cui di default WordPress assegna la stessa classe CSS (il pulsante "salva" degli articoli, e quello "pubblica" delle pagine statiche, che pur essendo simili hanno funzioni diverse).

Altro intervento che ha richiesto la modifica del codice è stato l'aggiunta di un nuovo piè

¹⁰ disponibile sulla Wordpress Plugin Directory, all'indirizzo <http://wordpress.org/extend/plugins/>

¹¹ Sito ufficiale: <http://subscribe2.wordpress.com>

¹² In particolare dei file `edit-page-form.php` e `edit-form-advanced.php` in `wp-admin`

¹³ la documentazione che mi è servita per vedere quali opzioni togliere si trova su <http://tinymce.moxiecode.com/>

di pagina alle pagine web dell'amministrazione, che ha richiesto una modifica diretta dei file di Wordpress relativi¹⁴ per aggiungere il contenuto del nuovo piè di pagina, e infine la cancellazione delle parti riguardanti gli indirizzi degli instant messaging nei profili degli utenti, ottenuta commentando il codice XHTML relativo nei file di Wordpress¹⁵.

3.6 Progettazione e implementazione del front-end verso

l'utente di Capri 2.0

In questa sezione illustrerò come è avvenuta la scelta e progettazione di quello che alla fine ha richiesto più tempo di lavoro, ovvero l'interfaccia con l'utente del sito, o front-end, o grafica del sito che si voglia dire.

Spesso questa parte è quella più sottovalutata, ma nei progetti reali come questo è la parte che più ha impatto finale con l'utente, e quindi quella sulla quale alla fine si spende la maggioranza del tempo del progetto.

3.6.1 Meccanismo generale di Wordpress per modificare i front-end

utente: Themes

Wordpress ha un meccanismo di template, illustrato in [13], per definire le pagine web generate verso l'utente. In pratica Wordpress usa una serie di file PHP che vengono richiamati quando Wordpress deve generare le pagine web finali mandate all'utente. Questi file PHP vengono richiamati in un certo ambiente software, in cui esistono sia una serie di oggetti contenenti i dati che la pagina deve includere (ad esempio gli articoli che deve visualizzare, con tutti i loro dati) sia una serie di funzioni già pre-esistenti, che Wordpress chiama "templates tag", elencati in [14], che permettono ai template sia di accedere a una serie di dati gestiti da Wordpress, sia che di agganciarsi nuovamente al motore

¹⁴in particolare il file `index.php` della parte amministrativa di Wordpress, contenuto in `wp-admin`.

¹⁵in particolare il file `profile.php` in `wp-admin`

3 Il progetto Capri 2.0

di Wordpress (ad esempio si pensi alle funzioni che generano i link per andare verso la pagina che visualizza il contenuto di un certo articolo o che generano i form XHTML per fare l'autenticazione dell'utente o la ricerca sugli articoli).

Questo dà molta libertà a chi scrive i template, che non ha restrizioni: in pratica i template sono veri e propri programmi PHP che usano Wordpress come un framework per semplificarsi la vita, permettendo al designer dei template di ottenere template anche molto sofisticati e che magari permettono di usare WordPress anche in contesti non usuali (come ad esempio Mimbo¹⁶ che trasforma WordPress in un motore per magazine grafici on-line).

A seconda del tipo di dato che Wordpress deve trasmettere all'utente, viene in particolare cercato un certo file PHP nel template, secondo uno schema ben preciso di priorità, illustrato in [13]. A questi file PHP inoltre Wordpress associa di base un file CSS, che imposta l'aspetto grafico della pagina, anch'esso definito dal template.

3.6.2 Design iniziale del Theme per Capri 2.0

Per effettuare il design del theme wordpress per Capri 2.0, si è iniziato con il verificare quale contenuto doveva essere presente in tutte le pagine finali del sito, sempre a disposizione dell'utente. Dai requisiti richiesti inizialmente, riportati in 3.2 nella pagina 22, i seguenti elementi li ho individuati come da presentare sempre all'utente, in ogni pagina:

- Un elenco delle categorie degli articoli.
- Un link per raggiungere l'elenco di tutti i giornalisti del sito.
- Una casella per autenticarsi con username e password.

A una successiva analisi, sono emersi altri contenuti da fissare nel layout generale del sito, e non nel contenuto delle singole pagine:

¹⁶Reperibile su <http://www.darrenhoyt.com/2007/08/05/wordpress-magazine-theme-released/>

3 Il progetto Capri 2.0

- L'insieme dei tag degli articoli, magari evidenziando quelli con maggiore quantità di articoli collegati.
- Un form per accedere alla funzionalità di ricerca sugli articoli di un termine.
- Un insieme di link esterni collegati al sito.
- I collegamenti ai feed RSS del sito, almeno nei formati RSS1, RSS2 e Atom.
- Il link per accedere a un archivio degli articoli organizzato in mesi e anni.

Questi sono i contenuti che dovevano apparire in ogni pagina, e quindi parte del layout vero e proprio del sito. I contenuti che dovevano apparire nella prima pagina del sito erano invece:

- I tre articoli più recenti, evidenziati in qualche maniera.
- Altri cinque articoli recenti.
- Uno spazio per i banner pubblicitari
- Una sezione con gli articoli in evidenza, scelti dall'amministratore.
- Una sezione con gli articoli più commentati nell'ultima settimana.
- Una sezione con gli articoli più visti nell'ultima settimana.

Il design iniziale del front end era un layout a 3 colonne, con intestazione e piè di pagina, molto classico, con sfondo bianco e nero con colori dominanti, completamente fluido (che si adattava alla risoluzione dell'utente finale) e senza uso di tabelle. La colonna di sinistra doveva contenere:

- Il form per effettuare il login al sito.
- L'elenco delle categorie delle notizie.
- L'insieme dei tag delle notizie, con evidenziati i tag più frequenti.
- I link ai vari feed RSS.

3 Il progetto Capri 2.0

Nella colonna centrale ci sarebbe stato il contenuto vero e proprio della pagina, quindi principalmente gli articoli veri e propri. Nella prima pagina sarebbero stati messi i primi tre articoli più recenti e i successivi cinque con larghezza ridotta a circa 2/3 della colonna, lasciando uno spazio libero di 1/3 per i banner pubblicitari. Gli articoli dovevano avere una intestazione con il titolo e data dell'articolo, il corpo vero e proprio dell'articolo e un piede con l'autore dell'articolo, le categorie, i tag e un link per aggiungere e vedere gli articoli.

Nella colonna di destra ci sarebbe dovuto essere il resto del contenuto fisso per ogni pagina, ovvero:

- Un blocco selezionabile in qualche maniera che contenesse uno fra: gli articoli più letti dell'ultima settimana, gli articoli in evidenza come scelti dall'amministratore, gli articoli più commentati dell'ultima settimana.
- Il form per effettuare la ricerca degli articoli contenenti un certo termine.
- Una sezione con la descrizione breve del progetto Capri 2.0
- Un link alla pagina con l'elenco dei giornalisti del sito
- Un elenco di link esterni al sito
- Eventuali altre inserzioni pubblicitarie

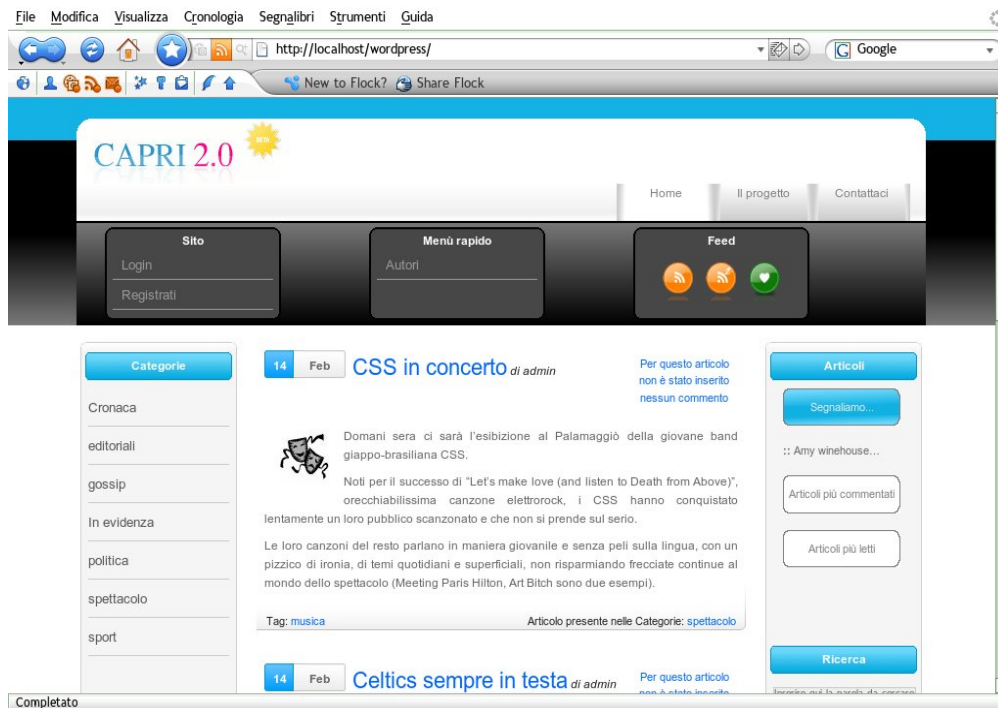
L'intestazione doveva contenere solo un logo grafico del progetto, che mi sarebbe stato inviato successivamente. Il piè di pagina doveva contenere il link per accedere alla pagina degli archivi degli articoli, e una nota di copyright.

Questo primo layout l'ho implementato con una grafica molto semplice. Da questo primo layout grafico, dopo una serie di iterazioni e prove, seguendo un modello di sviluppo iterativo, si è arrivati al layout grafico finale, visibile nella figura 3.1.

Le principali modifiche sono l'aggiunta di una zona in intestazione con i link di navigazione più importanti e i link ai feed RSS, l'uso di un accordion - un effetto che fa "scorrere"

3 Il progetto Capri 2.0

Figura 3.1: Layout finale del progetto Capri 2.0



i blocchi facendo apparire solo quello selezionato - per navigare la zona della colonna di destra della sezione con l'elenco selezionabile degli articoli più visti, evidenziati o più commentati, e lo spostamento nella zona dell'intestazione degli articoli dell'autore e del link per vedere o aggiungere commenti.

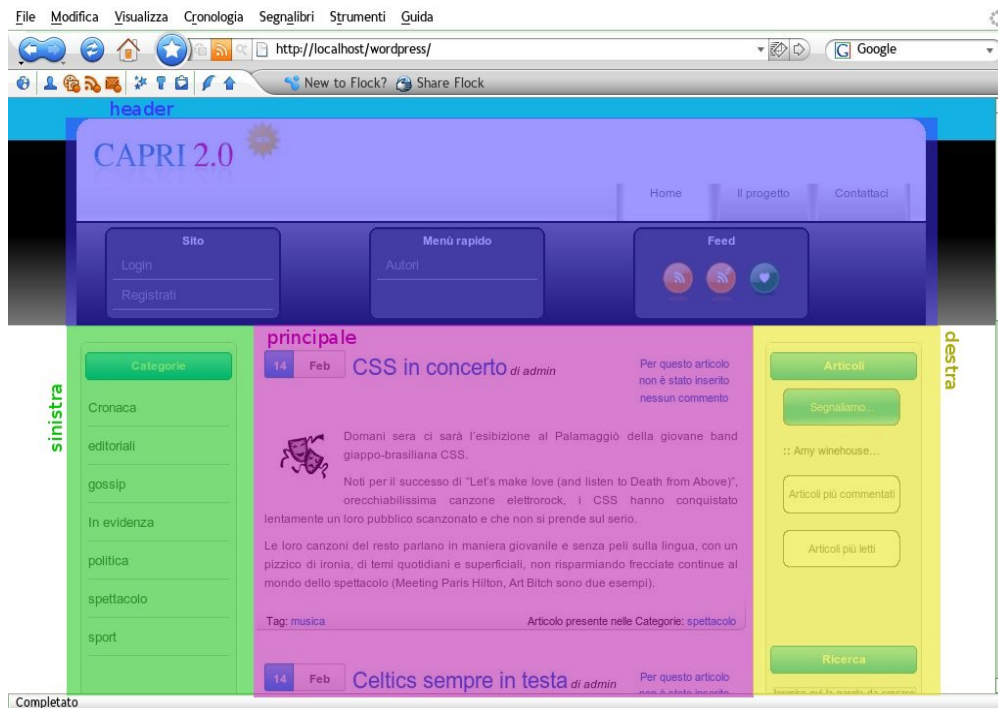
3.6.3 Implementazione del Theme per Capri 2.0 in CSS e XHTML:

Layout generale.

L'implementazione del front end, come detto precedentemente in 3.6.1, è stato fatto creando un nuovo template o tema o "Theme" di Wordpress. Il tema in particolare è stato implementato suddividendolo su vari file PHP, uno per zona principale del layout: sidebar.php, contenente la colonna di destra; header.php contenente l'intestazione e i menù superiori; footer.php contenente il piè di pagina; sinistra.php per la colonna di sinistra.

3 Il progetto Capri 2.0

Figura 3.2: Zone principali del layout



Questa suddivisione permette di gestire in maniera più semplice le modifiche alla grafica del sito, riducendo il codice da cambiare per cambiare una zona particolare della grafica. E' una tecnica che Wordpress non solo consiglia in [13], ma supporta attivamente offrendo delle funzioni che vadino a includere alcune zone predefinite (intestazione, pie di pagina, menu di navigazione laterale).

Il layout è stato implementato in maniera fluida e senza uso di tabelle, per migliorarne l'accessibilità, come da [5]. Per prima cosa si è divisa la pagina in cinque blocchi principali (usando DIV con nome): header, principale, sinistra, destra e footer. La figura 3.2 evidenzia le cinque zone suddette. Queste cinque zone sono state poi racchiuse in un blocco ausiliario a cui è stata assegnata la classe CSS wrapper.

Ora si è trattato di usare le capacità del CSS per allineare correttamente queste cinque zone. Per l'intestazione, è bastato imporre che la sua larghezza sia tutto lo schermo.

3 Il progetto Capri 2.0

Per le tre colonne, si è usato una tecnica, l’Holy Grail Revisited, a opera di Metthew Levine¹⁷, che prende però spunto da tecniche di altri autori. Questa tecnica permette di avere un layout a tre colonne fluido con le due colonne laterali a dimensione sia fissa sia fluida. Nel mio caso ho usato dimensioni fisse (200px). La tecnica usa un blocco addizionale, che aggiungiamo e di nome “contenuto”, che contiene i blocchi per le tre colonne. Questo blocco crea tramite la proprietà CSS padding lo spazio per le due colonne a sinistra e destra. Poi la colonna centrale si mette al 100% dello spazio, prendendosi tutta la parte centrale (escluso lo spazio creato precedentemente con i padding). La colonna sinistra sarà a dimensione fissa, con proprietà float a sinistra, e fissa la sua posizione usando combinatamente la proprietà “left” per fissarne il bordo sinistro, e poi usando un margine sinistro negativo, che “tira” la colonna nella giusta posizione, andando a invadere lo spazio creato con il padding a sinistra. La colonna destra usa analogamente un margine destro negativo per essere posizionata correttamente, invadendo lo spazio creato con il padding destro.

A questa tecnica poi sono aggiunti alcuni codici addizionali¹⁸ per risolvere dei problemi dovuti al parsing scorretto delle proprietà CSS da parte dei browser di casa Microsoft, Internet Explorer in versione 6 e 7, notoriamente deboli come implementazione del CSS¹⁹.

3.6.4 Implementazione dell’intestazione di Capri 2.0

Come si vede, l’intestazione del front-end finale ha una divisione grafica in tre zone di colori diversi. Per implementare questo effetto è stata usata una texture grafica applicata al body della pagina da CSS. Questa texture è una semplice immagine di larga 1 pixel

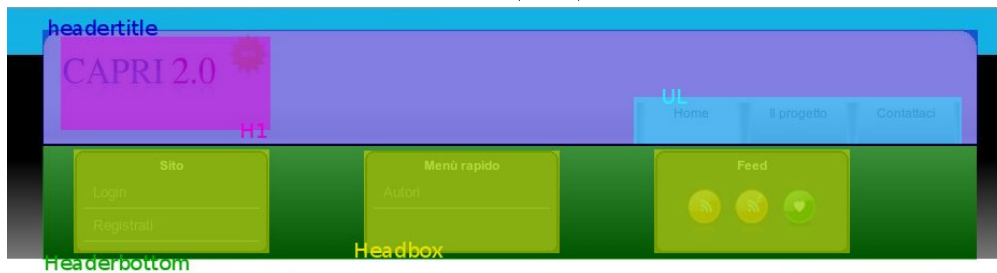
¹⁷vedi A list apart, “*In search of the holy grail*”, <http://www.alistapart.com/articles/holygrail>.

¹⁸a opera di Gerd Riesselmann, vedi “*The holy grail CSS fix for IE7*”, all’indirizzo <http://www.gerd-riesselmann.net/development/the-holy-grail-css-layout-fix-for-ie7>.

¹⁹La compatibilità ufficiale è reperibile all’indirizzo [http://msdn.microsoft.com/en-us/library/cc351024\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/cc351024(VS.85).aspx), ma i risultati della compatibilità misurati con i test che sono lo standard officioso di riferimento, Acid2 e Acid3, reperibili su <http://www.webstandards.org>, sono visibili ad esempio in SciActive, “*Acid Test Results on Popular Browsers*” all’indirizzo http://sciactive.com/main/index.php?option=com_content&task=view&id=133&Itemid=1, dove si vede che sia IE6 che IE7 hanno una compatibilità bassissima con lo standard CSS2 o 3.

3 Il progetto Capri 2.0

Figura 3.3: Divisione in blocchi (DIV) dell'intestazione del sito



e lunga quanto la lunghezza della zona header, che fisseremo nel CSS, e che crea le due zone colorate di nero e blu. Il body poi è impostato con lo sfondo bianco, che riempie il resto della pagina.

Successivamente, abbiamo suddiviso il blocco header in un insieme di sottoblocchi, come mostrato nella figura 3.3. I due sottoblocchi principali sono headertitle, che contiene un blocco, fatto usando semplicemente il tag xhtml H1, per il logo a sinistra, e un'altro blocco contenente i tre link a destra, implementato usando una lista non numerata (ul) con una apposita classe CSS. L'altro sottoblocco principale è headerbottom, che contiene tre sottoblocchi headbox, che sono i due blocchi con due link a testa, e il blocco con i link ai feed RSS.

Per implementare i bordi arrotondati dell'intestazione, abbiamo usato una tecnica CSS di nome "sliding doors"²⁰. In pratica, usando un blocco ausiliario che è contenuto dentro header, header2, abbiamo imposto un'immagine di sfondo ad entrambi. Lo sfondo di header è un'immagine lunghissima (3000 pixel) che rappresenta il bordo tondo a destra e lo sfondo dell'intestazione, con un blocco bianco in alto, una riga nera e un blocco sfumato grigio scuro sotto. Lo sfondo di header2 è invece l'immagine del bordo sinistro. Imponendo la posizione dei due sfondi, e sfruttando il fatto che lo sfondo di un blocco interno sovrascrive quello di un blocco esterno (proprietà di cascading dei CSS, appunto), possiamo creare uno sfondo grafico che si adatti automaticamente alla larghezza dello

²⁰vedi Douglas Bowman, "Sliding Doors of CSS", su "A list apart", all'indirizzo <http://www.alistapart.com/articles/slidingdoors/>

schermo (fino a un massimo di 3000 pixel di larghezza) senza dover usare altro che il CSS.

Per formattare i tre link in alto a destra, abbiamo imposto alla lista che i suoi elementi siano rappresentati come blocchi “galleggianti” a sinistra, usando la proprietà float del CSS. Così facendo, si ottenengono quindi tre blocchi messi uno dopo l’altro orizzontalmente. Il resto è normale uso delle proprietà del CSS per aggiungere uno sfondo e settare i colori. Inoltre abbiamo imposto una larghezza fissa dei tre elementi, per usare una sola immagine come sfondo, semplificando il codice CSS.

3.6.5 Implementazione delle colonne laterali

Passando alle due colonne laterali, queste contengono ognuno un sottoblocco che serve a creare il riquadro grafico contenente il loro contenuto. Questo riquadro è stato suddiviso in due parti, il bordo superiore e il contenuto vero e proprio.

Questo ha permesso di usare lo sfondo del blocco del riquadro vero e proprio per il bordo in basso del riquadro, lo sfondo sul sottoblocco superiore per il bordo superiore del riquadro e lo sfondo del sottoblocco corpo, ripetuto verticalmente per i due bordi destra e sinistra. Usando poi dei padding appositi, si è fatto in maniera che queste tre immagini si collegassero perfettamente creando il riquadro grafico.

Le intestazioni blu dei riquadri laterali sono immagini di sfondo, a larghezza fissa (visto che la colonna nel suo complesso ha già una larghezza fissa), e a lunghezza fissa (ragionevolmente grande da contenere il testo, la cui dimensione del carattere è stata fissata).

Nella colonna destra si trovano tre parti che visualizzano uno fra gli articoli in evidenza, gli articoli più letti o quelli più commentati, con l’utente che sceglie cliccando su una delle tre intestazioni quale visualizzare. Poiché questo compito può essere risolto completamente lato client, senza che il server debba spedire altre informazioni, è stato scelto di

3 Il progetto Capri 2.0

risolverlo usando uno script javascript che crei anche un effetto grafico che desse un minimo di “dinamicità” alla pagina. Lo script javascript manipola la pagina usando DOM, aumentando o riducendo la lunghezza dei sottoblocchi per dare l’effetto di scorrimento degli stessi.

In inizializzazione, lo script aggiunge delle chiamate a una funzione, che poi definisce, ai vari sottoblocchi che abbiano la classe CSS che termina per “-header”. Questa funzione controlla se il sottoblocco dal quale è stato attivato non sia già quello attivo correntemente, il cui ID DOM è contenuto in una variabile globale, altrimenti va a renderlo visibile (usando la proprietà display del CSS collegato, manipolandolo tramite la DOM) e prepara l’effetto di scorrimento settando a zero la sua lunghezza, e settando una variabile globale che indica qual’è il blocco da attivare settandola all’ID DOM del blocco che abbia come classe CSS quella del blocco dal quale ha ricevuto l’attivazione tranne che al posto di finire in “-header” finisca in “-content”. Poi va a settare un richiamo di un’altra funzione che viene definita nello script, temporizzata dopo un certo tempo, che è passato come parametro allo script, tramite l’oggetto document.timeout.

Quest’ultima funzione non fa altro che diminuire la lunghezza del blocco attualmente attivo e aumentare la lunghezza del blocco da attivare, manipolando le proprietà CSS dei due blocchi tramite la DOM, e se il blocco attuale non è andato a 0 come lunghezza setta un’altro richiamo di se stessa dopo lo stesso tempo. Altrimenti, nasconde settando la proprietà display CSS del blocco attualmente attivo a “none” e setta come blocco attualmente attivo quello che era da attivare. Lo script inizializza i blocchi con classe “-content” assegnandogli la proprietà CSS “display” come pari a “none”, nascondendoli, tranne il primo che trova, che è settato come il blocco attualmente attivo.

Lo script inoltre fa sì che il blocco attivo sia anche l’unico che abbia come classe CSS una classe extra passata allo script come parametro. In definitiva i tre parametri che riceve lo script sono l’identificativo CSS del blocco contenente tutti i sottoblocchi che lo script manipola, il tempo che deve essere aspettato fra una chiamata e l’altra della funzione

che rende l'effetto, regolandone la velocità, e una classe CSS da assegnare extra al blocco corrente.

Questo script era inizialmente opera di Dezinerfolio²¹, ma è stato modificato pesantemente da me per adattarlo alla struttura del sito, e cambiandone come è stato effettuato lo scorrimento (lo script originale generava problemi con l'attuale implementazione di javascript di Firefox).

3.6.6 Implementazione della zona con il contenuto specifico della pagina

Il contenuto vero e proprio delle pagine sarà posizionato quindi in una zona centrale, contenuto che varia a seconda della pagina.

Per la pagina iniziale, esso deve contenere una serie di articoli. In particolare i primi tre articoli occupano tutta la colonna, e i successivi cinque sono occupano solo due terzi della colonna, lasciando uno spazio libero alla sinistra per le inserzioni pubblicitarie. Questo è stato fatto semplicemente usando le proprietà CSS che controllano la larghezza e il margine sinistro dei blocchi.

Gli articoli sono contenuti in dei blocchi (DIV) di classe "news", suddivisi in tre sottoblocchi: l'intestazione (classe newsheader), il corpo dell'articolo vero e proprio (classe newsbody) e le informazioni in fondo agli articoli (classe newsfooter). La divisione è visibile in figura 3.4.

L'intestazione è poi suddivisa in tre sottoblocchi, uno per la data, uno per il titolo dell'articolo e uno per il link per aggiungere e leggere i commenti all'articolo. Questi tre blocchi sono stati posizionando usando la proprietà float, a sinistra la data e a destra i commenti, e usando dei margini opportuni per il blocco centrale del titolo. Il sottoblocco della data usa uno sfondo e una dimensione fissa, per rendere l'effetto grafico finale. Anche il blocco dei link ai commenti ha una dimensione fissa. Nel blocco del titolo ci

²¹vedi il loro sito, <http://www.dezinerfolio.com/>

Figura 3.4: Divisione in zone (DIV) degli articoli



sono due parti, il titolo vero e proprio e l'autore, che sono semplicemente formattati usando una dimensione del carattere e dei colori diversi.

Il sottoblocco del corpo dell'articolo specifica semplicemente la dimensione, colore e tipo del carattere, e altre proprietà del testo dell'articolo.

Il sottoblocco del piede dell'articolo usa uno sfondo che è un'immagine anche qui lunghissima (3000px) in maniera da adattarsi alla larghezza della colonna, che non è fissa. L'altezza complessiva è fissata, e a sua volta è divisa in due parti (sempre usando i DIV): la prima parte, a sinistra e larga il 50% della larghezza complessiva, contiene i tag dell'articolo, mentre la seconda parte, larghezza 50% e margine sinistro pari a 50%, contiene le categorie dell'articolo e il link al resto dell'articolo.

Per le gli articoli più stretti, è bastato semplicemente variare la larghezza del blocco contenente l'articolo, assegnando un'altra classe CSS che ha la larghezza settata al 70% e margine sinistro al 30%, in maniera da adattarsi alla larghezza del video dell'utente.

3 Il progetto Capri 2.0

Figura 3.5: Footer del sito



3.6.7 Implementazione del piede delle pagine

L'ultimo elemento del front-end con l'utente, presente in tutte le pagine, è il piede delle pagine, visibile in figura 3.5. L'implementazione è stata fatta usando un blocco (DIV) con settata larghezza al 100%, e con un'immagine di sfondo ripetuta lungo la larghezza (l'immagine infatti è larga 1px) e colore nero di sfondo. Inoltre ha settata la proprietà CSS "clear" in maniera da finire sicuramente sotto le tre colonne in ogni caso, infatti questa proprietà impone che l'elemento sia messo sotto ogni elemento con proprietà float.

3.6.8 Finalizzazione del front-end verso l'utente di Capri 2.0

La struttura precedentemente illustrata è stata usata per pressoché tutto il sito, adattandosi alla fine a ogni contenuto del sito. Infatti il progetto era incentrato principalmente sugli articoli, e si è adattato alla stessa struttura sia alla pagina presentante i risultati di una ricerca (sostituendo la parte indicante la categoria degli articoli con una indicazione di cosa si è ricercato) sia agli indici dell'archivio che all'indice degli articoli delle categorie che a quello degli articoli rispondenti a un certo tag (sostituendo la parte indicante i tag degli articoli con una indicazione del tag ricercato).

Unica pagina che richiedeva ancora lavoro era quella in cui si vedeva un articolo (risolto nella solita struttura) con però dei commenti, e avendo la possibilità di aggiungere un commento. Si vede un esempio di tale pagina in figura 3.6.

I commenti sono stati messi in una serie di blocchi (DIV) che a loro volta sono stati

Figura 3.6: Lista dei commenti di un'articolo

The image shows a screenshot of a blog article. At the top, there is a date and time indicator: '18 Gen'. The article title is 'Prova politica' by 'admin'. The main text of the article reads: 'Prova per vedere se le categorie funzionano, e se anche gli excerpt in prima pagina funzionano. Nel caso, vedo di aggiungerci il CSS e di vedere se tutto funziona come si deve. Sempre che trovi il tempo, visto che oggi proprio non vogliono farmi stare in pace. E con questo, direi che è tutto...'. Below the article text, it says 'Categorie: politica'. Underneath, there is a section titled 'Sono presenti 3 commenti a questo articolo:'. There are three comments, all from the user 'admin'. Each comment is followed by a timestamp: '14 Febbraio 2008 alle 12:46', '14 Febbraio 2008 alle 12:47', and '14 Febbraio 2008 alle 12:47'.

18 Gen **Prova politica** di admin

Prova per vedere se le categorie funzionano, e se anche gli excerpt in prima pagina funzionano. Nel caso, vedo di aggiungerci il CSS e di vedere se tutto funziona come si deve. Sempre che trovi il tempo, visto che oggi proprio non vogliono farmi stare in pace. E con questo, direi che è tutto...

Categorie: [politica](#)

Sono presenti 3 commenti a questo articolo:

admin scrive:
Commento 1 di prova
14 Febbraio 2008 alle 12:46

admin scrive:
Commento 2 di prova
14 Febbraio 2008 alle 12:47

admin scrive:
Commento 3 di prova
14 Febbraio 2008 alle 12:47

3 Il progetto Capri 2.0

Figura 3.7: Form per aggiungere un nuovo commento



admin scrive:
Commento 3 di prova
14 Febbraio 2008 alle 12:47

Connesso come [admin](#). [Esci](#)

divisi in tre parti: una intestazione, con la scritta in arancione e nero che indica l'autore del commento; una zona di corpo dell'articolo, con larghezza ridotta all'80% e margine sinistro del 20%, in maniera da adattarsi alla larghezza del display e contenente il corpo vero e proprio dell'articolo; e infine un piede, largo tutta la colonna centrale, e che usa come sfondo una immagine ripetuta lungo la lunghezza, e contenente la data e orario del commento. La formattazione di queste tre zone è stata effettuata come fatto con gli articoli, usando opportunamente le proprietà CSS padding e margin per allineare i contenuti e le zone in maniera che appaiono come un'unico blocco.

Il form per inserire i commenti, che è stato programmato partendo dal uno dei temi d'esempio dati in allegato con Wordpress, e che ho modificato solo nelle classi CSS e in un controllo addizionale, è visibile in figura 3.7.

3 Il progetto Capri 2.0

Il codice iniziale dei commenti, come preso dal tema di default di Wordpress è il seguente:

```
<?php if ( 'open' == $post-> comment_status ) : ?>
    <h3 id="respond">Scrivi un commento</h3>
    <?php if ( get_option('comment_registration')
        && !$user_ID ) : ?>
        <p>Per inviare un commento devi fare il
            <a href="<?php echo get_option('siteurl'); ?>
                /wp-login.php?redirect_to=
                <?php the_permalink(); ?>">log-in</a>.
        </p>
    <?php else : ?>
        <form action="<?php echo get_option('siteurl'); ?>
            /wp-comments-post.php"
            method="post"
            id="commentform">
        <?php if ( $user_ID ) : ?>
            <p>Sei collegato come <a href="<?php
                echo get_option('siteurl'); ?>/wp-admin/profile.php">
            <?php echo $user_identity; ?>
            </a>.
            <a href="<?php echo get_option('siteurl'); ?>/wp-login.php?action=logout"
                title="Esci da questo account">
                Logout &raquo;
            </a>
        </p>
    <?php else : ?>
        <p><input type="text" name="author" id="author">
```

3 Il progetto Capri 2.0

```
value="<?php echo $comment_author; ?>"
size="22" tabindex="1" />
<label for="author"><small>Nome
    <?php if ($req) (obbligatorio); ?>
</small></label>
</p>
<p>
    <input type="text" name="email" id="email"
        value="<?php echo $comment_author_email; ?>"
        size="22" tabindex="2" />
    <label for="email">
        <small>E-mail (non verr&agrave; pubblicata)
        <?php if ($req) (obbligatorio); ?>
        </small></label>
</p>
<p>
    <input type="text" name="url" id="url"
        value="<?php echo $comment_author_url; ?>"
        size="22" tabindex="3" />
    <label for="url"><small>Sito Web</small></label>
</p>
<?php endif; ?>
<p>
    <textarea name="comment" id="comment" cols="100%"
        rows="10" tabindex="4"></textarea>
</p>
<p>
```

3 Il progetto Capri 2.0

```
<input name="submit" type="submit" id="submit"
      tabindex="5" value="Invia commento" />
<input type="hidden" name="comment_post_ID"
      value="<?php echo $id; ?>" />
</p>
<?php do_action('comment_form', $post->ID); ?>
<?php endif; ?>
<?php endif; ?>
```

Analizzandolo, vediamo che viene fatto un controllo iniziale per vedere se l'articolo può essere commentato, andando a verificare lo stato della proprietà "comment_status" dell'oggetto "\$post", che è l'oggetto rappresentante l'articolo attuale. Se questa proprietà è settata su "open", allora viene visualizzato un form XHTML che deve passare alla pagina all'URL wp-comments-post.php nella root del sito (la funzione get_option('siteurl') non fa altro che ritornare l'indirizzo della radice del sito), i campi "comment" con il corpo del commento e i campi "author", "email" e "url" con il nome, e-mail ed eventuale sito dell'autore del commento. Inoltre deve essere chiamato nel form la funzione "do_action('comment_form', \$post-ID)" che non fa altro che aggiungere dei campi nascosti al form, di supporto al motore di Wordpress, e in particolare l'ID dell'articolo, ed eventualmente i tre campi dell'autore del commento se l'utente corrente è già stato autenticato da Wordpress, e poi richiamare eventuali plugin che si siano collegati all'azione di mostrare il form dei commenti, come spiegato già in precedenza.

Le modifiche che ho dovuto fare a questo codice è stato semplicemente di aggiungere un controllo extra iniziale, che decide se emettere il form per aggiungere un commento, aggiungendo come condizione extra che l'utente corrente fosse capace di modificare articoli, capacità che è assegnata nel nostro progetto solo ai giornalisti. Il codice del controllo quindi è diventato:

3 Il progetto Capri 2.0

```
<?php if ('open' == $post->comment_status) : ?>
    <?php if (get_option('comment_registration')
        && !$user_ID) : ?>
        <p class="nocomments">
            Per commentare questo post devi prima
            <a href="<?php echo(get_option('site_url')); ?>
                /wp-login.php?redirect_to=
                <?php the_permalink(); ?>">
                autenticarti
            </a>.
        </p>
    <?php else : ?>
        <?php if (!current_user_can('edit_posts')) : ?>
            <p class="nocomments">Il tuo livello non consente
                di aggiungere commenti.
            </p>
        <?php else : ?>
```

A questo codice segue poi il form vero e proprio, con gli stessi campi e struttura del codice estratto dal tema originale di Wordpress. Si noti come abbiamo controllato innanzitutto che l'utente corrente si sia già autenticato (indicato dalla presenza della variabile `$user_ID`) che possa poi modificare i post per assicurarsi che sia un giornalista, tramite la funzione `current_user_can`²².

Sistemati quindi le pagine con i commenti e la possibilità di inserirli da parte dell'utente, nessun'altra pagina ha presentato eccezioni particolari alla struttura base del sito, illustrata precedentemente.

²²questa funzione non è ufficialmente documentata nel Codex di Wordpress, ma suoi esempi d'uso sono ritrovabili ad esempio in <http://markjaquith.wordpress.com/2006/03/27/how-to-check-if-a-wordpress-user-is-an-administrator/>

3.7 Verifica dell'usabilità del front-end utente di Capri 2.0

Dopo aver messo questa prima versione del front-end verso l'utente, è stato fatto provare da un gruppo selezionato di beta tester. I test sono correntemente ancora sotto svolgimento.

Per verificarne l'usabilità, dopo una prima fase di uso libero in cui agli utenti saranno chieste solo eventuali funzionalità mancanti o da modificare, si procederà a una rigorosa analisi seguendo le tecniche in [15] per la valutazione dei siti Web.

Attualmente, al completamento del mio tirocinio, la ditta stava avviando la fase iniziale di beta testing.

3.8 Implementazione del front-end amministrativo di Capri 2.0

Oltre all'implementazione della parte di front-end verso l'utente, un'altra parte con cui gli utenti, anche se in questo caso solo gli utenti giornalisti e l'amministratore, interagiscono con il sito è la cosiddetta "parte amministrativa".

Questa è la parte, gestita da WordPress, con cui gli utenti giornalisti scrivono i loro articoli da aggiungere al sito, oppure controllano e modificano gli articoli che hanno scritti. Queste due funzionalità in particolare devono essere resi le più semplici possibili da effettuare, soprattutto da parte di utenti con poca esperienza precedente di interazione con un computer.

Questa parte è anche la parte con cui l'amministratore del sito interagisce per controllare gli utenti iscritti, per gestire quali articoli andranno in prima pagina, per gestire gli eventuali banner pubblicitari, per mandare le e-mail informative agli utenti del sito, e per effettuare tutte le configurazioni del sito. In questo caso è stato più chiesto di rendere

uniforme alcuni particolari rispetto al front-end utente, e di eliminare alcune funzionalità che sono state viste come superflue.

3.8.1 Funzionalità da implementare nella parte amministrativa di Capri 2.0

In particolare le richieste riguardanti questa parte che mi sono state chieste sono:

- Cambiare completamente la schermata per l'autenticazione nella parte amministrativa
- Semplificare la pagina iniziale della parte amministrativa, togliendo praticamente tutto tranne il riquadro con cui Wordpress ricorda le ultime azioni fatte nella parte amministrativa, e la parte di navigazione.
- Cambiare il piede della parte amministrativa per renderlo uniforme con quello della parte utente.
- Togliere i dati riguardanti gli instant messaging dai dati degli utenti memorizzati da Wordpress
- Proibire l'accesso alla parte di gestione dei commenti agli utenti che non siano l'amministratore
- Eliminare la possibilità di accedere al codice XHTML dell'articolo che si sta scrivendo nella sezione in cui si inseriscono nuovi articoli.
- Ridurre le azioni possibili dopo aver scritto un articolo nuovo alla sola pubblicazione o annullamento.
- Eliminare l'accesso alla parte in cui si può caricare un file sul sito agli utenti non amministratori.
- Ridisegnare la grafica della parte in cui si inseriscono le notizie, eliminando alcuni altri elementi

3 Il progetto Capri 2.0

- Ridurre al solo utente amministratore l'accesso alla parte del sito con cui si inseriscono nuove pagine statiche.

La maggioranza degli interventi richiesti alla fine ricadeva in una delle tre categorie seguenti:

1. Riduzione dell'accesso a certe funzionalità al solo utente amministratore. Questo è stato possibile semplicemente sfruttando opportunamente il meccanismo di permessi che Wordpress implementa, detti "Capabilities"²³, che regolano in maniera molto puntuale cosa è permesso agli utenti. In particolare, si è modificato il ruolo dei giornalisti spostandolo al ruolo predefinito di "Author", che è quello che più rispondeva a quello desiderato, permettendo solo l'inserimento di nuovi articoli e la modifica dei propri articoli. L'unica modifica da fare è stato proibire l'accesso alla capacità di inserire file, che è stata ottenuta modificando il codice di Wordpress.
2. Ridefinizione della grafica della parte amministrativa in alcune sue parti. Per effettuare questa operazione, la documentazione di Wordpress in [16] dice di usare un plug-in che implementi una funzione che aggiunga un richiamo a un foglio CSS nelle pagine amministrative, demandando poi al CSS la manipolazione della grafica del sito. Poiché tutte le nostre richieste di cambio erano fattibili usando il CSS, abbiamo seguito questa strada. L'unica modifica che ha richiesto la modifica del codice di Wordpress è stata quella che modifica il contenuto del piede delle pagine amministrative.
3. Eliminazione di elementi dalle pagine della sezione amministrativa. Questo è stato svolto, una volta agganciato un file CSS alle pagine amministrative come illustrato nel punto precedente, semplicemente identificando le giuste classi o riferimenti DOM, e usando la proprietà CSS "display" settata su "none", in maniera da non far

²³cfr. Wordpress Codex, "*Roles and Capabilities*", http://codex.wordpress.org/Roles_and_Capabilities.

visualizzare l'elemento. Questa scelta permette facilmente in futuro di riabilitare gli elementi, eliminando dal CSS aggiunto le righe desiderate.

Nella sezione successiva vedremo come questi tre punti possano essere implementati sfruttando il plug-in che già implementato precedentemente per aggiungere funzionalità specifiche richieste dal progetto Capri 2.0.

3.8.2 Implementazione del front-end amministrativo di Capri 2.0 usando il plug-in.

I tre punti illustrati precedentemente per essere implementati richiedevano, a parte qualche minore modifica del codice di Wordpress, semplicemente l'aggiunta di una nuova funzione al nostro plug-in per Wordpress, che aggiungesse il richiamo a un CSS opportuno per la sezione amministrativa. La funzione finale usata è la seguente:

```
function jrn_admin_css() {
    $url=get_settings('siteurl');
    echo('<link rel="stylesheet"
        type="text/css"
        href="' . $url . '/wp-content/plugins/capri20/admin.css"
        />');
    echo("\n");
    echo('<link rel="stylesheet"
        href="' . $url . '/wp-admin/css/upload.css"
        type="text/css" />');
    echo("\n");
}
```

Che come si vede non fa altro che emettere il codice per il giusto richiamo al CSS, come da standard XHTML illustrati in [17]. Questa funzione poi è semplicemente associata al-

3 Il progetto Capri 2.0

l'action "wp_admin_css", in maniera da essere richiamata al punto giusto di ogni pagina della sezione amministrativa.

Tutto il resto è stato implementato semplicemente scrivendo il giusto codice CSS nel file "admin.css", che per la maggioranza non fa altro che impostare a "display: none", ovvero nascondere, alcune parti dell'interfaccia amministrativa di base di Wordpress, tramite identificatori CSS.

4 Spunti dal progetto Capri 2.0

In questo capitolo vedremo alcuni possibili scenari futuri che prendono spunto dal progetto Capri 2.0, incentrati principalmente sia sull'impatto del CMS nell'ambito del web, che sull'impatto che il web stesso sta avendo sul giornalismo, proponendogli nuovi modi di intendersi e nuove realtà con cui confrontarsi.

Infine vedremo alcune organizzazioni di contenuti e dati che stanno emergendo dalla realtà del web, e che possono trovare uso anche in ambiti che non siano strettamente quello web, tracciando alcuni scenari ipotetici.

4.1 La rivoluzione “Web 2.0” nell'ambito giornalistico

4.1.1 Cos'è il “Web2.0”?

Il termine “Web 2.0”, come detto in [18] e [19], nasce ufficialmente nel 2004, a opera congiunta di Tim O'Reilly e Dale Dougherty, con la prima “Web 2.0 conference”, una conferenza creata e sostenuta dalla O'Reilly Group e MediaLive International, ma aperta a tutti gli operatori del settore internet, per rispondere alla crisi delle ditte operanti sul web iniziata nell'autunno 2001 con la cosiddetta “dotcom bubble burst”¹, un crollo in borsa delle azioni collegate alle ditte operanti nel campo del web, che ha come cause probabilmente un eccesso di fiducia degli investitori, guidati da motti come “New Economy”

¹cfr. Investopedia, “*Crashes: The dotcom crash*”, <http://www.investopedia.com/features/crashes/crashes8.asp>

4 Spunti dal progetto Capri 2.0

e che al primo segno negativo, probabilmente² il giudizio finale del processo aperto dal governo degli Stati Uniti contro Microsoft³, sono fuggiti via, mandando in crisi il mercato azionario e togliendo risorse finanziarie alle ditte nel settore del web.

Il termine “Web 2.0” ha tante definizioni. Tutti concordano però sul fatto che il “Web 2.0” non indica *qualcosa di preciso*, ma piuttosto un *insieme di concetti ben definito*, che caratterizzano le ditte che sono sopravvissute al “burst” citato prima, e che quindi si sono rivelate vincenti economicamente.

Un riassunto di questi concetti è offerta da vari autori, quali Tim O’Reilly in [18] e Paul Graham in [19]. Questi concetti sono criticati ad esempio da da Tim Bernes-Lee in una intervista[20], in quanto non innovativi, il che renderebbe quindi il termine stesso “Web 2.0” privo di vero significato.

I concetti però associati in maniera unanime al termine “Web 2.0” sono i seguenti:

1. Il web come piattaforma per offrire servizi agli utenti, non come bene da vendere. Quindi anche un modo diverso di intendere il software, come servizio da manutere ed evolvere costantemente, giorno per giorno.
2. Il “network effect”: il valore di un servizio deve aumentare con il numero di utenti che lo usano, e i servizi stessi devono sfruttare la conoscenza collettiva degli utenti.
3. Gli utenti come risorsa attiva, come vero valore fondamentale e quindi da rispettare e a cui dare potere. Devono essere il fondamento della ditta, e non bisogna mai perdere contatto con essi.
4. L’economia di tipo “long tail”, come descritta in [21]: si guadagna di più a vendere un varietà maggiore di prodotti, che individualmente vendono meno, ma che inte-

²cfr. Wikipedia, “*Dot-com bubble*”, http://en.wikipedia.org/wiki/Dot-com_bubble

³cfr. U.S.A. Departement of Justice vs Microsoft, 87 F.Supp. 2d30 e la pagina web dei casi contro la Microsoft da parte del dipartimento di giustizia americano, http://www.usdoj.gov/atr/cases/ms_index.htm

ressano a una fascia maggiore di utenti, che una varietà minore di beni che vendono individualmente tanto ma interessano solo una fascia minore di utenti.

Inoltre legato al termine “Web 2.0” si ritrovano una serie di altri concetti e tecniche di vario tipo, quali la decentralizzazione del potere, i sistemi di tipo emergente in cui i dati e i contenuti stessi sono creati dagli utenti, la “remixability” ovvero la possibilità degli utenti di scegliere quali dati visualizzare magari prendendoli da fonti diverse, le interfacce web basate su AJAX, i linguaggi di scripting usati nelle webapp come PHP, i concetti di “democrazia digitale” in cui gli utenti hanno potere, le strutture dati a gerarchia piatta, e molte altre.

4.1.2 Il giornalismo “dal basso”: I blog e la blogosfera

Essendo “Capri 2.0” molto legato al giornalismo in ambito Web 2.0, fin della sua ideazione, una esposizione e valutazione delle novità portate dal “Web 2.0” nel campo giornalistico è necessaria. In tale ambito, l’entità legata al “Web 2.0” che ha avuto maggiore impatto, sono sicuramente i blog e il concetto di “blogosfera”, inteso come l’insieme di persone che usano o leggono un blog. I blog attualmente sono la maggior fonte di informazioni e news disponibile sul web.

I blog, sotto il profilo tecnico, sono strettamente legati ai CMS. Infatti si possono vedere come CMS specializzati in contenuti testuali, i post appunto, e con certe particolari caratteristiche (tipo l’ordine cronologico solitamente inverso, l’accessibilità pubblica, la possibilità di aggiungere commenti ai post). Alcuni dei CMS più famosi, in effetti, nascono come piattaforma per blog, come ad esempio lo stesso Wordpress che ho usato per il progetto Capri 2.0.

I blog nascono storicamente come semplici diari on-line di una persona, che sceglie di rendere pubblici dei suoi pensieri su quello che fa e vede, in maniera saltuaria, detti “post”. Nascono quindi per uno scopo a metà fra il personale e il pubblico, e legati strettamente a

un utente specifico, che è colui che scrive il blog, e che è chiamato “blogger”. Uno dei primi blog della storia, secondo [22], è quello di Justin Hall, che successivamente lo trasformerà in un sito di link verso siti interessanti. Il concetto di “diffondere pubblicamente pensieri del giorno personali” comunque non è nuovo o legato al web, ed esempi di questo sono ad esempio il newsgroup mod.ber, forse la prima vera comunità personale, e il diario personale di John Carmak, diffuso tramite il protocollo finger. La differenza con i blog sta però in due fattori: i blog sono esposti sul web, e sono aperti agli utenti, che possono arricchirli usualmente potendo lasciare commenti ai post.

L’idea di un sito che espone articoli giornalieri con la possibilità agli utenti di commentarli si associa bene all’uso giornalistico, e in questo uno dei primi e più famosi siti a sfruttare il formato dei blog per uso giornalistico è Slashdot⁴, la cui storia è illustrata in [23], che però non è il primo in assoluto. Il primo vero giornale web in formato blog è probabilmente “Bluesnews”, ancora attivo e che ha articoli risalenti a luglio del 1996⁵.

La vera novità portata dai blog è il fatto che le notizie vengano scritte dagli utenti stessi, siano essi il “blogger” che possiede il blog o uno degli utenti abilitati a scrivere nuovi post. E che ci sia un valore aggiunto alla pura notizia, che è data dai commenti degli utenti, che aggiungono un valore spesso maggiore di quello della notizia stessa. In questo i blog sono un vero prodotto dell’idea chiave del “Web 2.0” di avere gli utenti come parte attiva del valore del prodotto.

Quando poi sono nati dei servizi che offrivano la possibilità a chiunque di avere un proprio blog, da usare come si vuole e visibile pubblicamente, come ad esempio Blogger⁶ o Splinder⁷, la diffusione dei blog è aumentata a dismisura. Attualmente il numero totale

⁴<http://www.slashdot.org>.

⁵Reperibile su <http://www.bluesnews.com>, ad esempio vedi l’archivio di Luglio 1996, <http://www.bluesnews.com/archives/july96.html>.

⁶<http://www.blogger.com>, uno dei più famosi servizi di bloggin, ovvero che permette a un utente di avere un blog gratuitamente

⁷<http://www.splinder.com>, una delle più grosse comunità italiane di blog.

4 Spunti dal progetto Capri 2.0

di blog esistenti è difficile da stimare, ma sono sicuramente più di 100 milioni⁸. I blog italiani sono molto di meno, circa tre milioni⁹, pari a circa il 3%.

L'altra novità dei blog è il fatto che ci siano collegamenti fra un blog e l'altro, spesso come una serie di links ad altri blog ritenuti interessanti, cosicché i blog non siano elementi isolati fra loro, ma formino una unica grande rete interconnessa tramite link, detta "blogosfera". Questo permette a una notizia postata magari su un blog personale poco conosciuto, di raggiungere altri blog, fino magari ad approdare a uno dei blog giornalistici più importanti, il tutto in pochissimo tempo.

Questa rapidità di diffusione, legata ovviamente al fatto di essere strumenti web interconnessi, e il fatto di poter anche postare spesso anonimamente o comunque dietro un nickname, magari aprendo al volo un nuovo blog solo per proteggere la propria identità, permette spesso anche di far filtrare notizie che altrimenti sarebbero censurate da regimi autoritari. Questa resistenza alla censura fa della blogosfera, nel suo intero, un mezzo interessante per uso giornalistico.

Svantaggi di questo approccio giornalistico sono che la credibilità della notizia è legata solo al nome di chi l'ha scritta sul proprio blog, senza poter spesso fare riscontri oggettivi, con tutte le conseguenze del caso. Il giudicare l'attendibilità delle notizie, nella blogosfera, è lasciato all'utente che legge la notizia stessa.

Legato e intrinseco al fenomeno dei blog è quello del "giornalismo dal basso", in cui sono gli utenti stessi, attivamente, a selezionare quali notizie diffondere, e a riportare attivamente critiche, analisi e semplici notizie. E' un giornalismo senza giornalisti, che parte dal basso, dai lettori che diventano anche scrittori e redattori. Un giornalismo democratico, che anche se non sempre ha avuto successo (come nel caso del Barbieri

⁸Il solo Technorati, un motore di ricerca specializzato in blog, ne conta circa 112 milioni nel dicembre 2007, come riportato su <http://technorati.com/about/>

⁹dati forniti da uno studio dell'università di Urbino, e riportati su Pandemia all'indirizzo http://pandemia.info/2008/05/18/i_social_media_in_italia.html

della Sera¹⁰, ad esempio), ha delle potenzialità evidenti, soprattutto per la possibilità di trovare notizie che i media tradizionali non pubblicherebbero per mancanza di interesse o per scelte editoriali. Questo lo rende ideale ad esempio per avere una fonte di notizie per nicchie molto specializzate.

Altro ambito in cui i blog, con la loro struttura peculiare, hanno influenza, soprattutto in tempi recenti, è quello politico. La struttura stessa dei blog permette di avere fonti con varie affinità politiche, permettendo magari all'utente finale di avere la possibilità di leggere una stessa notizia attraverso vari punti di vista, in quanto il blog rimane strettamente personale e legato alla soggettività della persona che lo gestisce. Avere un'influenza anche politica è uno degli scopi anche del progetto "Capri 2.0".

Inoltre i blog e la blogsfera sono il mezzo ideale per ospitare movimenti o idee politiche che non troverebbero, per un motivo o per un'altro, una visibilità pubblica. Essendo il numero di blogger ormai elevato o comunque rilevante a livello politico, diventa importante che la politica dialoghi anche con loro sullo stesso piano. E' interessante notare che negli ultimi anni infatti l'uso dei blog si sia diffuso anche come mezzo di propaganda politica, con l'affacciarsi di blog più o meno ufficiali degli stessi candidati politici¹¹, che cercano un dialogo con i blogger sul loro stesso piano. Famoso in questo senso il caso scoppiato di recente in Italia riguardante Beppe Grillo, che sta organizzando un vero e proprio movimento a connotazione politica tramite il suo blog, con tanto di proposte di leggi¹².

4.1.3 I social networks - i successori dei blog?

Il termine "social network" si intende solitamente un sito web che è centrato sulla comunità dei suoi utenti, e offre servizi che facilitano la costruzione di relazioni fra di essi, di

¹⁰blog giornalistico italiano, che per un periodo ha sospeso le pubblicazioni perché gli articoli e i commenti che arrivavano non erano di qualità sufficiente. Ha riaperto dopo circa un anno, ad aprile 2008.

¹¹siano essi i candidati come Obama e Clinton alla presidenza americana, o quelli di Antonio DiPietro e di altri politici italiani

¹²vedi il blog di Grillo, <http://www.beppegrillo.it>, e le proposte li ritrovabili.

4 Spunti dal progetto Capri 2.0

vario tipo. Non è però l'unica definizione, visto che lo stesso termine è stato utilizzato per siti più vicini ai CMS tradizionali, in cui però tutto il contenuto sia fornito dagli utenti del sito¹³ spesso in maniera che il contenuto sia pubblico. La creazione di una comunità di utenti è fra gli obiettivi del progetto “Capri 2.0”, che è possibile si evolva poi in un social-network nel senso di quest'ultima definizione nel futuro.

I “social network” sono una estremizzazione di uno dei principi del “Web 2.0”, quello che dice che gli utenti sono il vero cuore dell'azienda. In questo caso, gli utenti sono il vero bene su cui il sistema lavora, e sono loro l'unica risorsa della ditta. Infatti i “social networks” di successo sono quelli che hanno più utenti, ed è solo quello l'unico metro di paragone. In questo entrano effetti di rete, per cui la difficoltà è raggiungere una massa critica di utenti.

I social networks hanno avuto una forte diffusione negli ultimi anni, anche in Italia. Un sito tipico di social network permette di iscriversi gratuitamente, di generare un pagina web personale detta “profilo”, e a seconda dei casi di aggiungere dei contenuti di vario tipo. Alcuni social networks offrono ad esempio anche funzionalità di blog.

Una caratteristica dei social network è la possibilità di visualizzare pubblicamente i contatti con cui si è in rapporto sul proprio profilo, permettendo quindi di costruire reti di rapporti fra gli utenti, vedendo gli amici degli amici e così via.

La maggioranza delle critiche nei confronti dei social network è sulle questioni della privacy e della sicurezza degli utenti, che devono avere la possibilità di scegliere quali dati mostrare, e devono esserci meccanismi per assicurare la sicurezza degli eventuali minori presenti fra gli utenti.

Sotto il profilo tecnico, i social-network sono tipi speciali di CMS, che devono gestire una quantità notevole di utenti ognuno però con un quantitativo di contenuto utente.

Quello che li differenzia è che l'accento è posto sulla gestione degli utenti, e dei rapporti

¹³Alcuni casi di siti definiti in questa maniera “social networks” sono del.icio.us, che raccoglie indirizzi di siti, e Flickr, che raccoglie immagini digitali.

fra di loro. Attualmente la maggioranza usa dei sistemi scritti ad-hoc, e sono abbastanza rari¹⁴ CMS esistenti orientati a tale uso. In questo probabilmente non si è ancora vista una vera maturazione tecnologica che porti a una serie di piattaforme diffuse e standard, anche se probabilmente basterebbe aggiungere alcune funzionalità alle piattaforme di CMS esistenti.

Da notare che “Capri 2.0” si prefigura uno degli obiettivi dei social networks, ovvero di costruire una comunità sottostante di utenti con legami fra di loro. E’ un obiettivo secondario, ma rilevante del progetto.

4.2 Organizzazione dei contenuti con gerarchie piatte e possibili evoluzioni

Uno degli aspetti più interessanti emergenti dal movimento “web 2.0” e in particolare dai CMS è una nuova modalità di organizzazione dei dati che sta lentamente diffondendosi, ovvero le gerarchie piatte. In questa parte approfondiremo cosa è una gerarchia piatta, che origini ha avuto, e che possibilità future permette.

4.2.1 Gerarchie piatte - un nuovo modo di organizzare i dati

Le gerarchie piatte sono un nuovo modo di organizzare i dati che sta emergendo di recente sul web. Questo metodo prevede che a ogni contenuto o dato siano associati più etichette o “tag”, metadati che ne qualificano le sue qualità. L’accesso poi ai dati avviene passando per i tag, in maniera inclusiva o esclusiva, ovvero cercando i tag che contengano uno fra un insieme di tag o solo i dati che siano etichettati con tutti i tag di un insieme.

La differenza sostanziale rispetto alle gerarchie dati tradizionali sono principalmente due:

¹⁴anche ci sono adattamenti di CMS come Drupal o Joomla e qualche CMS apposito come iSocial

4 Spunti dal progetto Capri 2.0

- Ogni dato non è associato esclusivamente a una categoria, ma a un insieme di tag, non rendendo esclusiva la relazione fra dati e classificazione.
- Mentre le categorie hanno relazioni di tipo gerarchico fra loro, i tag non hanno alcuna relazione fra loro.

Questo modo di classificare i dati nasce sul web, ed è lì che ancora è maggiore la sua diffusione. Ad esempio una parte rilevante dei CMS permette di classificare anche in una gerarchia piatta i contenuti, e altri che non hanno questa capacità nativamente spesso hanno plug-in o estensioni che lo permettono. Nel progetto Capri 2.0 abbiamo ad esempio usato Wordpress anche perché aveva questa capacità nativamente, essendo la creazione di una gerarchia piatta degli articoli una delle richieste iniziali.

Il perché sia un'organizzazione che si sta diffondendo è ben spiegato da John Beeler in [24], a riguardo di perché Gmail è diventato rapidamente il sistema di e-mail più diffuso:

“But flat hierarchies are actually better suited to PCs, and gmail and delicious demonstrates that. The power of flat hierarchy, and gmail, is that if John sends you an email, about lasers, that contains an attachment, it meets three criteria. So in outlook, or yahoo, how do you file this? If you want to read all the emails you’ve received about lasers, you have to either copy that email from john into two different folders. Practically it can only be under one, since I’m not sure most email systems even let you copy emails into other folders. If you want to see all the emails you’ve received that were about lasers, you really can’t. You have to use some slow and cubersome search function that never seems to work, or just remember where you put the email. If you want to see all the emails you’ve received that have attachments, it is just as cubersome because you probably filed that email from me in “From John” if at all.

4 Spunti dal progetto Capri 2.0

Most likely, you've resigned that it's pointless to collect emails because you can't find them later anyway you deleted my precious email about lasers.

Gmail, and its flat hierarchy, eliminates that problem. That email is now designated with all three of those labels. It makes finding emails much more efficient than Outlook or Yahoo. You can click on the About Lasers label, and up comes my email as well as all the other emails you received from people about lasers. But what if you want to look at emails just from me, that are not necessarily about lasers (which is unlikely). You can do that too in gmail.”

Questo modo di classificare dati è stato storicamente iniziato ad usare sui blog, per classificare i post, ma si è diffuso anche ad altri ambiti, quali ad esempio la classificazione delle e-mail¹⁵, delle immagini¹⁶, delle canzoni¹⁷, dei collegamenti web preferiti¹⁸, e in molti altri ambiti.

Gli svantaggi delle gerarchie piatte come metodo di classificazione dei dati è legato alla delicatezza nella scelta dei tag: una scelta non corretta potrebbe portare a ripetizioni o ai cosiddetti “one shot tag”, ovvero a tag a cui è associato un solo contenuto, e che non portano quindi una vera classificazione utile dei dati. L'altro svantaggio è una maggiore complessità della struttura sottostante che prevede una associazione multi-a-molti fra i dati e i tag, rispetto alla convenzionale multi-a-uno delle strutture gerarchiche.

Le gerarchie piatte stanno iniziando a uscire dal web e essere usate anche in applicazioni classiche, ad esempio per classificare canzoni nei lettori multimediali¹⁹ secondo etichette

¹⁵Vedi ad esempio il servizio di e-mail di Google, Gmail, il primo a usare non una classificazione gerarchica (a cartelle) delle e-mail, ma una gerarchia piatta basata su etichette (label).

¹⁶Vedi l'IPTC/IIM su <http://www.iptc.org/IIM/> uno standard per classificare immagini che prevede anche una gerarchia piatta basata su tag.

¹⁷Vedi ad esempio LastFM, un sito che si propone di classificare tutta la musica tramite anche un sistema a gerarchia piatta basata su tag, in particolare una folksonomy come vedremo successivamente.

¹⁸Ad esempio la classificazione usata dalle ultime versioni di Firefox, e che prevede anche una gerarchia piatta basata su tag.

¹⁹Vedi la modifica per foobar2k reperibile su <http://www.kilobitspersecond.com/2005/05/01/a-flat-hierarchy-for-subjective-mp3-tags/>

magari collegate a cose come l'umore che la canzone ispira, e con la possibilità di creare playlist indicando uno o più tag; nella classificazione delle immagini²⁰, in maniera da ricercare rapidamente le immagini che rispondano a certi tag; o addirittura un file system intero²¹ che classifichi i file usando una gerarchia piatta e non la tradizionale gerarchia di cartelle e file, potendo quindi classificare i file stessi con tag.

4.2.2 TaggedNotes: un semplice esempio di uso delle gerarchie piatte in ambito personale

Un semplice esempio di uso delle gerarchie piatte è una applicazione web che ho scritto, TaggedNotes²², che permette di gestire un semplice insieme di note, attualmente solo testuali, tramite una gerarchia piatta. E' nata per un uso personale, e successivamente gli ho aggiunto alcune caratteristiche per un uso intra/inter-net.

Le note in particolare sono classificate su un sistema misto gerarchico/piatto, in quanto esiste una serie di "blocchi" che contengono al loro interno una serie di note e di tag organizzati su una gerarchia piatta. Gli utenti possono liberamente tramite una semplice interfaccia web, creare e cancellare i blocchi, e gestire la gerarchia piatta delle note aggiungendo, cancellando e modificando sia le note che i tag.

Al momento dell'inserimento di una nota, l'utente può scegliere a quali tag associarla, oltre a inserirne il titolo e il corpo. L'accesso poi alle note avviene indicando un insieme di tag, e l'applicazione risponde elencando quali note hanno associate quell'insieme di tag. E' presente anche una semplice ricerca agente sul titolo e sul corpo delle note.

Questa semplice applicazione permette di evidenziare come, appoggiandosi su una base dati relazionale, non sia particolarmente complicato implementare una gerarchia piatta, basta aggiungere infatti alla tabella che memorizza gli oggetti che si vogliono organizzare

²⁰cfr. IPTC/IIM, su <http://www.iptc.org/IIM/>, e i programmi che lo usano.

²¹vedi il progetto Tagsistant un "A semantic filesystem for Linux kernels" su <http://www.tagsistant.net/>

²²Reperibile su <http://caretaker.altervista.org/index.php?id=20>

in gerarchia piatta, altre due tabelle che memorizzano l'insieme dei tag e la relazione multi-a-molti fra i tag stessi e gli oggetti. Nel caso di TaggedNotes queste tabelle sono rappresentabili, usando l'SQL, come:

```
Tags ( IDtag INTEGER PRIMARY KEY; Tag VARCHAR(64) );  
MapTags ( IDMap INTEGER PRIMARY KEY; IDtag INTEGER;  
          IDNote INTEGER );
```

Dove IDNote rappresenta la chiave primaria della tabella che memorizza le note. Questa rappresentazione usante tre tabelle è la più semplice fra quelle normalizzate, e l'inserimento e associazione dei tag alle note diventa un'operazione molto semplice da implementare.

4.2.3 Folksonomies: organizzazione dei contenuti partendo “dal basso”

Una interessante metodologia che sta emergendo di recente dal web, e che combina l'uso delle gerarchie piatte con i principi del “Web 2.0”, e in particolare l'uso degli utenti per aggiungere valore ai dati, sono le cosiddette “folksonomy” o con brutto termine italianizzato, folksonomie.

Le folksonomie nascono con il progetto di Joshua Shacter, del.icio.us²³, un sito che si proponeva di raccogliere i siti preferiti dei suoi utenti, in maniera da averli sempre raggiungibili via web, e che permetteva di associare a questi link una serie di tag. Fin dall'inizio del.icio.us rendeva pubblici questi link sottomessi dai suoi utenti, permettendo a ogni utente di vedere i siti preferiti di un'altro utente. [Del.icio.us](http://del.icio.us) permetteva anche di vedere quali siti erano stati classificati con un certo tag.

L'idea di base delle folksonomie è essenzialmente questa: lasciare che siano gli utenti a classificare i dati, associando liberamente dei tag ad essi. Il termine in se è stato ideato da Thomas Vander Wal in [25], che definisce le folksonomie come:

²³Attivo e visibile su <http://del.icio.us>

La vera novità è dal punto di vista sociale, visto che la classificazione non è più fatta “dall’alto”, ovvero da un gruppo di esperti o responsabili che classifica secondo schemi rigidi e ben definiti, e che si pone al di sopra degli utenti, ma “dal basso”, ovvero dagli utenti stessi, usando la “hive mind” o mente collettiva degli stessi, e ottenendo una classificazione che è meno rigida, e che magari varia nel tempo in maniera fluida, seguendo le preferenze degli utenti. In questo è una classificazione più “democratica” nel suo senso più puro.

In questo senso diviene importante che il sito che offre la folksonomia invogli gli utenti a collaborare fra loro, come ricorda Isabel de Maurissens in [26]:

“La condivisione è importante; attribuire un tag tratto dal proprio vocabolario è necessario, ma non sufficiente perché si possa parlare di folksonomy. Secondo Vander Wal, il tag deve essere "parlante" perché possa essere ricercato e recuperato da altri e permettere un scambio tra culture e discipline diverse”

Gli svantaggi di una folksonomia sono principalmente legati ai tag usati dagli utenti, in particolare come analizzato da Mathes in [27], i problemi principali sono:

- L’ambiguità dei termini scelti: ad esempio al termine “Apple” può essere associato sia un frutto (la mela), sia la ditta informatica, sia la casa discografica dei Beatles. Questo limita la validità dei tag. Gli acronimi in particolare possono creare moltissime ambiguità sul loro significato.
- La gestione di tag che siano composti da più parole, che però è un limite tecnico facilmente superabile separando i tag con un separatore diverso dallo spazio.
- I sinonimi o comunque i tag che pur essendo diversi hanno lo stesso significato. In questa categoria cade ad esempio l’uso di tag plurali e singolari dello stesso termine (“Libro” e “Libri” ad esempio), o i veri e propri sinonimi, che creano rumore di fondo inutile nei tag.

Attualmente sono in creazione sistemi che superino questi problemi, in particolare la possibilità di usare tag di più parole e l'uso di "contenitori" che raggruppino insieme un insieme di tag con lo stesso significato, come i "bundle tags" introdotti di recente su del.icio.us.

Altro approccio per risolvere questi problemi è quello che il sistema semplicemente suggerisca dinamicamente all'utente, quando inserisce i tag, i tag più usati che somiglino a quelli che sta inserendo, facendo quindi una pressione "sociale" affinché usi dei tag che siano già diffusamente usati dagli altri utenti, eliminando quindi il "rumore di fondo" dei tag accennato prima.

Altre tecniche per ricavare una migliore classificazione dalle folksonomie è quella di limitare a visualizzare come associati a un oggetto solo un certo numero di tag che siano quelli più frequentemente associati a quell'oggetto, come sta facendo LastFM ad esempio, ottenendo quindi una classificazione che sia effettivamente quella più espressa dalla "mente collettiva" dei suoi utenti.

Va inoltre ricordato che è possibile avere un approccio misto, in cui gli utenti siano liberi di associare tag, ma solo tag scelti da una lista decisa non dagli utenti ma da un gruppo di responsabili: è la folksonomia a "vocabolario limitato", che in certi casi può ottenere ottimi risultati.

Per quanto riguarda il futuro uso delle folksonomie, le parole di Isabel de Maurissens sono chiare:

"Nel paesaggio digitale²⁴, la folksonomy porta con sé una vera rivoluzione che troverà molte applicazioni soprattutto in ambienti in cui esiste già un linguaggio condiviso, come una banca dati di buone pratiche, una piattaforma e-learning, un settore di ricerca, un ambito professionale ma anche nella didattica. Per gli utenti creare un proprio indirizzo, ad. es. in del.icio.us,

²⁴ cfr. Rosenfeld L., Morville P., "Dalla definizione di Information Architecture" in Caprio L., Ghiglione B., "Information Architecture", Tecniche nuove, 2003, p. 20.

4 Spunti dal progetto Capri 2.0

è semplice e intuitivo; porre i tag anche, in quanto, se la risorsa è già stata inserita, automaticamente il sistema suggerisce i tags già usati da altri navigatori.”[26]

E inoltre, forse le folksonomie sono la chiave per la soluzione del problema principale del Web semantico, ovvero associare appunto significati semantici ai contenuti web esistenti, mediante meccanismi che usino le folksonomie come base per arrivare tramite vocabolari limitati o strati di mediazione ad associare un significato semantico agli oggetti classificati dalla folksonomia in maniera utile per il web semantico, come suggerito da vari autori [28, 29, 30].

5 Conclusioni

Sicuramente, l'impatto avuto dai CMS sul mondo del web è stato molteplice, ancora di più se si considera che il uso è strettamente legato a uno dei concetti del cosiddetto "Web 2.0", che è quello di rendere gli utenti non dei fruitori passivi, ma dei contributori attivi, che generano contenuto che arricchisce i servizi web che si offrono. L'apparizione di questo "contenuto utente" ha aperto la problematica della sua gestione, che è l'esigenza che i CMS cercano di colmare, mettendosi in mezzo fra i contenuti stessi e gli utenti che, nel medesimo tempo, sono sia consumatori passivi che contributori attivi di questi contenuti. Questa problematica sarà sicuramente una delle problematiche sempre presenti nel futuro di un web usato come piattaforma per offrire servizi che prevedano una partecipazione attiva degli utenti, magari in forme che attualmente non immaginiamo, e che sempre più pongono i CMS (anche nella forma di CMS ad-hoc per specifici progetti) come un pezzo fondamentale del futuro del web.

I CMS inoltre sono uno strumento che ha permesso di abbassare la "soglia di difficoltà" che bisogna superare per essere un contributore attivo alla realtà del web, sia solamente per avere un proprio sito personale, sia per progettare e implementare un insieme di servizi complessi offerti sul web. Questo grazie al fatto che i CMS siano sistemi "già fatti" che si occupano di alcuni aspetti tipici dei servizi offerti sul web: la gestione degli utenti che possono accedere e contribuire, la gestione dei dati su cui il servizio funziona, la regolazione dell'accesso a una parte dei dati offerti. Sono parti che fondamentalmente fanno parte della maggioranza dei sistemi che offrono servizi sul web, e che quindi possono

5 Conclusioni

essere scaricati sul CMS, semplificando il compito del creatore del sito web. In questo si può dire che i CMS fungono come “framework” già pronti che semplificano la vita dello sviluppatore di siti web, permettendogli di concentrarsi solo sugli aspetti unici del particolare progetto, e avendo una base solida e ben sviluppata che si occupa di quegli aspetti generali.

Questo abbassamento della soglia ha permesso due cose: da una parte, una maggior parte degli utenti ha adesso degli strumenti per diventare parte attiva creando più facilmente un proprio sito, dall'altra ha permesso la creazione di servizi che appoggiandosi sui CMS permette agli utenti di aggiungere e rendere pubblico sul web un loro contributo informativo, come nel caso dei servizi che offrono a tutti con facilità la creazione di un proprio blog.

L'aumento del numero di utenti che attivamente contribuiscono al web ha degli impatti rilevanti non solo puramente informatici, com'è stato evidenziato con l'impatto avuto sul giornalismo dai blog e dalla blogosfera, impatti che influenzano già la società attuale, e ancora di più nel futuro. L'arricchimento dei dati disponibili in maniera facile e pubblica sul web, dovuta all'influsso di una quantità molto maggiore di utenti che forniscono un loro contributo, è sicuramente un fatto positivo per la Rete, anche se porta alcuni problemi e domande da rispondere, soprattutto sulla qualità di tale contributo, sul come ridurre le informazioni che non sono rilevanti in quel momento, nei mezzi che si usano per reperire le informazioni e nella eventuale regolamentazione anche legale di questa enorme massa di utenti attivi.

Tornando ai CMS in se, e guardando le direzioni future di sviluppo degli stessi, si possono individuare vari aspetti chiave dei CMS che nel futuro possono ricevere delle innovazioni.

Uno di questi aspetti è il dominio applicativo: finora la maggioranza dei CMS si è concentrata sulla gestione dei contenuti di siti pubblici sul web, con un contributo da parte degli utenti più o meno ampio, ma questo non è l'unico ambito possibile in cui esiste la

5 Conclusioni

problematica di organizzare e gestire una serie di contenuti. Ad esempio alcuni CMS, vedi [31], iniziano a mostrare interesse in ambiti quali quello delle applicazioni intranet aziendali, come gestori di contenuti e documenti aziendali o per creare workspaces collaborativi di produzione, dominio che in fondo è quello in cui i primi CMS sono nati. E in futuro non è da escludere che arrivino a magari un ambito di uso personale, come un'applicazioni per gestire in locale contenuti personali.

Ma anche l'aspetto della strutturazione dei contenuti gestiti può presentare evoluzioni in futuro. Oggi quasi tutti i CMS usano un sistema gerarchico affiancato spesso da un sistema basato su una gerarchia piatta e tag, ma possibili evoluzioni future possono essere l'uso di strutture di classificazione di origine bibliografica come le Topic Maps¹, con alcuni studi al riguardo sono che sono già stati fatti, ad esempio [32] e [33]. Non è da escludere, quindi, in futuro l'apparizione di CMS che siano basati sulle TopicMaps, definiti negli studi odierni come ITMS (Integrated Topic Management System, sistema di gestione di argomenti integrati), e di cui ci sono già alcuni prototipi in lavorazione². L'uso delle topic maps aggiungerebbe un nuovo livello aggiuntivo alla struttura a gerarchia piatta, le associazioni fra i topic/tag, che permetterebbero una esplorazione dei contenuti diversa e con una potenzialità ancora da definire con precisione. Altra direzione possibile è l'evoluzione dei CMS specializzati in argomenti collegati, i cosiddetti Wiki, che hanno come punto centrale l'alta integrazione e collegamento dei contenuti gestiti, particolarmente adatti per creare compendi di tipo enciclopedico, in particolare aperti e costruiti sul contributo dei suoi utenti, come la famosa Wikipedia.

Altro punto di interesse per il futuro dei CMS è il loro ruolo di "motore" su cui costruire il vero servizio/sito web, e quindi il ruolo principale che ha l'espandibilità dei CMS come funzionalità. I metodi attualmente usati sono principalmente l'aggiunta di moduli di codice che aggiungono funzionalità, plug-in o moduli che si voglia. In questo una

¹vedi lo standard ISO/IEC 13250:2003, che definisce lo standard ISO delle Topic Maps

²vedi il lavoro di Andrea Resmini per il dipartimento di Storia e Informatica dell'Università di Bologna, Abulafia, reperibile su <http://www.resmini.net/projects>

5 Conclusioni

definizione di una API precisa, magari universale, che permetta di scrivere plug-in universali per tutti i CMS sarebbe un'interessante, anche se probabilmente impraticabile, evoluzione. A parte l'espansione delle funzionalità, l'altro lato critico di un CMS come interfaccia con lo sviluppatore è la creazione del front-end o interfaccia con l'utente. In questo abbiamo attualmente un insieme molto variegato di CMS che permettono più o meno libertà, dagli schemi quasi fissi di PHP-Fusion, alla quasi totale libertà dei template di Wordpress, vere e proprie pagine PHP che si interfacciano con wordpress tramite una API. Però strade nuove che strutturino la struttura complessiva in maniere diverse da una serie di template associati per nome a certe sezioni dell'interfaccia utente sono in sviluppo. Esempi interessanti sono la divisione in template, snippets - ovvero pezzi di codice PHP richiamabile dai template, e chunks - pezzi di codice XHTML con variabili di template che possono essere richiamabili dagli snippets, adottata ad esempio da Modx³; o l'uso di un linguaggio apposito per scrivere i template delle pagine, come usato ad esempio da Texpattern⁴.

Parlando di interfaccia con l'utente, questo è un'altro punto in cui il futuro può riservare ancora nuovi avanzamenti nei CMS. Partendo dalla una divisione "modulare" dell'interfaccia utente, espandibile aggiungendo moduli javascript, i cosiddetti widgets, e di cui si sta delineando una API standard⁵, permettendo quindi ad utenti non esperti di "personalizzarsi" la propria interfaccia in maniera semplice e senza dover scrivere codice, sino ad arrivare alla questione spinosa dell'editor che l'utente deve usare per aggiungere contenuto, che può portare a problemi, studiati in [34] e [35], di codice XHTML non standard o non comunque uniforme con il resto del template. Interessante in questo senso il cosi-

³Un CMS reperibile su <http://www.modxcms.com>

⁴CMS che usa un suo linguaggio interno per definire gli equivalenti dei template, Textile, reperibile su <http://textpattern.com/>

⁵vedi la Universal Widget API annunciata da Netvibes, che promette appunto di uniformare l'API di tutte le applicazioni che vogliono supportare l'aggiunta di moduli all'interfaccia utente web, delineata su <http://blog.netvibes.com/?2007/03/09/125-new-developer-website-preview-of-universal-widget-api-uwa>

5 Conclusioni

detto approccio “WYSWYM”⁶, invece che il classico approccio “WYSWYG”⁷, seguito da alcuni editor javascript per l’inserimento di contenuti XHTML come WYMentor⁸, che rinunciando a un minimo di chiarezza con l’utente permettono di avere un codice finale prodotto più uniforme e standard. Altra possibilità per risolvere questo problema è limitare le possibilità dell’utente usando un linguaggio di inserimento dei dati più limitato dell’XHTML come ad esempio BBcode⁹. In questo discorso si inserisce ovviamente anche l’evoluzione delle interfacce web, con tecniche quali AJAX, ovvero l’uso di javascript e di chiamate asincrone verso la base dati, e in generale l’uso di javascript e delle future nuove capacità che i browser metteranno a disposizione (come HTML5, le versioni future della DOM, e le evoluzioni di Flash e Silverlight).

Inoltre l’evoluzione dei CMS e in generale del web come piattaforma porta anche delle innovazioni legate al nuovo modo di usare gli utenti, non più come usufruttori passivi ma come agenti attivi. Un esempio interessante in questo senso sono le folksonomie, gerarchie piatte unite al nuovo ruolo attivo degli utenti, che sono un nuovo modo di creare classificazioni dei dati, che sfruttano la “mente collettiva” degli utenti. Una classificazione inoltre che non è statica, ma che si adegua esattamente a quello che gli utenti vogliono, seguendo le loro indicazioni e non quelle di un gruppo di esperti magari scollegato dai desideri degli utenti. Questi nuovi modi di pensare il problema dell’organizzazione dei dati possono essere la chiave, almeno secondo [28], per effettuare quella transizione al Web Semantico che da tempo è studiata, e che porterà a nuove soluzioni al problema della organizzazione di contenuti in maniera utile per gli utenti, organizzazioni che oggi sono difficili anche solo da prevedere. Esempi già esistenti di questa nuova organizzazione è il successo di servizi come del.icio.us o Flickr, che hanno nella folksonomia la loro base per la classificazione dei dati che gestiscono, siano essi link o immagini.

⁶“What You See Is What You Mean”, Quello che vedi è quello che intendi. Approccio che cerca di evidenziare il valore semantico di quello che si scrive e non quello grafico finale.

⁷“What You See Is What You Get”, Quello che vedi è quello che ottieni, che cerca di rendere mentre si inserisce esattamente l’aspetto finale del testo.

⁸reperibile su <http://www.wymeditor.org/en/>

⁹metalinguaggio standard nato dai forum web, descritto meglio su <http://www.bbcode.org>

5 Conclusioni

Lasciando il quadro generale e concentrandoci sul progetto che ho svolto, nel futuro Capri 2.0, che attualmente è in fase di beta testing, spera innanzitutto di diventare, come si era prefissa, un punto di riferimento per una comunità di utenti interessata alle problematiche locali dell'isola di Capri. La crescita di questa comunità permetterà poi di effettuare iniziative che portino l'attenzione nazionale sui problemi dell'isola, dando la giusta voce ai problemi di quest'isola, spesso dimenticati, e creando una base comune per promuovere attività che portino magari a una "seconda versione" migliore dell'isola. Questa base di utenti porterà al poter implementare servizi che siano basati su di essi, magari sfruttando appunto folksonomie o comunque contributi attivi di vario tipo degli utenti.

Sotto il profilo tecnico, a parte gli ovvi e costanti aggiornamenti e manutenzione della piattaforma WordPress, in futuro si può prevedere l'aggiunta di nuove funzionalità, quali gallerie fotografiche, archivi video, altri mezzi di comunicazione fra i membri e altri servizi secondo le richieste della comunità che si sta creando, quali ad esempio forum web dedicati. Queste funzionalità si cercherà di implementarle usando i meccanismi naturali di estensione della piattaforma Wordpress (plug-in e tema), e in fede con la visione OpenSource del progetto, magari le funzionalità più interessanti potranno essere separate e rilasciate in licenza GPL alla comunità, in maniera da creare progetti separati indipendenti, che siano utili sia alla comunità che aiutati e sostenuti da essa. In questo a breve alcune delle funzionalità che ho implementato (spedizione delle mail alle classi di utenti e le funzioni statistiche sugli articoli) sono in fase di raffinazione prima di essere rilasciati come plug-in separati e internazionalizzati, sotto licenza open-source.

Personalmente questo progetto mi ha permesso di accrescere la mia esperienza in ambito professionale, di approfondire il lavoro partendo da un framework o piattaforma già esistenti, in questo caso Wordpress, e per la prima volta di dover imparare a integrare ed estendere un CMS, cosa che non avevo mai fatto prima. Questa esperienza mi ha sicuramente fornito nuovi strumenti e aperto una visione su un campo delle applicazioni

5 Conclusioni

web che non avevo finora incrociato, e che mi ha permesso da un lato di usare in maniera concreta le nozioni che già possedevo, perché insegnatemi durante il percorso universitario, in particolare per quanto riguarda l'interazione uomo-macchina e le applicazioni web, e dall'altro lato di dovermi confrontare con problematiche che non avevo affrontato prima, come l'organizzazione aziendale del lavoro, il riutilizzo di soluzioni già esistenti, e la comunicazione con il cliente finale che solo il lavoro reale in una azienda porta a doversi confrontare.

Un'ultimo appunto e ringraziamento va alla DigitalSparks che mi ha permesso di effettuare il progetto in esperienza di telelavoro, che mi ha messo di fronte ai vantaggi e svantaggi di questa soluzione, con una esperienza che personalmente è stata molto positiva, e che spero anche per la ditta sia stata altrettanto positiva.

Bibliografia

- [1] CMS Wiki, “*History of CMS*”, <http://www.cmswiki.com/tiki-index.php?page=HistoryOfCms>.
- [2] Jeremy Reimer, “*A history of the GUI*”, Ars Technica, <http://arstechnica.com/articles/paedia/gui.ars>.
- [3] Tim Berners-Lee, “*Information Management: A proposal*”, CERN, 1989, visibile su <http://www.w3.org/History/1989/proposal.html>.
- [4] Bob Doyle, “*CMS Genesis: Who Did What When?*”, E-Content, <http://www.ecmag.net/Articles/ArticleReader.aspx?ArticleID=6819&ContextSubtypeID=71>.
- [5] WAI, “*Web Content Accessibility Guidelines 1.0*”, 5 maggio 1999, <http://www.w3.org/TR/WAIWEBCONTENT>.
- [6] “*Writing a plugin*” in AA.VV., “*Wordpress Codex*”, http://codex.wordpress.org/writing_a_plugin.
- [7] “*Plugin API*”, in AA.VV., “*Wordpress Codex*”, http://codex.wordpress.org/Plugin_API.
- [8] “*Adding administrator menu*” in AA.VV., “*Wordpress Codex*”, http://codex.wordpress.org/Adding_Administration_Menus.
- [9] “*Function Reference*” in AA.VV., “*Wordpress Codex*”, http://codex.wordpress.org/Function_Reference.

Bibliografia

- [10] “*Creating tables with plugins*” in AA.VV., “*Wordpress Codex*”, http://codex.wordpress.org/Creating_Tables_with_Plugins.
- [11] “*Roles and Capabilities*” in AA.VV., “*Wordpress Codex*”, http://codex.wordpress.org/Roles_and_Capabilities.
- [12] “*Predefined Variables*”, nella documentazione di PHP su <http://www.php.net/manual/it/reserved.variables.php>.
- [13] “*Theme Development*”, in AA.VV., “*Wordpress Codex*”, http://codex.wordpress.org/Theme_Development.
- [14] “*Template Tags*”, in AA.VV., “*Wordpress Codex*”, http://codex.wordpress.org/Template_Tags.
- [15] Roberto Polillo, “*Il check up dei siti web*”, Apogeo, Milano 2004.
- [16] “*Creating Admin Themes*”, in AA.VV., “*Wordpress Codex*”, http://codex.wordpress.org/Creating_Admin_Themes.
- [17] W3C organization, “*XHTML1.0, The extendible hypertext mark-up language*”, <http://www.w3.org/TR/xhtml1/>
- [18] Tim O’Reilly, “*What is Web2.0*”, <http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>.
- [19] Paul Graham, “*Web 2.0*”, <http://www.paulgraham.com/web20.html>.
- [20] Nate Anderson, “*Tim Berners-Lee on Web 2.0: "nobody even knows what it means"*”, <http://arstechnica.com/news.ars/post/20060901-7650.html>.
- [21] Chris Anderson, “*The long tail*”, <http://www.wired.com/wired/archive/12.10/tail.html>.
- [22] Blockbuster, “*Blog history in timeline form*”, http://www.blockstar.com/blog/blog_timeline.html.
- [23] Rob Malda, “*A brief history of Slashdot, part 1: Chips & Dits*”, <http://meta.slashdot.org/article.pl?sid=07/10/02/1553218>.
- [24] John Beeler, “*Gmail and flatness*”, <http://www.beelerspace.com/index.php?p=806>.

Bibliografia

- [25] Thomas Vander Wal, “*Folksonomy*”, <http://vanderwal.net/folksonomy.html>.
- [26] Isabel de Maurissens, “*Folksonomy: una classificazione sociale del web*”, Indire, <http://www.indire.it/content/index.php?action=read&id=1332>.
- [27] Adam Mathes, “*Folksonomies - Cooperative Classification and Communication Through Shared Metadata*”, University of Illinois, <http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html>.
- [28] Specia L., Motta E., “*Integrating folksonomies with the Semantic Web*” in AA.VV., “*The Semantic Web: Research and Applications*”, Springer, Berlin, 2007, pp. 624-639.
- [29] Szomszor, M., Cattuto, C., Alani, H., O’Hara, K., Baldassarri, A., Loreto, V. and Servedio, V. D. P., “*Folksonomies, the Semantic Web, and Movie Recommendation*” in AA.VV., “*4th European Semantic Web Conference, Bridging the Gap between Semantic Web and Web 2.0*”, <http://eprints.ecs.soton.ac.uk/14007/1/ESWC2007.pdf>.
- [30] Tom Gruber, “*Where the Social Web meets the Semantic Web*” dalla quinta ISWC, reperibile su <http://tomgruber.org/writing/social-meets-semantic-web.htm>.
- [31] Alexander Limi, “*18 things I wish were true about Plone*”, <http://limi.net/articles/18-things-i-wish-were-true-about-plone>.
- [32] Lars Marius Garshol, “*Topic Maps in Content Management: The rise of ITMS*”, <http://www.ontopia.net/topicmaps/materials/itms.html>.
- [33] Steve Carton, “*Classifying content with Topic Maps*”, <http://www.idealliance.org/proceedings/xml04/papers/145/XTMClassification.html>.
- [34] Nicol J., “*The trouble with content management systems*”, <http://f6design.com/journal/2007/11/30/the-trouble-with-content-management-systems/>.

Bibliografia

- [35] Claudio Garau, “*Editor WYSWYG fra esigenze dei clienti e pulizia del codice*”, <http://www.onecms.it/09/05/2008/editore-wyswyg-tra-le-esigenze-dei-clienti-e-pulizia-del-codice>.