Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Nuove strategie per la conoscenza, la valorizzazione e la fruizione del patrimonio archeologico dell'Area Flegrea

Francesco Maiorano e Mario Rosario Guarracino

RT-ICAR-NA-2013-2 **Marzo 2013**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Napoli, Via P. Castellino 111, I-80131 Napoli, Tel: +39-0816139508, Fax: +39-0816139531, e-mail: napoli@icar.cnr.it, URL: *www.na.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Nuove strategie per la conoscenza, la valorizzazione e la fruizione del patrimonio archeologico dell'Area Flegrea

Francesco Maiorano e Mario Rosario Guarracino [1]

Rapporto Tecnico N.:
RT-ICAR-NA-2013-2

Data:
Marzo 2013

---

[1] ICAR-CNR

Indice

1. Introduzione

In questo documento, analizziamo gli aspetti che costituiscono la progettazione, la realizzazione e la messa in opera di un progetto web. Lo scopo è la realizzazione di una Web Application che fornisca un sistema per la catalogazione, la gestione e la visualizzazione geolocalizzata di reperti archeologici e architettonici, oggetto di studio del gruppo di ricerca dell'Università degli Studi di Napoli Federico II. La piattaforma realizzata prende il nome di Matchmed ed è stata realizzata durante il tirocinio presso l'ICAR-CNR tra fine 2012 e inizio 2013.
Da oltre vent'anni si producono a livello sperimentale sistemi informativi finalizzati ad introdurre nuove modalità di fruizione del bene culturale. In questo ambito di sperimentazione la tecnologia ha svolto un ruolo trainante spesso a discapito dei contenuti, alimentando l'esigenza di definire nuovi modelli comunicativi in sintonia con la cultura iconografica contemporanea. La maggior parte delle applicazioni è stata infatti finalizzata alla realizzazione di "cataloghi digitali" che consentono il libero accesso all'informazione, banche dati e archivi on-line, che offrono la possibilità di studiare il patrimonio artistico e culturale anche a distanza, favorendo la divulgazione della conoscenza. A queste applicazioni, ormai abbastanza collaudate, se ne aggiungono altre, che hanno come obiettivo la definizione di nuovi e diversi livelli di fruizione dei beni culturali, reti materiali e immateriali che strutturano la conoscenza. Il tema proposto dall'Unità di ricerca locale - Federico II - ha come oggetto lo studio del patrimonio archeologico sommerso dell'area flegrea e delle relazioni con i siti archeologici terrestri. In questo ambito territoriale il fenomeno del bradisismo negativo e positivo ha determinato nel corso dei secoli rapide modifiche con il conseguente sprofondamento dell'antica fascia costiera e la sommersione di tutti gli edifici che su questa fascia insistevano. Ne deriva che siti di grande importanza, come il porto di Miseno, sede della flotta militare romana, e Baia antica, sono oggi in gran parte sommersi. Questo evento rende i reperti archeologici difficilmente fruibili, se non tramite le escursioni subacquee organizzate dalle varie associazioni locali, con un'oggettiva difficoltà ad inquadrare il reperto stesso in un contesto più ampio, ne scaturisce la necessità di costruire un progetto di comunicazione che sia in grado di rendere esplicito l'evento e consentire la contestualizzazione del reperto, per andare oltre il fascino dell'immersione. L'obiettivo è quello di costruire un diverso livello di conoscenza in grado di valorizzare il bene culturale e consentirne la comprensione. Le lacune insite nel concetto stesso di reperto e la possibilità di contestualizzarlo offerto dalla virtualità, creano nuove modalità di fruizione. Nella maggior parte delle applicazioni in questo settore la componente spettacolare e ludica generalmente prevale, la ricostruzione è quasi sempre affidata ad esperti che spesso curano prevalentemente l'aspetto tecnologico della comunicazione senza entrare nel merito del contenuto da comunicare. La ricerca è quindi prevalentemente orientata a definire un linguaggio in grado di rendere distinguibile il dato certo dall'ipotesi, e l'utilizzo della tecnologia non è finalizzato ad ottenere simulazioni virtuali sofisticate ma piuttosto a individuare nuovi codici comunicativi. In questo senso la Virtual Archeology consente di realizzare un nuovo prodotto la cui materia prima è il bene culturale che sottoposto ad un processo di elaborazione si trasforma in conoscenza. l'inserimento di questo prodotto culturale nella piattaforma multimediale (network, Netheritage), consentirà di diffondere il livello di conoscenze acquisito ma soprattutto di sollecitare il dibattito sulle metodologie di analisi, di studio e di valorizzazione dei beni culturali.
La scelta dell'area di studio presenta una molteplicità di problematiche tanto da favorire la sperimentazione su diversi livelli:

- a fronte di un ricco patrimonio naturale, archeologico e storico, il territorio è caratterizzato da una condizione di degrado ambientale diffusa e ciò rende necessaria l'individuazione di idonee strategie per la riqualificazione ambientale;

- la gestione del patrimonio culturale non risulta essere integrata e quindi è necessario mettere a sistema la conoscenza creando una rete che favorisca l'interscambio e tenda a superare il gap comunicativo connesso alla divulgazione del patrimonio archeologico;

- alla naturale difficoltà di fruizione del reperto archeologico si aggiunge nel caso specifico l'ulteriore difficoltà derivante dal fatto che molti dei reperti sono sommersi, ciò rende necessario più che negli altri casi la definizione di un prodotto culturale che trasformi il bene in conoscenza.

I principali obiettivi della ricerca sono quindi:

- Costruire una mappa interattiva (piattaforma multimediale) dei siti archeologici dell'area dei Campi Flegrei per la definizione di percorsi turistici alternativi in grado di innescare meccanismi di riqualificazione territoriale;

- Sperimentare nuove metodologie per il rilievo, l'analisi e la rappresentazione dei reperti archeologici sommersi;

- Individuare nuovi codici comunicativi per la fruizione del patrimonio archeologico (virtual archeology).

L'ubiquità di accesso a informazioni online apre nuovi scenari nella fruizione dei beni culturali. La consultazione di banche dati e archivi in situ, integrata con strumenti interattivi, amplia e completa l'esperienza del visitatore. Le attuali soluzioni tecnologiche consentono di ampliare l'esperienza del visitatore di siti archeologici. Richiedono l'utilizzo di smartphone o tablet (iPhone, iPad, ecc...) sfruttandone le funzionalità (giroscopio, GPS, videocamera). Con Matchmed è possibile catalogare un numero arbitrario di reperti, memorizzando per ognuno di essi informazioni aggiuntive: area, comune, tema, ambito, ubicazione, tipologia, condizione, proprietà, accessibile, visitabile, vegetazione, latitudine, longitudine, altitudine, conservazione e note. Per ogni reperto è possibile poi allegare contenuti multimediali come immagini, video, file dwg o dxf, file di testo. Allo stato attuale il sistema Matchmed conta 117 reperti catalogati, con un'immagine per ognuno di essi. Le soluzioni scelte per la realizzazione hanno lo scopo di rendere possibile l'ampliamento del pool di dati del gruppo di ricerca Matchmed fornendo uno strumento semplice, intuitivo e di facile utilizzo.

1.2 Analisi e studio di fattibilità

La necessità di effettuare uno studio di fattibilità nasce dal fatto che si è individuato un possibile progetto che per dimensione economica, complessità dell'intervento, incertezza sui requisiti e scelte da compiere sulle possibili alternative richiede un approfondimento prima di avviare la fase realizzativa, pena la possibilità di avviare un progetto ad alto rischio di insuccesso.
Lo studio di fattibilità quindi nasce sempre in presenza di una "idea progettuale" già esistente che comprende gli elementi essenziali dell'individuazione del problema e dell'area di intervento, le principali linee di intervento previste, una definizione preliminare del progetto.
Non è quindi compito dello studio di fattibilità quello di individuare le esigenze di fondo che stanno all'origine del progetto e quindi non ha senso caricare quest'analisi di teoriche necessità di definire, attraverso tecniche specifiche, le opzioni di fondo del "dove operare", "cosa ricercare" ecc...
Il vero problema in questa fase è di dare concretezza all'idea progettuale e di fornire tutti gli elementi per l'avvio della fase realizzativa, sviluppando ciò che in precedenza era solo una "idea progettuale", inevitabilmente generica e non sufficientemente verificata e valutata.
E' attraverso lo studio di fattibilità che si dà sostanza all'ipotesi di sistema che si intende realizzare e se ne cominciano a definire contenuti, servizi da erogare, componenti, si descrivono e misurano i benefici attesi, si individuano gli impegni necessari alla realizzazione e i relativi costi, si evidenziano e valutano rischi definendo nel contempo le modalità di realizzazione e di controllo del progetto. Lo scopo è di acquisire una

maggiore consapevolezza sul problema che ci consente di arrivare ad una decisione ragionata sull'investimento.

La necessità di definire in termini generali i progetti e di formalizzare tale definizione attraverso la descrizione degli elementi essenziali del progetto di massima vale per tutti i progetti di informatizzazione. Tuttavia la realizzazione di un vero e proprio studio di fattibilità, che inevitabilmente implica impegno e risorse di tempo, va prevista essenzialmente per quei progetti ai quali possa portare un beneficio in termini di valutazione preliminare, progetti tesi alla realizzazione di sistemi informativi che impattano sui processi di servizio e di dimensioni medio-grandi. Tra questi figurano in particolare i progetti di realizzazione o reingegnerizzazione di sistemi applicativi, di infrastrutture tecnologiche, progetti di automazione d'ufficio.

La situazione attuale

Si vuole realizzare una piattaforma web attraverso la quale si possa catalogare e gestire i reperti archeologici e architettonici che sono oggetto di studio del gruppo di ricerca della facoltà di architettura dell'Università Federico II. Il progetto commissionato consiste di una web application che offra un'interfaccia web usabile per la consultazione e la catalogazione. La web application nella sua forma più semplice, si riconduce al paradigma CRUD (Create, Read, Update, Delete).
Il paradigma CRUD associa utenti e risorse, o loro aggregazioni, indicando i relativi privilegi di accesso. Per ogni coppia utente/risorsa, CRUD, indica se c'è un privilegio di accesso e di che tipo (C, R, U, D). Gli utenti possono essere persone o applicazioni che accedono a una determinata risorsa. La risorsa ammette una differente granularità: può essere un singolo campo (valore numerico o un file), una directory o una vista sui dati oppure l'insieme delle informazioni gestite con un certo applicativo.
Attualmente, il sistema proposto non è informatizzato e i reperti sono catalogati da operatori che, per ogni reperto, annotano su dispositivi o materiale cartaceo le informazioni relative. Successivamente le informazioni raccolte devono essere vagliate, controllate e aggregate in fogli Excel. In maniera rudimentale si associa un numero identificativo incrementale ad ogni reperto per mantenere una corrispondenza con il materiale multimediale raccolto, che viene memorizzato su file system. Le informazioni relative ad un solo reperto si presentano quindi sconnesse, e anche l'eventuale presentazione delle stesse è possibile solo dopo un lavoro di riorganizzazione.
Lo scopo di questo progetto è di realizzare un sistema in grado di offrire un modo semplice e intuitivo di catalogare i reperti e tutto il materiale relativo agli stessi (immagini, video, documentazione) e, allo stesso tempo, offrire un'interfaccia per la consultazione e l'analisi geolocalizzata.

Progetto di massima della soluzione

Si propone un sistema basato sulla famosa piattaforma di blogging Wordpress, opportunamente estesa per implementare le specifiche funzionali richieste dal gruppo di ricerca della facoltà di architettura dell'Università Federico II. Con una piattaforma di questo tipo, siamo riusciti ad abbattere i tempi di sviluppo per la messa in opera di un sistema CRUD e, grazie alla natura modulare di Wordpress, abbiamo sviluppato alcuni plugin per gestire la geolocalizzazione dei reperti, l'export di informazioni in formato Excel e la presentazione di contenuti multimediali.
Inoltre, Wordpress offre le seguenti funzionalità:

- Editing avanzato: infatti, nasce come piattaforma per l'editoria online, ottimizzata per la gestione dei blog. La gestione dei contenuti è per questo ottimizzata e facile da gestire anche per chi non ha competenza tecnica o di sviluppo. Gestisce in modo avanzato l'editing HTML ed evita in maniera intelligente la "produzione di codice" non ottimizzato tipico delle piattaforme online con rich text editor.

- User Management: è possibile gestire in modo semplice la registrazione degli utenti ed i relativi permessi il commento dei post o degli articoli pubblicati sul sito. Wordpress gestisce diversi livelli di utenza che permettono di gestire un sito multiautore.

- Ottimizzazione SEO: Wordpress è ottimizzato per i motori di ricerca in modo nativo. Il software permette di gestire, in modo automatico, i meta tag title, description e keyword oppure di utilizzare dei plug-in per ottimizzare manualmente tutti i tag SEO più importanti, generare la sitemap del sito, e gestire tutte le attività di ottimizzazione per l'indicizzazione sui motori di ricerca. La gestione degli URL rewrite, ovvero i Permalinks di Wordpress, permette di generare indirizzi ottimizzati per tutte le pagine del sito.

- Gestione template: è possibile gestire in moto ottimizzato il layout esterno del sito.

- Moduli avanzati: uno dei principali vantaggi dei CMS e di Wordpress è la possibilità di attivare facilmente nuove funzionalità per il sito sviluppando e installando nuovi moduli. Nel corso degli anni, la community ha già sviluppato diversi moduli aggiuntivi.

- Aggiornamento facile: Wordpress implementa un sistema di gestione degli aggiornamenti di versione molto semplice. E' possibile infatti aggiornare il sistema direttamente dal pannello di controllo.

Analizzeremo nel dettagli i vantaggi dell'utilizzo della piattaforma Wordpress nella sezione 2.4. Confronteremo la piattaforma scelta con altre concorrenti evidenziando le differenze con i framework più diffusi per lo sviluppo web, giustificando la scelta di Wordpress.

Raccomandazioni per le fasi realizzative

Si raccomanda l'adozione di metodologie di sviluppo agili che consistono di un gruppo di metodi per la realizzazione di software basati sullo sviluppo iterativo ed incrementale, dove le specifiche funzionali e le soluzioni evolvono attraverso la collaborazione tra gruppi di lavoro. Queste tecniche promuovono la pianificazione adattiva, l'evoluzione del software e la messa in opera, un approccio iterativo in un tempo prestabilito che incoraggi la risposta rapida e flessibile ai cambiamenti di progetto. Possiamo vedere lo sviluppo agile come un framework concettuale che migliori le interazioni previste durante lo sviluppo. Analizzeremo nel dettaglio l'insieme di queste tecniche nella sezione 3.1.
Lo sviluppo deve avvenire in ambiente protetto e su macchine locali al team di sviluppo. La versione di sviluppo della web application deve poggiare su un database che contiene un subset dei dati presenti in produzione. E' importante adottare un sistema di controllo versione integrato con un sistema di task management al fine di supportare il singolo nel raggiungimento degli obiettivi di sviluppo, o team di lavoro nella collaborazione e la condivisione di informazioni per il completamento degli obiettivi collettivi.
Gestire i task di più individui o veri e propri gruppi di lavoro può richiedere un software di task management specializzato. La fase di task management è parte del project management e può rappresentare le fondamenta per un'organizzazione efficiente del flusso di lavoro.

2. Progettazione

In questo capitolo, approfondiremo l'analisi della progettazione della web application Matchmed introducendo i linguaggi e le tecnologie scelte. In ingegneria del software, la progettazione è una fase del ciclo di vita del software. Sulla base della specifica dei requisiti prodotta dall'analisi, il progetto definisce come tali requisiti saranno soddisfatti, entrando nel merito della struttura che dovrà essere data al sistema software che deve essere realizzato.

Inizieremo con una panoramica sulla storia del web e il cambiamento che i siti web hanno subito nel corso degli anni, ponendo la nostra attenzione sull'evoluzione delle interfacce e delle piattaforme per lo sviluppo di portali e CMS (Content Management System). Introdurremo le tecnologie per lo sviluppo web, focalizzandoci su quelle scelte per la realizzazione di Matchmed. Confronteremo framework e piattaforme CMS per supportare la scelta della piattaforma utilizzata.

2.1 Evoluzione del web e delle interfacce

Quando Tim Berners-Lee preparò la prima proposta per il World Wide Web nel 1990, l'idea di base era piuttosto semplice: creare una "ragnatela" (web) di informazioni collegate fra loro attraverso ipertesti e URI (Uniform Resource Identifiers). La possibilità di collegare i documenti più disparati in ogni pare del mondo rappresentava un enorme potenziale da un punto di vista accademico: studiosi e ricercatori avrebbero potuto avere accesso quasi istantaneo ai materiali di riferimento bibliografico. La prima incarnazione del linguaggio HTML (HyperText Markup Language) non presentava granché oltre a comandi per la formattazione del testo e i collegamenti, e non si trattava certo di una piattaforma per realizzare software riccamente interattivo, bensì per condividere semplicemente le varie tipologie di informazioni testuali ed esplicative che dominavano la tarda era della stampa. Furono queste pagine web statiche il terreno di crescita del Web.
Con l'evolversi del Web, le varie imprese commerciali non tardarono a notare un grosso potenziale nella possibilità di distribuire alle masse informazioni su prodotti e servizi. La generazione successiva del Web vide un aumento delle capacità di formattare e visualizzare informazioni grazie all'evoluzione congiunta dell'HTML, che rispondeva sempre più alle esigenze e alle aspettative di questi utenti più abili nell'utilizzo dei nuovi media. Ma una piccola compagnia chiamata Netscape avrebbe presto accelerato l'evoluzione del web.

2.2 Usabilità di un sito web e psicologia del visitatore

Secondo la definizione data dalla norma ISO 9241, l'usabilità è il "grado in cui un prodotto può essere usato da particolari utenti per raggiungere certi obiettivi con efficiacia, efficienza e soddisfazione in uno specifico contesto d'uso".
La normativa ISO 9241 è del 1993 e si riferisce a prodotti informatici in genere. Tuttavia l'usabilità è un concetto molto precedente ed esteso: nasce negli anni 60 nell'ambito dell'ergonomia in relazione a qualunque interazione uomo-artefatto. In seguito trova maggior fortuna proprio per i software, nel settore dell'ergonomia cognitiva. In questo specifico settore dell'ergonomia si studia il modo in cui un utente si costruisce un modello mentale del prodotto che sta usando, e si crea perciò determinate aspettative sul suo funzionamento.
Compito degli studi di usabilità è fare in modo che il modello mentale di chi ha progettato il software (design model), da cui deriva il suo reale funzionamento, corrisponda il più possibile al modello mentale del funzionamento del software così come se lo costruisce l'utente finale (user model). L'usabilità nasce dunque soprattutto come ausilio alla progettazione, e si applica in particolare alle interfacce. E' con l'interfaccia di un software, infatti, che l'utente si relaziona.
Ad ogni sua azione l'interfaccia proporrà un risultato, un cambiamento di stato. Poco importa, ai fini dell'usabilità, come l'interfaccia sia giunta a quello stato, attraverso cioè quali meccanismi di programmazione, che rimangono racchiusi in una vera e propria scatola nera impermeabile all'utente.
Va sottolineato che l'usabilità ha senso solo in presenza di un utente e di una relazione d'uso, e non esiste nel prodotto in sè. Le tecniche di usabilità tentano dunque di porre al centro dell'attenzione progettuale proprio l'utente. Può sembrare un dettaglio da poco, ma non lo è. In realtà sembra ovvio che il prodotto, siccome deve venir usato da un utente, venga progettato per lui. Invece, dato che fino a tutti gli anni 70 il computer non era un prodotto di massa, i principali utilizzatori dei prodotti software finivano per essere gli stessi progettisti o persone esperte con una formazione simile ai progettisti. Di conseguenza l'usabilità era un problema assente (o meglio implicito): se uno sapeva progettare un software, sapeva anche usarlo. Design

model e user model coincidevano.
Tale problema è invece emerso dapprima negli anni 80, con la diffusione delle tecnologie informatiche a livello di ufficio e di famiglia, ed è definitivamente esploso negli anni 90, con la diffusione del personal computer. Improvvisamente gli utenti finali del software (ma naturalmente anche dell'hardware) non erano più i progettisti. Ora il mercato imponeva di vendere macchine e programmi a chiunque. E "chiunque" non era un esperto di informatica.
Il seme che avrebbe consentito l'utilizzo del computer a masse di utenti inesperti fu gettato dal Macintosh, il primo computer con un sistema operativo completamente visuale, basato sulla metafora della scrivania e dello spostamento intuitivo di oggetti. Il cambiamento fu epocale.
Macintosh si impose come computer user-friendly, orientato all'uso da parte di persone completamente a digiuno di informatica. Poco dopo Windows riutilizzò la metafore con una politica di vendita molto più decisa, addirittura aggressiva. Oggi computer che montano sistemi operativi come Mac OS, Windows o Linux sono sulle nostre scrivanie, e tutti i programmi che utilizziamo presentano un'interfaccia di tipo visuale, metaforico. Non serve essere dei superesperti per farli funzionare. L'usabilità, dunque, trionfa? Non proprio, ma il discorso per il momento esula dai nostri fini. La cosa più importante, comunque, è che i progettisti abbiano iniziato a spostare l'attenzione sull'utente finale, e che questo sia avvenuto (non senza grosse difficoltà) solo dal momento in cui tale utente-utilizzatore non era più un esperto di informatica.
Con l'avvento di internet e la proliferazione dei siti web, il problema dell'usabilità si è spostato sul nuovo dominio, dove naturalmente dovrà tener conto della caratteristiche dell'interazione, in qualche caso anche molto diverse da quelle tipiche del software (applicazioni desktop): se un software viene normalmente usato dopo essere stato acquistato, un sito web prima viene usato, e solo se l'uso risulta soddisfacente può dar vita ad una transazione ed eventualmente un guadagno. Ne consegue che, se entro certi limiti l'usabilità nei software è un problema che incide relativamente sui guadagni, nei siti web è determinante, perché è condizione preliminare al realizzarsi stesso del guadagno. Così i problemi da porsi sono: a cosa serve un determinato sito web? Chi lo userà e cosa si aspetterà di trovarci? Le stesse domande che guidano tutto il progetto e anche la stesura dei contenuti. Gli esperti di usabilità interagiscono quindi con la progettazione di un sito in ogni fase della timeline di realizzazione: dalla definizione degli obiettivi alla costruzione dei contenuti, per andare in definitiva a incidere sull'interfaccia finale (che dipende da tutti questi e da tutti gli altri fattori coinvolti nel progetto). Se in alcuni casi questo può portare anche ad un ripensamento dell'information design e di alcuni meccanismi di programmazione, tale risultato emergerà esclusivamente attraverso l'interfaccia.

La natura del web pone però un problema nuovo all'usabilità, assente nel campo del software.
Poiché l'interfaccia di un sito ha anche compiti di comunicazione della brand identity, e più in generale di immagine, l'usabilità oltre che con le funzionalità di un sito deve fare i conti con il design, inteso appunto nel senso di veicolo d'immagine. Il rapporto pare tutt'altro che semplice, perché si tratta di conciliare due logiche di lavoro opposte. Orientata ad una pragmatica standardizzazione che salvaguardi la funzionalità, quella dei guru dell'usabilità estrema; tutta concentrata al valore dell'innovazione come attributo di un design seducente, anche a rischio di perdere usabilità, quella di alcuni art director o web designer. Sarebbe facile cavarsela dicendo che la verità sta nel mezzo. Il rapporto fra usabilità ed estetica ricorda molto di più il dibattito in corso all'interno del design industriale (così com'è discusso in "Dall'oggetto all'interfaccia" di Gui Bonsiepe, Feltrinelli '93) che il tradizionale campo del software engeneering, del tutto avulso da tentazioni estetiche.

2.3 Tecnologie per il Web

In questo capitolo, analizziamo nel dettaglio le tecnologie web più diffuse per lo sviluppo di portali, siti web, e-commerce e web application in generale. Focalizziamo l'attenzione sugli standard di realizzazione, stabiliti dal W3C (World Wide Web Consortium) e proponiamo una struttura di massima per l'ambiente di sviluppo e di produzione. Analizziamo nel dettaglio le tecnologie utilizzate in questo progetto, distinguendo quelle

“client-side” e quelle “server-side”. Infine, discutiamo alcune tecniche di interazione tra client e server sincrone e asincrone.

2.3.1 Lo Standard W3C

Il motto del W3C è “Guidare il World Wide Web fino al massimo del suo potenziale”.
Il World Wide Web Consortium, anche conosciuto come W3C, è un’organizzazione non governativa internazionale che ha come scopo quello di sviluppare tutte le potenzialità del web.
Al fine di riuscire nel proprio intento, la principale attività del W3C consiste nello stabilire standard tecnici per il World Wide Web inerenti sia i linguaggi di markup che i protocolli di comunicazione.
Il W3C è stato fondato nell’ottobre del 1994 al MIT (Massachussetts Institute of Technology) dal “padre” del Web, Tim Berners-Lee, in collaborazione con il CERN (laboratorio da cui proveniva).
Attualmente esistono diversi tipi di apparecchi che accedono a internet (come cellulari, pc, smartphones, tablet ecc...). Ciò è possibile solo grazie ad un comune linguaggio di comunicazione (un protocollo) tra server, PC e altri dispositivi. Il W3C si occupa di aggiornare e creare queste specifiche.
Il web ha un potenziale praticamente illimitato, ed apre nuove strade ai portatori di handicap, anche gravi. Il W3C cerca di studiare i modi per rendere quanto più agevole l’accesso al web da parte di questa categoria di persone.
Il W3C cerca inoltre di garantire che il web sia libero, evitando che interessi di qualsiasi genere possano porre freno alla libertà sul web. Chiunque può creare un documento html e metterlo liberamente online.
Nell’ambito del W3C sono stati proposti, discussi, definiti e ufficializzati oltre 50 standard industriali di vasta portata, tra cui HTTP, URI, URL, XML, XHTML, CSS, XSLT, DOM, SOAP, PNG.

2.3.2 Cenni su ambienti di sviluppo e produzione

Nello sviluppo software, sia applicazioni desktop, sia web, sia mobile, è necessario costruire un ambiente di sviluppo adatto e specifico per il tipo di applicazione che si vuole sviluppare.
Esistono tool, detti IDE (Integrated Development Enviroment), di ausilio allo sviluppo software, ad esempio per la stesura del codice, per il tracciamento di errori o note durante in corso d’opera. E’ quindi un software che, in fase di programmazione, aiuta i programmatori nello sviluppo del codice sorgente di un programma. Normalmente è uno strumento software che consiste di più componenti, da cui appunto il nome integrato:

- un editor di codice sorgente
- un compilatore e/o un interprete
- un tool di building automatico
- un debugger

A volte è integrato anche con un sistema di controllo di versione e con uno o più tool per semplificare la costruzione di una GUI. Alcuni IDE, rivolti allo sviluppo di software orientato agli oggetti, comprendono anche un navigatore di classi, un analizzatore di oggetti e un diagramma della gerarchia delle classi. Sebbene siano in uso alcuni IDE multi-linguaggio, come Eclipse, NetBeans e Visual Studio, generalmente gli IDE sono rivolti ad uno specifico linguaggio di programmazione, come Visual Basic o Delphi.
Tra gli IDE più noti possiamo trovare: Eclipse, Aptana (un’estensione di Eclipse pensata per lo sviluppo web), NetBeans, Visual Studio, Dev-C++, XCode.

Nello sviluppo di un’applicazione web il solo editor di codice sorgente a volte non basta, anche se corredato da funzionalità object-oriented o di altro tipo (aspect-oriented, ecc...).

Per sua natura una webapp consiste di un codice che, solitamente, dev'essere interpretato per essere eseguito, ad esempio linguaggi come PHP o Ruby sono linguaggi interpretati mentre Java è un linguaggio compilato. Una Web application deve essere eseguita su un server per poter offrire on-line le funzionalità implementate. In fase di sviluppo, il server viene montato sulla macchina di sviluppo al fine di testare periodicamente il funzionamento del software che si sta sviluppando.
Per questo progetto l'ambiente di sviluppo scelto è LAMP.
LAMP è un acronimo che indica una piattaforma software per lo sviluppo di applicazioni web che prende il nome dalle iniziali dei componenti software con cui è realizzata.
I componenti base da cui prende il nome sono:

- GNU/Linux: sistema operativo
- Apache: il server web
- MySql: database management system
- Perl, PHP e/o Python: i linguaggi di scripting

La piattaforma LAMP è una delle più utilizzate a livello mondiale. Ognuna delle applicazioni dalle quali è composta è predisposta per l'eccellente funzionamento in concomitanza con le altre. Un pacchetto contenente tali software per diversi sistemi operativi è XAMPP, il quale riunisce a sè le piattaforme LAMP (o LAMPP) e WAMP (o WAMPP).
Esempi di server LAMP sono quelli in uso da Wikipedia, quelli di Mozilla e quelli di Facebook.

2.3.3 Tecnologie lato client: HTML, CSS, JavaScript

Nell'ambito delle reti di calcolatori, il termine lato client (client-side in inglese) indica le operazioni da un client in un rapporto client-server.
Un rapporto tipico di questo tipo è quello effettuato da un'applicazione, come un Browser, che avvia una connessione ad un server per poter funzionare.
Le operazioni client-side sono effettuate quando le risorse richieste non possono essere raggiunte dal server, e quindi da script che non girano sul client. Ad esempio, la validazione dei campi di un form di inserimento, oppure l'animazione di un interfaccia, o ancora l'accesso a risorse disponibili solo quando l'utente visualizza la pagina sul proprio computer, come webcam o tastiera.
Inoltre se non serve che i risultati siano salvati sul server, le operazioni risulteranno nettamente più veloci (ovviamente a seconda della potenza del client). Quando il protocollo utilizzato è uno dei più comuni, come l'HTTP (HyperText Transfer Protocol) o l'FTP (File Transfer Protocol), possono esistere vari programmi client (ad esempio i browser moderni supportano spesso sia HTTP che FTP). Di conseguenza spesso uno script che passa per questi protocolli è compatibile con più PC.
Programmi che girano sul computer locale, senza mai inviare o ricevere dati in rete, non sono considerati programmi client, e così le operazioni di tali programmi non vengono considerate operazioni client-side proprio perché non c'è interazione client-server.
I progetti di calcolo distribuito come SETI@home o Google Earth contano principalmente sul lato client. I client SETI@home si connettono al server a cui richiedono alcuni dati. Il server seleziona una serie di dati (operazione server-side), e li invia al client. Il client poi analizza i dati (operazione client-side), e, quando l'analisi è completata, trasmette i suoi risultati al server.
Analogamente, siti web o web application che fanno largo utilizzo di librerie JavaScript (come jQuery) per realizzare interfacce, eseguono necessariamente operazioni di tipo server side.

HTML

HTML è l'acronimo di Hypertext Markup Language ("Linguaggio di contrassegno per gli ipertesti") e non è un linguaggio di programmazione (come possono esserlo il C, C++, Java, oppure linguaggi di scripting come il PHP, Perl o JavaScript).
Si tratta di un linguaggio di contrassegno che permette di indicare come disporre gli elementi all'interno di una pagina: le informazioni vengono date attraverso marcatori detti "tag".
Questo significa che l'HTML non possiede meccanismi che consentono di prendere delle decisioni e non è in grado di compiere interazioni, né ha altri costrutti propri della programmazione.
Pur essendo dotato di una sua sintassi, l'HTML non presuppone la logica ferrea e inappuntabile dei linguaggi di programmazione. Se la sintassi non viene rispettata probabilmente non otterremo la visualizzazione della pagina desiderata, ma nessun errore.
L'organizzazione che si occupa di standardizzare la sintassi del linguaggio HTML (il W3C) ha rilasciato diverse versioni di questo linguaggio (HTML 2.0, HTML 3.2, HTML 4.0 e il recente HTML5). Da un certo punto in poi, l'HTML si è evoluto in XHTML, cioè l'HTML riformulato come linguaggio XML. L'HTML è il linguaggio con cui si può indicare come i vari elementi vanno disposti in una pagina Web. Un documento HTML non è nient'altro che un file di testo con delle indicazioni sul colore del testo, sulla posizione delle immagini della pagina, su come far scorrere il testo e altre operazioni simili.
l'HTML è un linguaggio di pubblico dominio la cui sintassi è stabilita dal World Wide Web Consortium. E' stato sviluppato negli anni ottanta da Tim Berners-Lee al CERN di Ginevra assieme al noto protocollo HTTP che supporta invece il trasferimento di documenti in tale formato. Nel corso degli anni, seguendo l'evoluzione di Internet, l'HTML ha subito molte revisioni, ampliamenti e miglioramenti indicati secondo la classica numerazione usata per descrivere le versioni dei software.
Nel dicembre 1999 è stata resa disponibile la versione 4.01 mentre nel 2007 è ricominciata l'attività di specifica con la definizione di HTML5. Lo sviluppo venne avviato dal gruppo di lavoro Web Hypertext Application Technology Working Group (WHATWG, fondato nel 2004 da sviluppatori appartenenti ad Apple, Mozilla Foundation e Opera Software) che si pose come obiettivo quello di progettare specifiche per lo sviluppo di applicazioni web, focalizzandosi su miglioramenti e aggiunte ad HTML e alle tecnologie correlate.
Inizialmente in contrasto con il W3C per le lungaggini nel processo di evoluzione dello standard HTML e per la decisione del W3C di orientare la standardizzazione verso l'XHTML 2 che non garantiva retro compatibilità, lo stesso W3C ha poi riconosciuto valide le motivazioni del WHATWG e ha quindi annunciato di creare un apposito gruppo per la standardizzazione dell'HTML5 e abbandonare l'XHTML 2.0.
Dal 2007 il WHATWG ha collaborato con il W3C in tale processo di standardizzazione, per poi decidere nel 2012 di separarsi dal processo, creando di fatto due versioni dell'HTML5: la versione WHATWG viene definita come "HTML Living Standard" e quindi in continua evoluzione, mentre quella del W3C sarà una unica versione corrispondente ad uno "snapshot" del Living Standard.
Il W3C ha annunciato che la prima versione dello standard sarà pronta per fine 2014 e l'HTML5.1 per il 2016.
Le novità introdotte dall'HTML5 rispetto all'HTML4 sono finalizzate soprattutto a migliorare il disaccoppiamento tra struttura, definita dal markup, caratteristiche di resa (tipo di carattere, colori, ecc..), definite dalle direttive di stile, e contenuti di una pagina web, definiti dal testo vero e proprio.
Inoltre l'HTML5 prevede il supporto per la memorizzazione locale di grosse quantità di dati scaricati dal browser, per consentire l'utilizzo di applicazioni basate su web (come per esempio le caselle di posta di Google o altri servizi analoghi) anche in assenza di collegamento a Internet.
In particolare l'HTML5 introduce le seguenti novità:

- le regole per la strutturazione del testo in capitoli, paragrafi e sezioni sono più stringenti
- vengono introdotti elementi di controllo per i menu di navigazione

- gli elementi di controllo per i moduli elettronici sono migliorati ed estesi
- vengono introdotti elementi specifici per il controllo di contenuti multimediali (tag <video> e <audio>)
- elementi di scarso utilizzo sono deprecati o eliminati
- gli attributi finalizzati all'accessibilità sono stati estesi a tutti i tag, finora previsti solo per alcuni
- viene introdotto il supporto a Canvas che permette di utilizzare JavaScript per creare animazioni e grafica bitmap
- introduzione della geolocalizzazione, dovuta ad una forte espansione di sistemi operativi mobili (come iOS e Android)
- introduzione di un sistema alternativo ai cookie, chiamato Web Storage, più efficiente e che consente un notevole risparmio di banda
- standardizzazione di programmi JavaScript
- sostituzione del lungo e complesso doctype con un semplice <!DOCTYPE html>

CSS

Cascading Style Sheets (CSS) è un linguaggio che definisce fogli di stile usato per descrivere regole di presentazione (la formattazione e l'aspetto) di un documento scritto con linguaggio di markup (HTML). L'applicazione più comune è definire lo stile di una pagina Web scritta in HTML o XHTML, ma il linguaggio può essere applicato anche ad altri tipi di documenti come XML o SVG.
Il CSS è stato progettato principalmente per separare la presentazione del documento dal suo contenuto. Questa separazione può migliorare l'accessibilità al contenuto, presentare più flessibilità e controllo nella specifica delle caratteristiche di presentazione, più pagine possono condividere la formattazione e può ridurre la complessità e la ripetizione di contenuto strutturale (permettendo un design così detto "table-less"). CSS permette inoltre la presentazione di una pagina in stili differenti a seconda del metodo di rendering utilizzato, che sia su schermo, in stampa o letto da una voce automatica (in presenza di browser per diversamente abili). CSS può essere utilizzato per visualizzare pagine su dispositivi di diverse dimensioni (come i moderni dispositivi mobile, smartphone e tablet). CSS definisce uno schema di priorità per determinare quale regola deve essere applicata se più d'una corrisponde ad un particolare elemento. In questo schema a "cascata", le priorità sono calcolate e assegnate alle regole al fine di generare un risultato prevedibile.
CSS è una sintassi semplice e utilizza parole chiave in Inglese per specificare i nomi degli attributi.
Un foglio di stile CSS consiste di una serie di regole. Ogni regola, o insieme di queste, consiste di uno o più selettori e un blocco di dichiarazione. Un blocco di dichiarazione consiste di una lista di attributi racchiusi tra parentesi graffe. Ogni dichiarazione è a sua volta una proprietà che definisce un valore (separati da due punti). Se ci sono più dichiarazioni all'interno di un blocco, ognuna di queste va separata con punto e virgola.
Nel CSS i selettori sono utilizzati per dichiarare quale parte del markup si applica lo stile. I selettori possono essere applicati a tutti gli elementi di un tipo specifico, oppure a elementi specificati da un attributo, oppure a elementi in relazione alla loro posizione nel documento. Molto comune è l'organizzazione ad albero, dove un elemento detto "radice" può contenere "figli" o "foglie" al suo interno, creando una gerarchia nella struttura del markup.
Esistono varie versioni del CSS. Ognuna si basa sulla versione precedente, aggiungendo nuove funzionalità: CSS1, CSS2, CSS3 e CSS4.

La prima specifica risale al 1996 con il CSS1 che diventò lo standard ufficiale e raccomandato dal W3C. CSS2, rilasciato nel 1998, aggiungeva nuove funzionalità come il posizionamento assoluto o relativo degli elementi, la profondità gestita tramite l'attributo z-index. A partire dal febbraio 2004 si è iniziato a lavorare su una nuova versione del CSS, chiamata CSS2.1, che risolveva alcuni bug presenti nella versione precedente. Dopo diverse edizioni e modifiche, il CSS2.1 diventa raccomandato dal W3C solo nel giugno 2011.
Il CSS3, a differenza del CSS2 che è una raccolta di definizioni e funzionalità, viene diviso in documenti separati chiamati "moduli". Ogni modulo aggiunge una nuova funzionalità ed estende quelle del CSS2, al fine di preservare la retro compatibilità. Il lavoro sul CSS3 è iniziato alla pubblicazione delle raccomandazioni CSS2. La prima bozza del CSS3 fu pubblicata nel 1999.
Il CSS4 non è ancora una vera e propria versione che raccoglie nuove funzionalità, ma i moduli introdotti nelle specifiche CSS3 possono evolvere autonomamente nel CSS4. Quindi alcuni componenti, che si basano su specifiche CSS2.1, restano di livello 3, mentre esistono nuove funzionalità basate sulla versione 3.
Per facilitare lo sviluppo di interfacce web, sono stati creati alcuni framework che consistono di una serie di librerie preparate per rispettare gli standard e per formattare pagine web utilizzando il linguaggio Cascading Style Sheets. Alcuni di questi framework, basati su griglie per definire il layout, sono: Blueprint, 960 grid e YUI CSS grids. Come per le librerie nella programmazione classica, i framework CSS sono di solito incorporati nelle pagine tramite richiami a file esterni con estensione .css referenziati nel tag <head>.

JavaScript

JavaScript è un linguaggio di scripting orientato agli oggetti comunemente usato nella creazione di siti web. Fu originariamente sviluppato da Brendan Eich della Netscape Communications con il nome di Mocha e successivamente di LiveScript. In seguito rinominato JavaScript e formalizzato con una sintassi più vicina a quella del linguaggio Java di Oracle. JavaScript è stato standardizzato per la prima volta tra il 1997 e 1999 dalla ECMA.
Non c'è una vera relazione tra Java e JavaScript. Le loro somiglianze sono soprattutto nella sintassi (che per entrambi deriva dal linguaggio C). Le semantiche dei linguaggi sono piuttosto diverse e in particolare i loro object model non hanno nessuna relazione, ma ciò nonostante JavaScript si è imposto come linguaggio di scripting lato client più diffuso per lo sviluppo web.
Le caratteristiche principali sono:

- il codice non viene compilato ma interpretato lato client da un interprete incluso nel browser
- la sintassi è simile a quella del C, C++ e Java
- definisce funzionalità tipiche dei linguaggi di programmazione ad alto livello (strutture di controllo, cicli, ecc...) e consente l'utilizzo del paradigma object oriented.
- è un linguaggio debolmente tipizzato
- è un linguaggio debolmente orientato agli oggetti. Ad esempio, il meccanismo dell'ereditarietà è lontano da quello del linguaggio Java (che è un linguaggio fortemente orientato agli oggetti). In JavaScript gli oggetti ricordano più array associativi del Perl o del PHP che gli oggetti di Java o C++.

Poiché JavaScript è un linguaggio lato client, il codice viene eseguito direttamente sul client e non sul server. Il vantaggio di questo approccio è che, anche con la presenza di script particolarmente complessi, il web server non viene sovraccaricato a causa delle richieste dei client. Di contro, nel caso di script che presentino un codice sorgente particolarmente grande, il tempo per lo scaricamento può diventare abbastanza lungo. Un altro svantaggio è il seguente: ogni informazione che presuppone un accesso a dati memorizzati in un

database remoto deve essere rimandata ad un linguaggio che effettui esplicitamente la transazione, per poi restituire i risultati ad una o più variabili JavaScript. Con l'avvento di AJAX questi limiti sono superati.

2.2.4 Tecnologie lato server: PHP, MySQL, Apache

In modo del analogo alle operazioni client-side, nelle reti informatiche, l'espressione lato server (server-side in inglese) fa riferimento a operazioni compiute dal server in un ambito client-server.
Di solito, un server è un programma software, come un server web, che gira su una macchina remota (chiamata per estensione "server") rimanendo in ascolto su determinate porte e raggiungibile da un computer client. Alcune operazioni devono essere compiute dal lato server perché richiedono l'accesso a informazioni o funzionalità non disponibili sul client, o richiedono misure di sicurezza che sarebbero inaffidabili se eseguite lato client.
Le operazioni lato server includono anche trattamento e immagazzinamento di dati da client a server, perché possano essere disponibili a un gruppo di client.
Si indica con questo termine il database management system DBMS. Sistema centralizzato (programmi coordinati) o distribuito (rete) che permette di memorizzare, modificare ed estrarre informazioni da un database.
Il termine server-side è nato con l'avvento del web, in una rete locale, il server è l'elaboratore che svolge funzioni di servizio così da allegerire gli elaboratori collegati. Nell'ambito della programmazione web, si definiscono linguaggi lato server quei linguaggi di programmazione che vengono interpretati ed elaborati dal server il quale, successivamente, invia i risultati al client (il browser dell'utente). I linguaggi lato server più diffusi sono il PHP e l'ASP, ma da tempo ormai prendono sempre più spazio linguaggi come Ruby. Un programma scritto con questo tipo di linguaggi viene sempre elaborato sul server e mai reso disponibile all'utente, il quale può visualizzare solo il risultato del programma. Questo concetto è molto importante in quanto sta alla base della sicurezza e dell'affidabilità offerti dalla programmazione lato server.
Per citare un esempio, Wikipedia usa script lato client per la modifica delle pagine. Premendo il tasto "Modifica" sarà possibile modificare il testo della pagina (operazione client-side). Invece, premendo il tasto "Salva la pagina", la modifica sarà inviata al server che la salverà nel suo database (operazione server-side), rendendo disponibile la modifica per altri utenti.

PHP

PHP è l'acronimo di "Hypertext Preprocessor" ed è un linguaggio di programmazione interpretato, originariamente concepito per la programmazione Web e per la realizzazione di pagine Web dinamiche. L'interprete ha una licenza open source e libera (ma non compatibile con la GPL).
Attualmente è utilizzato principalmente per sviluppare applicazioni web lato server ma può essere usato anche per scrivere script a riga di comando o applicazioni stand-alone con interfaccia grafica. Nato nel 1994 ad opera di Rasmus Lerdorf, PHP era in origine una raccolta di script CGI che permettevano una facile gestione delle pagine personali. Il pacchetto originario venne esteso in C, aggiungendo funzionalità quali il supporto al database mSQL, prevedendo la possibilità di integrare il codice PHP nel codice HTML in modo da semplificare la realizzazione di pagine dinamiche. A questo punto il linguaggio cominciò a godere di una certa popolarità tra i progetti open source del web e venne così notato da due giovani programmatori: Zeev Suraski e Andi Gutmans. I due collaborarono nel 1998 con Lerdorf allo sviluppo della terza versione di PHP riscrivendone il motore che fu battezzato Zend. Le caratteristiche chiave della versione PHP 3.0 erano la straordinaria estensibilità, la connettività ai database e il supporto iniziale per il paradigma a oggetti. Nel 2005 la configurazione LAMP (Linux, Apache, MySQL, PHP) supera il 50% del totale dei server sulla rete mondiale.
Tra le caratteristiche principali, PHP riprende per molti versi la sintassi del C, come peraltro fanno molti linguaggi moderni, e del Perl. E' un linguaggio a tipizzazione debole e dalla versione 5 migliore il supporto al paradigma di programmazione ad oggetti. Certi costrutti derivati dal C, come gli operatori fra bit e la

gestione di stirnghe come array, permettono in alcuni casi di agire a basso livello. Tuttavia, PHP resta un linguaggio di alto livello, caratteristica rafforzata dall'esistenza di un gran numero di API, oltre 3.000 funzioni presenti nel nucleo base. Inoltre, PHP è in grado di interfacciarsi a innumerevoli database tra cui MySQL, PostgreSQL, Oracle, Firebird, IBM DB2, Microsoft SQL Server, solo per citarne alcuni, e supporta numerose tecnologie XML, SOAP, IMAP, FTP. Si integra anche con altri linguaggi di programmazione quali Java e .NET. E' anche ottimamente integrato con il database MySQL, presentando più di una API per interagire con esso.
La versione 5, integra al suo interno un piccolo database embedded, SQLite.
PHP non ha ancora un supporto nativo per le stringhe Unicode o multibyte; il supporto Unicode è in fase di sviluppo per una futura versione di PHP, e consentirà di usare caratteri non ASCII in stringhe e nomi di funzioni, classi e metodi.
La percentuale di software non sicuro scritto in PHP, sul totale di tutte le falle nei software elencate dal Common Vulnerabilities and Exposures, ammontava al: 12% nel 2003, 20% nel 2004, 28% nel 2005, 43% nel 2006, 36% nel 2007, 34.8% nel 2008, 29.9% nel 2009 e 27.2% nel 2010.[4] La maggior parte di questi punti vulnerabili possono essere sfruttati tramite remoto, ovvero senza accedere al computer che ospita l'applicazione vulnerabile. Le falle più comuni sono dovute al mancato adempimento delle best practice nella programmazione e da vulnerabilità presenti in codice scritto in versioni vecchie di PHP.

MySQL

MySQL, definito come Oracle MySQL, è un Relational database management system (RDMS), composto da un client con interfaccia a riga di comando e un server, entrambi disponibili sia per sistemi Unix o Unix-like come GNU/Linux che per Windows.
Dal 1996 supporta la maggior parte della sintassi SQL e si prevede in futuro il pieno rispetto dello standard ANSI. Possiede interfacce per diversi linguaggi, compreso un driver ODBC, due driver Java, un driver per .NET ed una libreria python. MySQL fa parte di pacchetti come piattaforma LAMP e WAMP usati per sviluppare in locale siti web, anche in congiunzione con pacchetti come Wordpress o altri.

Apache

Apache HTTP Server, o più comunemente Apache è il nome dato alla piattaforma server Web modulare più diffusa, in grado di operare da sistemi operativi UNIX/Linux e Microsoft.
Apache è un software che realizza le funzioni di trasporto delle informazioni, di collegamento, ha il vantaggio di offrire anche funzioni di controllo per la sicurezza come quelli che compie il proxy.
Il Web Server Apache presenta un'architettura modulare, quindi ad ogni richiesta del client vengono svolte funzioni specifiche da ogni modulo di cui è composto, come unità indipendenti. Ciascun modulo si occupa di una funzionalità, ed il controllo è gestito dal core.
I moduli:

- Core: programma principale composto da un ciclo sequenziale di chiamate ai moduli
- Translation: traduce la richiesta del client
- Access Control: controlla eventuali richiesta dannose
- MIME Type: verifica il tipo di contenuto
- Response: invia la risposta al client e attiva eventuali procedure
- Logging: tiene traccia di tutte le operazioni effettuate

Il core suddivide la richiesta ai vari moduli in modo sequenziale, usando i parametri di uscita di un modulo come parametri di accesso per l'altro, creando così l'illusione di una comunicazione orizzontale fra i moduli. Sopra il ciclo del core c'è un ulteriore ciclo di polling svolto da un demone che interroga continuamente le linee logiche da cui possono pervenire messaggi di richiesta.

2.2.5 Interazione avanzate client-server: Ajax

Dal 2001 al 2005 il World Wide Web è passato attraverso una crescita spaventosa in termini di tecnologie e metodologie utilizzate per portare questo strumento statico alla vita. Le brochure e i cataloghi online hanno smesso di dominare Internet nel momento in cui le applicazioni web sono divenute una parte significativa delle destinazioni online. Le applicazioni web si differenziavano dai loro "antenati" siti web nell'offrire ai propri utenti un servizio immediato non solo di tipo informativo.
Che lo scopo fosse la gestione di procedure commerciali o l'interesse personale, gli sviluppatori erano costretti a creare nuovi paradigmi di interazione, visto che gli utenti iniziavano ad aspettarsi funzionalità più avanzate.
Stimolato da tecnologie poco note e ancor meno utilizzate, già presenti nei browser web da tempo, il Web ha compiuto un notevole progresso, sconvolgendo il modo d'uso tradizionale che necessitava il caricamento completo di una pagina ogni volta che si accedeva a nuove informazioni o a una nuova parte della logica dell'applicazione. Le varie aziende cominciarono a effettuare esperimenti con la ricarica dinamica di porzioni di pagine web, per trasmettere al client solo una quantità ridotta di dati, realizzando in questo modo un'esperienza utente più veloce e migliore.
A condurre questo movimento è stato Google. Dopo che il gigante dei motori di ricerca uscì allo scoperto, iniziarono ad apparire nuovi esperimenti condotti dai suoi ingegneri da una sezione del sito chiamata Google Labs (labs.google.com). Molti dei progetti dei Google Labs, come Google Suggest e Google Maps, consistevano di un'unica pagina web che pur non venendo mai scaricata veniva tuttavia costantemente aggiornata. Tali innovazioni, che cominciarono a portare le affordance delle interfacce dei software desktop entro i confini di un browser, furono elogiate in tutto il Web e considerate come l'avvento di una nuova era dello sviluppo web. E lo furono davvero.
Svariati prodotti open source e commerciali diedero il via allo sviluppo per approfittare di questo nuovo modello di applicazione web. Tali progetti tentarono di spiegare la propria tecnologia servendosi di espressioni JavaScript remoting, web remote procedure calls, e dynamic updating. Presto però avrebbe prevalso un nuovo termine.
Nel febbraio 2005 Jesse James Garrett di Adaptive Path LLC pubblicò online un articolo intitolato "Ajax: a New Approach to Web Applications", tuttora disponibile all'indirizzo www.adaptivepath.com/pubblications/essays/archives/000385.php. In questo scritto Garrett spiegava come, a suo avviso, le applicazioni web stessero colmando lo scarto fra il Web e le applicazioni desktop tradizionali. Citava poi le nuove tecnologie e molti dei progetti Google come esempi del fatto che modelli di interazione utente tipicamente basati su applicazioni desktop fossero ora impiegati nel Web. Di seguito venivano le due frasi che avrebbero sollevato un polverone di interesse, agitazione e controversie: "Google suggest e Google Maps sono due esempi di un nuovo approccio alle applicazioni web che noi di Adaptive Path abbiamo chiamato Ajax. Il nome è una contrazione di Asynchronous JavaScript + XML e rappresenta un passaggio fondamentale in ciò che è possibile realizzare sul Web." Dal quel momento, un'ondata di articoli su Ajax, di esempi di codice e di discussioni apparve ovunque sul Web. Gli sviluppatori ne parlavano nei propri blog, le riviste di tecnologia nelle proprie pagine, e le aziende iniziarono a legare ad Ajax i proprio prodotti. Malgrado la presenza di svariate FAQ a supplemento dell'articolo di Garrett, rimane tuttora una certa confusione su ciò che concerne la vera essenza di Ajax. In parole semplici, Ajax non è altro che un approccio all'interazione web. Tale approccio riguarda la trasmissione di solo una piccola quantità di informazioni da e verso il server, in modo da offrire all'utente l'esperienza più reattiva possibile. Invece del modello di applicazione web tradizionale, in cui il browser stesso si occupa di avviare richieste al server web e di processare le richieste provenienti da quest'ultimo, il modello Ajax prevede un livello intermedio, che Garrett chiama motore Ajax (Ajax engine),

atto a gestire tale comunicazione. Un motore Ajax è in realtà un semplice oggetto o funzione JavaScript che viene chiamato ogni qual volta è necessario richiedere informazioni al server. A differenza del modello tradizionale, che fornisce un collegamento a un'altra risorsa (per esempio un'altra pagina web), ogni collegamento esegue una chiamata al motore Ajax, che programma ed esegue la richiesta. Essa viene eseguita in modo asincrono, ovvero l'esecuzione del codice non attende una risposta prima di continuare.
Il server, che solitamente servirebbe HTML, immagini, CSS o JavaScript, viene configurato per inviare dati che il motore Ajax può utilizzare. Questi dati possono essere testo puro, XML, o qualsiasi altro formato di informazioni necessario. L'unico requisito è che il motore Ajax possa comprenderlo e interpretarlo.
Quando il motore Ajax riceve la risposta del server, passa all'azione, spesso esaminando i dati e approtando svariati cambiamenti all'interfaccia utente basandosi sulle informazioni ricevute. Dato che questo processo comporta il trasferimento di una minore quantità di informazioni rispetto al tradizionale modello di applicazione web, l'interfaccia utente viene aggiornata più velocemente, e l'utente è in grado di svolgere il proprio lavoro con maggiore efficienza. La Figura 1.1 è un adattamento della figura presente nell'articolo di Garrett che mostra le differenze fra il modello di applicazione web tradizionale e quello di Ajax.

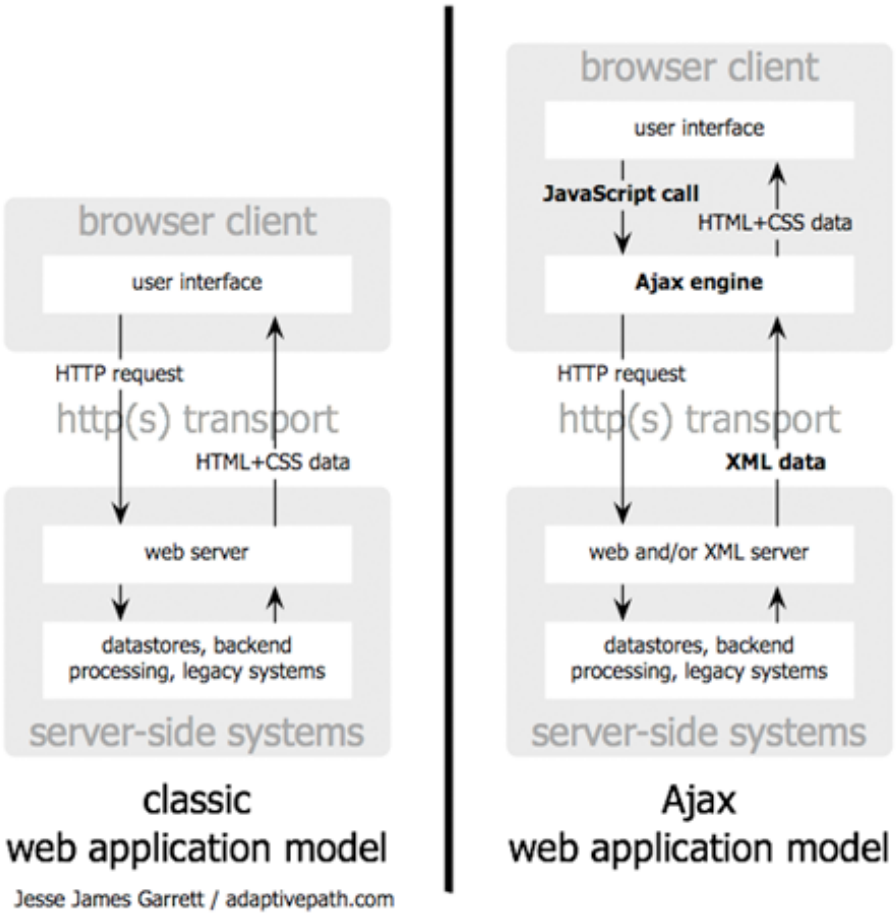

Figura 1.1

Molti sviluppatori web hanno considerato questo nuovo approccio come una sfida. La sfida è distinguere ciò che rende ottima un'applicazione web Ajax da ciò che la rende pessima o mediocre. Michael Mahemoff (www.mahemoff.com), sviluppatore di software ed esperto di usabilità, ha identificato svariati principi chiave di ottime applicazioni Ajax; vale la pena ripeterli qui:

- Traffico minimo:
- Niente sorprese:

- Convenzioni prestabilite:
- Nessuna distrazione:
- Accessibilità:
- Eveitare interi caricamenti di pagina:
- L'utente innanzi tutto:

Il comune denominatore di tutti questi principi è l'usabilità. Lo scopo primario di Ajax è migliorare l'esperienza Web dei propri utenti; la tecnologia che lo permette è semplicemente un mezzo per tale fine. Attenendosi ai principi qui elencati, si può essere ragionevolmente sicuri che la propria applicazione Ajax risulterà utile e usabile.
L'articolo di Garrett cita diverse tecnologie che egli vede come parti di una soluzione Ajax:

- HTML/XHTML: linguaggi di rappresentazione del contenuto primario
- CSS: fornisce formattazione stilistica allo XHTML
- DOM: aggiornamento automatico di una pagina caricata
- XML o JSON: formato di interscambio dati
- XSLT: trasforma XML in XHTML (con stili forniti dai CSS)
- XMLHttp: agente di comunicazione primario
- JavaScript: linguaggio di scripting utilizzato per programmare un motore Ajax

In realtà, tutte queste tecnologie possono essere sfruttate in soluzioni Ajax, ma soltanto tre sono fondamentali: HTML/XHTML, DOM e JavaScript. XHTML è ovviamente necessario per la visualizzazione delle informazioni, mentre DOM è richiesto per cambiare porzioni di una pagina XHTML senza ricaricarla. L'ultimo componente, JavaScript, è indispensabile per avviare la comunicazione client-server e manipolare il DOM per aggiornare la pagina web. Le altre tecnologie dell'elenco precedente sono utili per affinare una soluzione Ajax, ma non sono necessarie.
Vi è un componente importante che Garrett ha dimenticato di citare nel suo articolo: la necessità dell'elaborazione lato server. Tutte le tecnologie summenzionate si relazionano direttamente al motore Ajax lato client, ma non vi è Ajax senza un server stabile e reattivo in attesa di inviare contenuti al motore. Per questo scopo è possibile utilizzare l'applicazione server che si desidera. Che si scelga di scrivere i componenti lato server come pagine PHP, servlet Java o componenti .NET, occorre solo assicurarsi che venga inviato in risposta al motore Ajax il formato di dati corretto.

2.4 Framework e piattaforme per lo sviluppo

Nella produzione del software, il framework è una struttura di supporto su cui un software può essere organizzato e progettato. Alla base di un framework c'è sempre una serie di librerie di codice utilizzabili con uno o più linguaggi di programmazione, spesso corredate da una serie di strumenti di supporto allo sviluppo del software, come ad esempio un IDE, un debugger, o altri strumenti ideati per aumentare la velocità di sviluppo del prodotto finito. Lo scopo di un framework è di abbattere i tempi di sviluppo, evitando di

riscrivere funzionalità già implementate in precedenza per risolvere problemi comuni in quel particolare settore. Questa esigenza è diventata sempre più evidente quando le interfacce utente sono diventate più complesse, o più in generale quando è aumentata la quantità di software con funzionalità secondarie simili. Nei casi in cui un framework permette di aggiungere funzionalità con poche righe di codice a carico del programmatore, o magari permettendogli di disegnare l'interfaccia in un ambiente di sviluppo, esso permetterà al programmatore di concentrarsi sulle vere funzionalità dell'applicazione, senza doversi far carico di scrivere codice "di contorno". Il termine inglese framework quindi può essere tradotto come intelaiatura o struttura, che è appunto la sua funzione, a sottolineare che al programmatore rimane solo da creare il contenuto vero e proprio dell'applicazione, ovvero le specifiche funzionali di un progetto che utilizza come fondamenta un software che risolve problematiche note. Un framework è definito da un insieme di classi astratte e dalle relazioni tra esse. Istanziare un framework significa fornire un'implementazione delle classi astratte.

2.5 Il pattern MVC (Model View Controller)

In ingegneria del software MVC (Model View Controller) è un pattern architetturale molto diffuso nello sviluppo di interfacce grafiche di sistemi software object-oriented.
Originariamente impiegato dal linguaggio Smalltalk, il pattern è stato esplicitamente o implicitamente sposato da numerose tecnologie moderne, come i framework basati su PHP (Symfony, Zend Framework, CakePHP, Yii Framework), su Ruby (Ruby on Rails), su Python (Django, TurboGears, Pylons, Web2py, Zope), su Java (Swing, JSF, Struts), su Objective C o su .NET. A causa della crescente diffusione di tecnologie basate su MVC nel contesto di framework o piattaforme middleware per applicazioni Web, l'espressione framework MVC è diventata di largo utilizzo anche per indicare in modo specifico questa categoria di sistemi.
Il pattern è basato sulla separazione dei compiti fra i componenti software che interpretano tre ruoli principali:

- il model fornisce i metodi per accedere ai dati utili all'applicazione
- il view visualizza i dati contenuti nel model e si occupa dell'interazione con utenti e agenti
- il controller riceve i comandi dell'utente e li attua modificando lo stato degli altri due componenti

Questo schema implica anche la tradizionale separazione tra logica applicativa (spesso chiamata "logica di business"), a carico del controller e del model, e l'interfaccia utente, a carico del view.
Il pattern MVC è stato originariamente sviluppato per applicazioni desktop ma è stato successivamente riadattato come architettura per realizzare web application. Per migliorare il pattern MVC sono stati sviluppati, nel tempo, diversi framework commerciali e non commerciali molti dei quali variano nell'interpretazione del pattern dividendo le responsabilità dei vari componenti tra client e server.
Inizialmente, l'approccio delle applicazioni web che utilizzavano il pattern MVC era quello di posizionare il model, il view e il controller esclusivamente lato server. In questo caso, il client inviava l'input di un form o richieste di URL al controller e riceveva una pagina completamente nuova e aggiornata dal view. Nell'evoluzione delle tecnologie web, e con l'avvento di AJAX, sono nati alcuni framework, ad esempio JavaScriptMVC (http://javascriptmvc.com) e Backbone (http://backbonejs.org), che permettono l'esecuzione di alcune componenti direttamente sul client.
La tripletta di classi Model/View/Controller fu descritta inizialmente nel 1988 da Krasner e Pope ed era impiegata per costruire interfacce utente in Smalltalk-80. Il pattern MVC consiste di tre oggetti.
Il Model rappresenta l'oggetto applicazione, il View è la presentazione a video e il Controller definisce il modo in cui l'interfaccia utente reagisce all'input dell'utente. Prima dell'MVC le interfacce utente erano progettate combinando e fondendo insieme le tre componenti, senza operare distinzioni.

MVC scompone i view e i model utilizzando un protocollo di iscrizione/notifica (subscribe/notify) tra di essi. Una view deve assicurarsi che l'aspetto dell'applicazione rifletta lo stato del model. Ogni qualvolta che i dati nel model cambiano, questo notifica le view che dipendono da esso, in questo modo ogni view può modificare l'aspetto in funzione dello stato del model aggiornato. Questo approccio permette di associare più istanze view ad un solo model, al fine di fornire diversi tipi di presentazione del contenuto. Questo tipo di comportamento può essere descritto dal pattern Observer.
Il pattern Observer è un pattern di tipo comportamentale (Object Behavioral Pattern) il cui intento è quello di definire dipendenze uno a molti tra gli oggetti. Quando un oggetto cambia il proprio stato, tutti gli oggetti dipendenti sono notificati e aggiornati automaticamente.
Un'altra caratteristica dell'MVC consiste nella possibilità di nidificare le view. Ad esempio un pannello di controllo costituito da bottoni può essere implementato costruendo una view complessa che al suo interno contiene altre view che rappresentano i bottoni. MVC supporta questo approccio con la classe CompositeView, una sottoclasse di View. Una CompositeView si comporta esattamente come un oggetto View. Essa può essere utilizzata ogniqualvolta si voglia utilizzare una View, ma supporta la nidificazione. Anche questo tipo di approccio si può generalizzare con il pattern Composite.
Il pattern Composite è un pattern di tipo strutturale, il cui intento è quello di comporre oggetti in strutture ad albero modellando gerarchie all'interno della struttura concettuale dell'applicazione. Composite permette di trattare oggetti singoli e composizioni di oggetti allo stesso modo.
In applicazioni di grafica come editor di disegno, l'utente può costruire strutture complesse partendo da componenti semplici. L'utente può raggruppare componenti per formare strutture più ampie, che a loro volta possono essere raggruppate per formare altre strutture. Una semplice implementazione può definire classi per primitive grafiche come il testo o le righe con l'aggiunta di altre classi che fungono da contenitori per queste primitive.
MVC permette inoltre di cambiare il modo in cui le view rispondono all'input dell'utente modificando la presentazione. MVC incapsula il meccanismo di risposta in un oggetto Controller. Esiste una gerarchia di classi di controller che rende facile la creazione di un nuovo controller come variante di uno esistente. Una view utilizza un'istanza di una sottoclasse di Controller per implementare una particolare strategia di risposta. Per implementare una nuova strategia è sufficiente rimpiazzare l'istanza con quella di un controller differente. La relazione View-Controller è un esempio del design pattern Strategy.
Il pattern Strategy, come per il Composite, è un pattern di tipo comportamentale il cui intento è quello di definire famiglie di algoritmi, incapsulando ognuno, e renderli interscambiabili. Questo pattern permette all'applicazione di cambiare il comportamento in funzione dell'algoritmo utilizzato indipendentemente dall'utilizzo esterno.
Il pattern MVC si basa quindi su tre pattern fondamentali che costituiscono la tripletta Observer, Composite e Strategy, ma si avvale anche di altri tipi di pattern molto diffusi, Factory Method e Decorator.
Il pattern Factory Method è un pattern per la creazione di classi (Class Creational) il cui intento è di definire un'interfaccia per la creazione di oggetti, lasciando libertà alle sottoclassi di decidere quale classe istanziare. Questo tipo di comportamento permette alle classi di rinviare l'inizializzazione alle sottoclassi. Il pattern Decorator è un pattern di tipo strutturale (Object Structural) il cui intento è quello di aggiungere responsabilità, o funzionalità, aggiuntive ad un oggetto, in modo dinamico. Le classi decorator forniscono un'alternativa flessibile all'approccio di estensione di classi per estendere le funzionalità di un'applicazione.

2.6 Scelta della piattaforma CMS (Content Management System)

Un Content Management System, CMS, è uno strumento software, installato su un web server, il cui compito è facilitare la gestione dei contenuti di siti web, svincolando l'amministratore da conoscenze tecniche di programmazione Web.
Esistono CMS specializzati, cioè appositamente progettati per un tipo preciso di contenuti (blog, forum, riviste, enciclopedie online) e CMS generici, che tendono a essere più flessibili per consentire la pubblicazione di tipi di contenuti diversi.

Tecnicamente, un CMS è un'applicazione lato server che si appoggia su un database preesistente per memorizzare i contenuti. Il CMS consiste di due parti:

- una sezione di amministrazione, chiamata back-end, che serve ad organizzare e supervisionare la produzione dei contenuti
- una sezione applicativa, chiamata front-end, che l'utente web usa per fruire dei contenuti e delle applicazioni del sito. L'amministratore del CMS gestisce dal proprio terminale, tramite un pannello di controllo, i contenuti da inserire o modificare

I CMS possono essere realizzati utilizzando vari linguaggi per lo sviluppo web tra cui, i più comuni, PHP, ASP e Microsoft .NET. Il tipo di linguaggio adoperato è slegato rispetto alle funzionalità.
In un approccio orientato al problema della gestione dell'informazione si affrontano le seguenti fasi:

- identificazione degli utenti di back-end e dei relativi ruoli di produzione o fruizione dell'informazione, di controllo e coordinamento
- assegnazione di responsabilità e permessi a differenti categorie di utenti per distinti tipi di contenuti (in un progetto complesso il prodotto finito non è frutto del lavoro del singolo, che pertanto non ha la possibilità o esigenza di intervenire in tutti gli ambiti)
- definizione delle attività di flusso di lavoro, cioè formalizzazione di un percorso per l'assemblaggio del prodotto finale che, in quanto frutto di produzione frammentaria, deve acquisire la sua unitarietà sottostando a opportune procedure di supervisione. Per poter rendere efficiente la comunicazione tra i vari livelli della gerarchia, è necessaria un'infrastruttura di messagistica, con la quale i gestori del contenuto possono ricevere notifica degli avvenuti aggiornamenti
- tracciamento e gestione delle versioni del contenuto
- pubblicazione del contenuto

L'utilizzo più diffuso dei CMS è rivolto alla gestione di siti web, soprattutto se sono di grandi dimensioni e richiedono un frequente aggiornamento. Una delle applicazioni più utili dei sistemi CMS, infatti, è nella gestione dei portali, dove vengono impiegati come strumento di pubblicazione flessibile e multiutente. Ad esempio la gestione di contenuti testuali (notizie, articoli, ecc...), link, immagini, forum, materiale scaricabile. Può essere modificata la struttura stessa delle pagine in numero ed organizzazione. A volte i CMS permettono la gestione di più versioni dello stesso sito (HTML e Mobile). I CMS consentono di definire utenti, gruppi e diritti in modo da poter permettere una distribuzione del lavoro tra più persone. Per esempio, è possibile definire una classe di utenti abilitati esclusivamente all'inserimento delle notizie, mentre si può riservare la scrittura degli articoli ad un altro gruppo.
Un CMS permette di costruire e aggiornare un sito dinamico, anche molto grande, senza necessità di scrivere una riga di HTML e senza conoscere linguaggi di programmazione lato server (come PHP) o progettare un apposito database. L'aspetto esteriore delle pagine può essere personalizzato scegliendo un foglio di stile CSS appositamente progettato per un determinato CMS. Tuttavia, un CMS è tanto più efficiente quanto più è specializzato. I portali più piccoli fanno ricorso a CMS di tipo generico (messi a disposizione gratuitamente o a pagamento). Per quanto un CMS possa essere flessibile, un sito basato su questa struttura in genere presenta un aspetto poco personalizzato se non è possibile intervenire direttamente sul codice sorgente del prodotto per modificarlo. Analogamente i contenuti saranno sempre ancorati a quanto previsto da chi ha progettato il CMS e non alle esigenze di chi pubblica il sito. Inoltre, alcuni problemi di gestione possono sorgere quando chi pubblica o gestisce il sito può usare il CMS per intervenire sui contenuti o sull'aspetto,

ma generalmente non è in grado di intervenire direttamente sulla struttura del CMS stesso (nel caso di software proprietario). Questo è un limite strettamente legato al vantaggio principale di un CMS: pubblicare un portale senza doverne progettare la struttura o senza possederne le conoscenze tecniche (o le risorse finanziarie) per uno sviluppo personalizzato. Questi problemi sono tuttavia risolvibili utilizzando software open source: la possibilità di accedere al codice sorgente del prodotto permette di personalizzare il software sulla base delle proprie esigenze a patto di non avere necessità di apportare modifiche al prodotto adottato. Anche in questo caso, vanno messi in conto i costi per lo sviluppo di moduli (o plugin) personalizzati o funzionalità particolari a meno di non possedere in proprio o nella propria struttura aziendale le conoscenze tecniche per intervenire nel codice sorgente.
Di seguito, confrontiamo le tre piattaforme più conosciute e utilizzate in ambito web mettendo a confronto le caratteristiche che determinano la scelta finale. Wordpress è la piattaforma scelta per la realizzazione del progetto Matchmed poiché offre versatilità, usabilità e affidabilità. E' efficiente e utilizzata da circa il 14% dei siti web esistenti.
Nella Tabella 1 mettiamo a confronto tre piattaforme CMS evidenziandone la storia e l'impatto sui social media. Nella Tabella 2 mostriamo le caratteristiche principali che hanno indirizzato la scelta sulla piattaforma Wordpress.

Tabella 1	Wordpress	Drupal	Joomla
Data di rilascio	27/05/2003	15/01/2001	16/09/2005
Storia	Fork da b2/cafelog, una piattaforma creata da Matt Mullenweg	Avviato da Dries Buytaert come una community	Fork di Mambo nel 17/08/2005
Versioni	3	7	6
Media delle visite mensili (U.S.)	50 milioni	55.700	59.600
Siti che utilizzano la piattaforma	TIME - http://www.time.com/time/ WSJ.com - http://europe.wsj.com/home-page MTV Newsroom - http://newsroom.mtv.com The Ford Story - http://social.ford.com NPQ - http://www.napoliperquartiere.it	The White House - http://www.whitehouse.gov The Economist - http://www.economist.com Mtv - http://www.mtv.com	Linux.com - http://www.linux.com iHop - http://www.ihop.com Porsche - http://www.porsche.com
Numero di follower su Twitter	229.895	38.954	41.049
Media di tweet al giorno	0.57	0.82	2.6

Tabella 1	Wordpress	Drupal	Joomla
Fan su Facebook	519.043	42.712	102.431
Media dei costi di configurazione e personalizzazione	$250 - $15.000	$5.000 - $50.000	$2.000 - $20.000
Media dei costi di manutenzione mensile	$250	$1.500	$500
Ricerche su Google mensili	30.4 milioni	5 milioni	11.1 milioni
Costo Google AdWords per click	$2.18	$4.58	$2.53
Facilità di installazione	facile	difficile	medio
A chi si rivolge	non esperti	esperti	intermedio

Wordpress

"WordPress is web software you can use to create a beautiful website or blog. We like to say that WordPress is both free and priceless at the same time." [wordpress.org]
Citando la pagina originale: "Wordpress è nato dal desiderio di un sistema di pubblicazione personale, elegante e ben strutturato, costruito attorno a PHP e MySQL e distribuito sotto licenza GPL. E' il successore ufficiale di b2/cafelog. Wordpress è un software nuovo ma le cui radici ed il cui sviluppo risalgono al 2001. Si tratta di un prodotto stabile e maturo. Speriamo, focalizzandoci sui web standard e sull'interfaccia utente, di poter creare uno strumento differente da qualsiasi altro esistente."
Wordpress è un software di publishing focalizzato sulla semplicità di utilizzo, sulla velocità e su una ottima interfaccia utente. Wordpress gode anche di una ampia e attiva comunità, che è il cuore di qualsiasi software open source.
La documentazione prodotta e il codice stesso, sono stati realizzati da e per la comunità. WordPress è un progetto open source, centinaia di persone in tutto il mondo contribuiscono alla sua crescita ogni giorno. E' possibile utilizzare la piattaforma gratuitamente per qualsiasi tipo di progetto. WordPress è completamente personalizzabile.
Nasce come piattaforma di blogging, ma si è evoluto per essere utilizzato come CMS grazie all'implementazione di migliaia di plugins, widgets e layout grafici.

Joomla

"Joomla is an award-winning content management system (CMS), which enables you to build Web sites and powerful online applications" [http://www.joomla.org]
Joomla si propone come progetto collaborativo e non come un prodotto software. Nasce nel settembre 2005, da un gruppo di sviluppatori volontari, come fork dal codice del CMS Mambo. Joomla è un software CMS, realizzato completamente in linguaggio PHP. E' pubblicato con licenza open source GNU GPL.

Drupal

"Use Drupal to build everything from personal blogs to enterprise applications" [http://drupal.org]
Drupal è un pacchetto software gratuito per la semplificazione, la gestione e la pubblicazione di contenuti, con grande varietà nella personalizzazione. E' un software open-source supportato da una grande comunità di sviluppatori. E' distribuito sotto licenza GNU (General Public License) o GPL. I principi di sviluppo del progetto Drupal incoraggiano la modularità, gli standard, la collaborazione e la facilità di utilizzo. Il progetto Drupal inizia nel 1999 quando Dries Buytaert sviluppa un software di messaggistica online. Dopo circa un anno, grazie all'interesse suscitato, altri sviluppatori contribuirono al progetto e il software fu reso open-source.

Tabella 2	Wordpress	Drupal	Joomla
Aggiornamenti	164	77	dalla versione 1.5 e 1.6: 27 (incluse le beta)
Freq. di aggiornamento (media)	Una volta ogni 17.8 giorni	Dalla versione 4.3: una volta ogni 36 giorni	Versione 1.5: ogni 49 giorni Versione 1.6: ogni 25 giorni
Numero di plugin/ moduli/estensioni	14.629	8.039	7.608
Numero di temi	1.392	885	-
Numero di siti che utilizzano la piattaforma	14.3%	1.6%	2.7%
Costo Google AdWords per click	$2.18	$4.58	$2.53
Potenza SEO nativa	5/5	4/5	4/5
Facilità di installazione	facile	difficile	medio

Fonti: wordpress.org, drupal.org, joomla.org, mashable.com, adwords.google.com, quanocast.com, indeed.com, w3techs.com, socialtechnologyreview.com, deviousmedia.com

Per confrontare le tre piattaforme abbiamo effettuato i test utilizzando Siege (http://www.joedog.org/siege-home/), un tool per il benchmarking e il load testing. Il test è stato effettuato su installazioni standard delle tre piattaforme Wordpress, Drupal e Joomla. Al fine di testare le performance in modo equilibrato, utilizziamo installazioni basiche dei CMS. Installiamo le diverse piattaforme e un set di dati demo per avere un sito perfettamente funzionante. L'installazione e la configurazione dei dati dimostrativi è stata effettuata con le configurazioni automatiche disponibili nei pacchetti standard scaricati dai portali delle piattaforme scelte. Utilizzando uno standard report di Siege per verificare le prestazioni delle tre piattaforme, otteniamo, ad alto livello e tralasciando alcune variabili, una stima utile per la scelta della piattaforma che meglio si adatta alle esigenze Matchmed. Tutte le piattaforme si basano su PHP e database MySQL, nei test effettuati

utilizziamo MySQL 5.1 su Fedora 14 su Intel i7 i960 con 24Gb di RAM e un hard disk a stato solid (SSD). Lanciamo il test con il comando: siege -c20 localhost/app -b -t30s. I risultati sono mostrati nella Tabella 3.

CMS	Transactions	Data Transfer	Response time	Transaction rate	Longest Transaction
Joomla 1.7.0	2848	23.16MB	.20s	97/s	1.2s
Drupal 7.7	3426	13.09MB	.17s	117/s	.82s
Wordpress 3.2.1	3284	10.01MB	.18s	112/s	.55s

Tabella 3 - Fonte: http://imperialwicket.com/zotonic-cms-performance-comparison

2.6.1 Integrazione con le specifiche Matchmed e Plugin

“Plugins can extend WordPress to do almost anything you can imagine.” [http://wordpress.org/extend/plugins/].
Un plugin è un programma non autonomo che interagisce con un altro programma per ampliarne le funzioni. La capacità di un software di supportare i plugin è un’ottima caratteristica, perché rende possibile l’ampliamento e la personalizzazione delle sue funzioni da parte di terzi, in maniera relativamente semplice e veloce. Ciò favorisce da un lato il minor invecchiamento del software e dall’altro una maggiore diffusione, quanto più sono numerosi e funzionali i plugin esistenti per uno specifico programma. Per facilitare il compito degli sviluppatori di terze parti che intendono realizzare dei plugin, l’azienda produttrice del software e ideatrice dello standard distribuisce dei sistemi detti Kit di Sviluppo Software (SDK o Software Development Kit), che racchiudono funzionalità, esempi e documentazione per lo sviluppatore.
Per la natura del progetto Matchmed, è stato necessario introdurre nuove tassonomie in Wordpress, estendendo quelle native come i post, le pagine e le categorie. A questo proposito, abbiamo scelto il plugin “GD Custom Posts And Taxonomies Tools” [http://wordpress.org/extend/plugins/gd-taxonomies-tools/] per estendere le tassonomie esistenti e i tipi di post supportati. Inoltre, fornisce una serie di strumenti per la gestione di tipi di post personalizzati e di tassonomie e widget per la term cloud relativa agli stessi. Come da specifiche di progetto, Matchmed cataloga reperti sul territorio della provincia di Napoli. Con GD Custom Posts and Taxonomies abbiamo introdotto il tipo di post “Reperto”.
In Matchmed, ogni reperto possiede informazioni aggiuntive di dettaglio non previste da Wordpress nativamente. Il plugin utilizzato per l’introduzione e la gestione di informazioni aggiuntive è “verve meta boxes” [https://github.com/avenueverve/verve-meta-boxes].
Verve meta boxes è un plugin robusto e intuitivo per la gestione di campi aggiuntivi. E’ dotato di un’interfaccia utente per la creazione di campi come i text, textarea, image, file, date, time, datetime, select, radio e checkbox da poter associare a post o pagine. I valori dei campi aggiuntivi sono memorizzati nella tabella wp_postmeta del database di Wordpress e possono essere recuperati utilizzando le funzioni native: the_meta(), get_post_meta(), get_post_custom(), get_post_custom_values(), ecc...
Una volta installato, verve meta boxes inserisce una voce nel menu principale dell’amministrazione di Wordpress. Nella nuova sezione è possibile gestire molteplici box di campi aggiuntivi con la possibilità di ordinarli arbitrariamente con drag & drop.
La pagina di visualizzazione dei reperti di Matchmed utilizza le API di Google Maps per la visualizzazione dei reperti catalogati e memorizzati nel database. L’interfaccia consiste del layout Matchmed con una Google Map che visualizza un vettore di reperti geolocalizzati. E’ possibile effettuare ricerche e filtrare i risultati interagendo con i tasti nel form di interazione presente sulla pagina di visualizzazione. Sono state definite due classi di categorie, utilizzate poi in fase di filtraggio, per catalogare i reperti in Matchmed. La prima

classe rappresenta la “condizione” di un reperto (fuori terra, inglobato, interrato, sommerso, ecc...) e la seconda rappresenta le “tipologie” (acquedotto, ambiente termale, anfiteatro, cripta, ecc...). Selezionando uno di questi filtri, il sistema effettua una richiesta Ajax al server con i parametri selezionati. Il server recupera la lista di reperti che rispettano i criteri di ricerca, recupera le informazioni relative agli stessi e risponde con i risultati ottenuti. L’interfaccia, e la parte client-side, di Matchmed visualizzano sulla mappa la lista di reperti ottenuta dal server. Per realizzare questa specifica funzionale, abbiamo creato classi PHP apposite per la gestione di una richiesta parametrizzata.
Un’altra esigenza del gruppo di lavoro Matchmed è quella di poter esportare in formato Excel tutte le informazioni relative ai reperti. A questo proposito abbiamo realizzato uno script che si basa sulla ricerca e il filtraggio di reperti. Una volta ottenuta la lista dei risultati, è possibile esportarli in formato Excel. Il sistema formatta automaticamente i dati generando un file .xls.
Esaminiamo in dettaglio nella sezione 3.4 i plugin installati e le implementazioni effettuate per rispondere alle specifiche funzionali definite nel progetto Matchmed.

2.6.2 Caching

La cache è una memoria temporanea, non visibile al software, che memorizza un insieme di dati che possono successivamente essere velocemente recuperati su richiesta.
Nelle applicazioni web, l’operazione di caching è la memorizzazione temporanea di contenuti, come pagine HTML, immagini, file e oggetti web, al fine di ridurre il carico di richieste verso un server web e ridurre i tempi richiesti per l’invio dei contenuti al client. Esistono due forme principali di caching: forward-proxy e reverse-proxy.
Una cache di tipo forward-proxy è tipicamente distribuita in vari punti della rete, negli ISP (Internet Service Provider) o come parte di CDN (Content Delivery Network), lontano dal server originario.
Una cache di tipo reverse-proxy invece, risiede vicino a uno o più server web e filtra le richieste, da parte dei client, al server originario. Per la posizione che assume il reverse-proxy nei datacenter, è logico che la funzionalità di caching sia integrata con tecnologie di load balancing.
Il load balancing è un metodo per distribuire il carico applicativo su più server o su un cluster al fine di ottimizzare l’utilizzo delle risorse, massimizzare il throughput, minimizzare i tempi di risposta e evitare il sovraccarico. L’utilizzo di componenti multiple nel load balancing incrementa inoltre l’affidabilità del sistema aumentandone la ridondanza. Il caching è considerato una funzionalità chiave nelle prossime generazioni di load balancers, comunemente chiamate Application Delivery Controllers (ADCs).
Un reverse-proxy agisce come proxy server. Per un client, il reverse proxy appare come un normale proxy server. Con questa configurazione, il server proxy è attivo con un IP raggiungibile via internet. I client sono indirizzati al proxy server tramite la risoluzione DNS del dominio e i contenuti sono forniti direttamente dalla cache del server se già presenti in storage (quando si verifica un “cache hit”). Se il contenuto non è ancora memorizzato in cache, si verifica un “cache miss”, la richiesta è reindirizzata al server di origine per ottenere una copia dei contenuti che viene distribuita ai client e memorizzata in cache per prossime eventuali richieste. Quando si ricevono richieste successive per lo stesso contenuto, questo sarà servito dal cache server.
E’ possibile memorizzare in cache sia il contenuto statico, sia quello dinamico. Possono essere adottate politiche per assicurare che il contenuto dinamico, memorizzato in cache, sia sempre aggiornato e validato al fine di prevenire la distribuzione di contenuti datati.
I principali vantaggi del caching sono l’alleggerimento del carico sul server originario e la distribuzione accelerata di contenuti.

Alleggerire il carico:

- Ridurre il carico per richieste frequenti

- Riservare spazio sul server per contenuti non compatibili con la cache e altre operazioni
- Risparmiare sui costi di attività del server ottimizzandone l'utilizzo

Distribuzione del contenuto accelerato:

- Ridurre i tempi di risposta recuperando i contenuti richiesti dalla memoria cache
- Eliminare la necessità di recuperare e assemblare i contenuti dal web, database, file e server di immagini
- Offrire una migliore esperienza d'uso a più client
- Traffic shaping (si intende l'insieme di operazioni di controllo sul traffico di una rete dati finalizzate a ottimizzare o a garantire le prestazioni di trasmissione, ridurre o controllare i tempi di latenza e sfruttare al meglio la banda disponibile tramite l'accodamento e il ritardo dei pacchetti che soddisfanno determinati criteri [Wikipedia]).

Molti software per il reverse proxy includono l'abilitazione del traffic shaping delle richieste http in entrata. Questa funzionalità può risultare molto utile per i redirect o i rewrite, fortemente utilizzati in wordpress, o per migliorare le funzionalità di caching di applicazioni web non orientate in questo senso.
Generalmente, la gran parte dei contenuti statici è pronta per essere memorizzata in cache. Per definizione, il contenuto statico (ad esempio una pagina web) è fornito al client esattamente come viene memorizzato. Di contrasto, le pagine web dinamiche sono generate da web application. Il contenuto dinamico è quello generato con informazioni recenti (i contenuti o il layout) per ogni singola richiesta. Non è statico perché il contenuto cambia con il tempo, l'user ID, l'interazione con l'utente o altre possibili combinazioni.
Il contenuto dinamico quindi non può sempre essere memorizzato in cache, ma può esserlo se parte di un'applicazione HTTP. I più recenti reverse-proxy utilizzato tecnologie di cache parametrizzata per determinare se il contenuto di una specifica applicazione può essere memorizzato in cache o servito direttamente all'utente. Il contenuto sarà aggiornato periodicamente utilizzando meccanismi di scadenza controllati da un amministratore che si assicurerà che il contenuto sia sempre recente.
Una cache reverse-proxy può anche comunicare con altri server di cache esterni il cui compito è quello di memorizzare quantità maggiori di contenuti. Nel caso in cui la memoria di cache interna ad un application delivery controller non sia sufficiente, è possibile impiegare la cache redirection per effettuare richieste di contenuto memorizzato in cache più grandi.
I server ADC sono in grado di offrire memoria cache fino a 2GB. Server ADC più moderni che operano su architetture a 64-bit sono in grado di offrire capacità di gran lunga maggiori, da 24GB e oltre. Memorie cache più grandi offrono nuove possibilità per il caching di contenuti richiesti con grande frequenza, come video, grafiche ad alta risoluzione, file di presentazione, CAD o altri tipi di file di grandi dimensioni.
Tra i software più efficienti per il caching troviamo Squid [http://www.squid-cache.org]. Squid è un caching proxy Web che supporta HTTP, HTTPS, FTP e altri tipi di protocolli. Riduce la banda impiegata nel trasferimento dei contenuti e migliora i tempi di risposta riutilizzando, tramite caching, le pagine web richieste con frequenza. Tra gli utilizzatori di Squid troviamo Wikimedia, la piattaforma Wiki utilizzata da Wikipedia.
"[The Squid systems] are currently running at a hit-rate of approximately 75%, effectively quadrupling the capacity of the Apache servers behind them. This is particularly noticeable when a large surge of traffic arrives directed to a particular page via a web link from another site, as the caching efficiency for that page will be nearly 100%." [Wikimedia Deployment Information].

3. Sviluppo

In questo capitolo, affronteremo gli aspetti tecnici del progetto e le scelte in termini di organizzazione del codice e dei componenti necessari per realizzare la piattaforma Matchmed.
Esamineremo nel dettaglio l'ambiente di sviluppo scelto per la realizzazione di una web application, sia dal punto di vista dello sviluppatore, sia dal punto di vista della messa in opera.
Mentre nel primo caso è necessario installare un ambiente di sviluppo locale sulle macchine degli sviluppatori che fanno parte del team, nel secondo caso è necessario progettare e acquistare un ambiente per il rilascio del software realizzato (produzione). L'ambiente di produzione deve soddisfare i requisiti minimi di compatibilità con l'applicazione implementata e deve garantire una buona efficienza e buone performance, al fine di garantire il corretto funzionamento del sito anche con carichi di richieste elevati. Forniremo inoltre diagrammi di classe, casi d'uso e altri schemi UML che descrivono in dettaglio il funzionamento tecnico dell'applicazione Matchmed. Infine, esamineremo alcune soluzioni scelte per implementare le specifiche funzionali richieste, riportando come esempi alcune parti del codice realizzato.

3.1 L'ambiente di sviluppo

L'IDE (Integrated Development Environment) scelto per la realizzazione del progetto Matchmed è Aptana (http://www.aptana.com), che fornisce un ambiente di sviluppo integrato per la creazione di siti e applicazioni per il Web. E' un software rilasciato con licenza open source (GNU General Public License): può essere scaricato gratuitamente e utilizzato liberamente anche per scopi commerciali. Esso presenta una qualità e una quantità di funzionalità tali da costituire un'alternativa valida a software commerciali di larga diffusione presso i professionisti del Web, come ad esempio Adobe Dreamweaver. Gli strumenti disponibili attraverso Aptana sono numerosi:

- digitare codice HTML, CSS e JavaScript in modo assistito, per la creazione di fogli di stile CSS vengono anche forniti dei suggerimenti dal programma stesso e per il markup HTML sono disponibili tutte le proprietà e gli elementi
- supporto per l'evidenziazione della sintassi dei codici tramite colorazione
- auto-completamento del codice in sede di sviluppo per velocizzare la digitazione del testo listato
- supporto tramite plug-in per vari linguaggi di sviluppo e programmazione tra cui anche Ruby on Rails, PHP e Python
- possibilità di creare Web application tramite un apposito plug-in per Adobe Air
- strumenti per la creazione e la gestione di progetti all'interno dei quali sviluppare pagine e applicazioni; inoltre include un sistema di visualizzazione per tenere traccia delle gerarchie e dei percorsi di file, cartelle e sottocartelle
- una grande disponibilità di libreria Ajax, tra cui anche il supporto per i framework jQuery, Prototype e Scriptaculous; sono inoltre disponibili plug-in aggiuntivi per l'utilizzo di soluzioni basate su Ajax come per esempio MooTools
- strumenti integrati per il debugging di applicazioni JavaScript grazie all'integrazione di Firebug con Firefox

- disponibilità di strumenti per l'interazione con i database relazionali, sia in locale che in remoto

- supporto completo per il trasferimento di file da locale a remoto e viceversa, Aptana può essere utilizzato anche come client FTP per l'upload dei propri progetti in rete

- "snippets" (frammenti di codice) pronti all'uso per l'inserimento veloce all'interno dei listati

Aptana è un software multi-piattaforma e può essere utilizzato sia come applicazione standalone sia come estensione dell'ambiente di sviluppo open source Eclipse.
Come già accennato in questo documento, per lo sviluppo di software web in locale è necessario l'utilizzo di un application server. Quest'ultimo è un software che fornisce l'infrastruttura e le funzionalità di supporto, sviluppo ed esecuzione di applicazioni nonché altri componenti server in un contesto distribuito. Si tratta di un complesso di servizi orientati alla realizzazione di applicazioni multilivello ed enterprise, con alto grado di complessità, spesso orientate per il web.
L'application server scelto per lo sviluppo del progetto Matchmed è MAMP che può riprodurre l'ambiente LAMP necessario. MAMP è un acronimo che si riferisce ad un set di programmi liberi comunemente utilizzati insieme per far girare un sito web dinamico sul sistema operativo Mac OS X della Apple. L'acronimo sta per Mac OS X, Apache, MySQL, PHP, Perl o anche Python. MAMP è una piattaforma open source costruita sul sistema operativo Mac OS X.
Sebbene originariamente solo Apache fosse integrato in Mac OS X, questa piattaforma è diventata popolare molto presto per via del basso costo e per l'onnipresenza dei suoi componenti. Inoltre, Mac OS X 10.5 (e successivi) include un motore PHP ed è predisposto a MySQL. Se questi componenti vengono utilizzati insieme, l'utilizzo concorrente degli stessi è un'ottima piattaforma tecnologica per i server applicativi. In generale, MAMP è utilizzato come piattaforma di sviluppo e generalmente non è utilizzata per ambienti di produzione. Chiaramente, le applicazioni sviluppate nella sandbox di sviluppo locale sono compatibili anche con l'ambiente di utilizzo finale.

3.1.1 Metodologie agili

"*Gli individui e le interazioni* più che i processi e gli strumenti.
Il software funzionante più che la documentazione esaustiva.
La collaborazione col cliente più che la negoziazione dei contratti.
Rispondere al cambiamento più che seguire un piano.

Ovvero, fermo restando il valore delle voci a destra, consideriamo più importanti le voci a sinistra."
(dal Manifesto per lo sviluppo agile di software, http://www.agilemanifesto.org/iso/it/).
Il termine *Metodologie Agili* fu coniato nel 2001 quando il Manifesto Agile è stato formulato. La gran parte dei metodi agili tentano di ridurre il rischio di fallimento sviluppando il software in finestre di tempo limitate chiamate iterazioni che, in genere, durano qualche settimana. Ogni iterazione rappresenta un progetto a sé stante. Esso deve contenere tutto ciò che è necessario per il rilascio di un incremento nelle funzionalità del software: la pianificazione, l'analisi dei requisiti, il progetto, l'implementazione, i test e la documentazione. Anche se il risultato delle singole iterazioni non ha sufficienti funzionalità da essere considerato completo, questo deve essere rilasciato e, nelle successive iterazioni, deve rispecchiare sempre più alle richieste del cliente. Alla fine di ogni iterazione il team deve rivalutare le priorità di progetto. L'obiettivo è la piena soddisfazione del cliente e non solo l'adempimento di un contratto. L'uso di queste metodologie, inoltre, serve ad abbattere i costi di sviluppo del software.
L'eXtreme Programming, comunemente chiamato XP, è forse la metodologia agile più famosa ponendosi lo scopo di rispondere ai principi del Manifesto dello Sviluppo Software Agile. Le singole pratiche che sono applicabili all'interno di una metodologia leggera sono decine e dipendono essenzialmente dalle necessità dell'azienda e dall'approccio del project manager. Nella scelta però bisogna tenere conto delle caratteristiche

di ogni pratica per i benefici che apporta e le conseguenze che comporta. Ad esempio, in eXtreme Programming, si integra alla mancanza di ogni forma di progettazione e documentazione con il forte coinvolgimento del cliente nello sviluppo e con la progettazione in coppia.
L'XP propone l'utilizzo di pratiche agili già affermate e suggerisce di usarle al massimo delle loro potenzialità, da qui il termine "extreme". Possiamo raggruppare in categorie molte delle pratiche più diffuse:

- *Automazione*: se l'obiettivo delle metodologie agili è concentrarsi sulla programmazione senza dedicarsi alle attività collaterali, allora possono essere eliminate o automatizzate. Nel caso della seconda soluzione è possibile eliminare la documentazione aumentando il testing, ma non è possibile eliminare entrambe.

- *Comunicazione stretta*: uno dei primi teorici delle metodologie agili, Alistar Cockburn, sostiene che questo sia il vero aspetto che renda *agile* una metodologia. Per comunicazione diretta si intende la comunicazione interpersonale, fra tutti gli attori del progetto, cliente compreso. Ciò serve ad avere una buona analisi dei requisiti ed una proficua collaborazione fra programmatori anche in un ambito di quasi totale assenza di documentazione.

- *Coinvolgimento del cliente*: esistono differenti gradi di coinvolgimento del cliente. Ad esempio, in XP il coinvolgimento è totale, il cliente partecipa alle riunioni settimanali dei programmatori. In altri casi, il cliente è coinvolto solo in una prima fase di progettazione. In altri casi il cliente partecipa indirettamente o solo nella fase di test finale.

- *Progettazione e documentazione*: sarebbe un errore pensare che le metodologie agili eliminino la progettazione e la documentazione. In effetti queste introducono un'iterazione nel ciclo di vita del progetto. Quanta progettazione fare e quanta documentazione realizzare, è una scelta che spetta a chi gestisce il progetto e spesso i teorici dell'Agile Alliance avvisano che è un errore trascurare o addirittura omettere queste due fasi.

- *Consegne frequenti*: effettuare rilasci frequenti di versioni intermedie del software permette di ottenere più risultati contemporaneamente. Si ricomincia l'iterazione avendo già a disposizione un blocco di codice funzionante in tutti gli aspetti, si offre al cliente un prodotto quasi funzionante in tempi brevi e lo si distrae da eventuali ritardi nella consegna del progetto completo. Il questo caso il cliente diventa a tutti gli effetti un tester poiché utilizzerà il software e riscontrerà eventuali anomalie. In questo modo, si ottengono dal cliente informazioni più precise sui requisiti e le specifiche funzionali che probabilmente non sarebbe riuscito ad esprimere senza avere a disposizione utilità e carenze del progetto.

- *Gerarchia*: la scelta di creare una struttura gerarchica nel team di sviluppo dipende in gran parte dall'approccio del project manager, tuttavia si ha una conseguenza da non sottovalutare effettuando questa scelta. Se si decide per una struttura gerarchica ad albero e frammentata si ottiene la possibilità di gestire un numero molto alto di programmatori e di lavorare a diversi aspetti del progetto parallelamente. Se si decide invece di non adottare una struttura gerarchica si avrà un team di sviluppo molto compatto e motivato, ma necessariamente piccolo in termini di numero di programmatori.

- *Pair programming*: lo sviluppo viene fatto da coppie di programmatori che si alternano alla tastiera.

- Refactoring: la ristrutturazione di parti di codice mantenendone invariato l'aspetto e il comportamento esterno.

- *Miglioramento della conoscenza*: nata con l'avvento della programmazione Object-Oriented, non è altro che la presa di coscienza della produzione di conoscenza che si fa in un'azienda quando si produce codice.

Questa conoscenza prodotta non deve andare perduta ed è per questo che si sfruttano spesso le altre pratiche, come la comunicazione stretta o la condivisione della proprietà del codice.

- *Reverse engineering*: ottenere, spesso in maniera automatica, la documentazione a partire dal codice prodotto. E' una delle pratiche più diffuse e controverse. Nel primo caso, perché permette un guadagno enorme in termini di tempo, ma controversa perché spesso la documentazione prodotto è inutilizzabile oppure è prodotta solo per una richiesta burocratica del cliente senza preoccuparsi del reale utilizzo che se ne può fare in seguito.

- *Semplicità*: un aspetto ereditato direttamente dalla programmazione Object-Oriented è la semplicità. Semplicità nel codice, nella programmazione, nella progettazione e nella modellazione, promuovendo una maggiore leggibilità dell'intero progetto ed una conseguente facilitazione nelle fasi di correzione e modifica.

- *Test*: pratica molto diffusa anche prima della nascita delle metodologie agili, ha prodotto una letteratura vastissima ed una serie di approcci come il Rapid Testing o il Pair Testing. Nell'ambito delle metodologie agili vengono spesso utilizzati insieme tre tipi di test differenti: i test funzionali, utilizzati per verificare che il software esegue ciò per cui è stato scritto, i test unitari (Unit Test), utilizzati per verificare che ogni pezzo di codice funzioni correttamente e i test indiretti effettuati dal cliente ogni volta che gli si consegna una versione.

- Controllo della versione (versioning): una delle conseguenze dirette dell'iterazione nella produzione è la necessità di introdurre un modello, un metodo, uno strumento, per il controllo versioni del software prodotto e rilasciato. Tra gli strumenti per il controllo di versione più diffusi troviamo SVN, CVS e gitHub.

Da un punto di vista concettuale possiamo rappresentare il ciclo di vita iterativo e incrementale, comune a tutte le diverse metodologie agili, nel modo seguente:

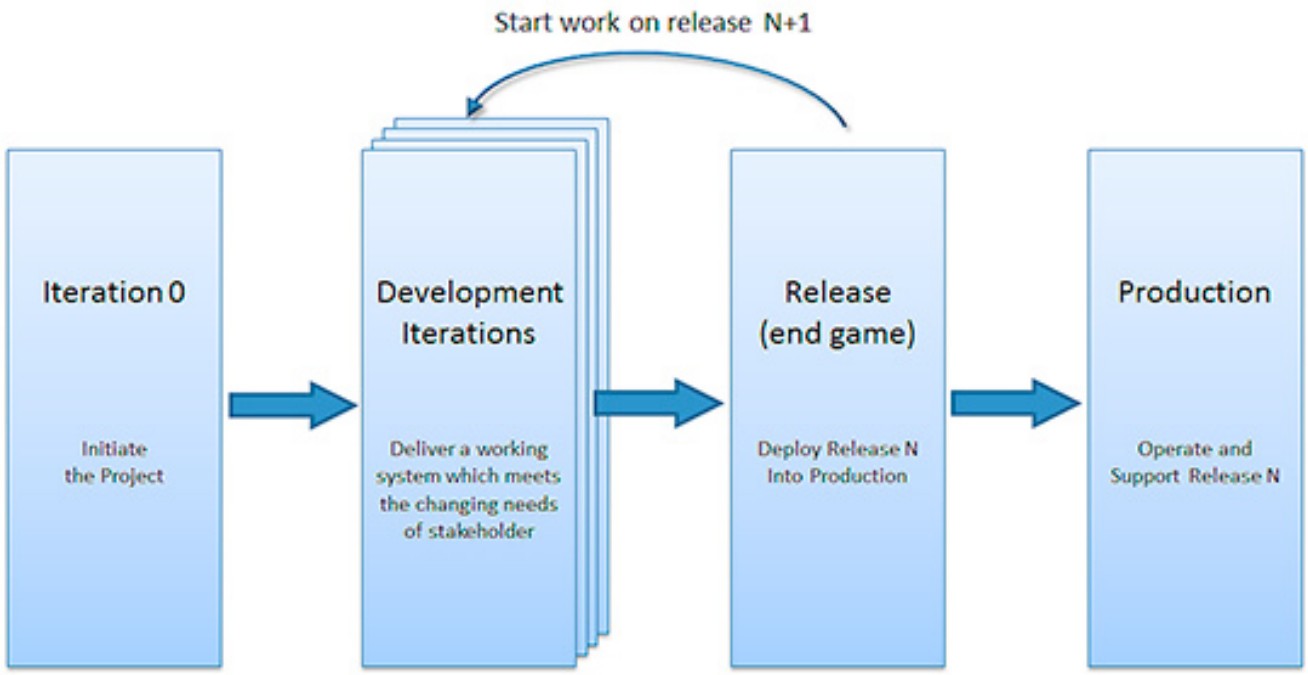

Figura 1.2 - Immagine gentilmente concessa da Fabio Armani, mokabyte.it

- Iteration 0 o Cycle 0.
- iterazioni di sviluppo, in cui si realizza il sistema in maniera incrementale.
- una o più iterazioni di rilascio relative alla release n-esima al termine della quale si può procedere allo sviluppo della release successiva.
- produzione

L'obiettivo della fase di produzione è quello di mantenere i sistemi utili e produttivi dopo che sono stati distribuiti agli utenti. Questa fase termina quando l'uscita di un sistema è stato previsto per "pensionamento" o quando il supporto per quella determinata versione è terminato; in pratica il sistema viene dismesso.

3.1.2 gitHub

"GitHub is the best place to share code with friends, co-workers, classmates, and complete strangers. Over three million people use GitHub to build amazing things together." (https://github.com/about).
gitHug.com fornisce funzionalità di social networking come i feed, i follower e grafici del network per avere un'idea di come gli sviluppatori lavorino sulle versioni e i repository. Esso basa il suo funzionamento su Git, un sistema di controllo versione e gestione del codice sorgente (SCM - source code managemente) che pone grande attenzione sull'efficienza e velocità. Git è stato progettato inizialmente da Linus Torvalds per lo sviluppo del kernel Linux. Ogni directory di lavoro in Git è a tutti gli effetti un repository con storico e capacità di tracciamento delle revisioni che non dipende da un server centrale. Git è un software libero ed è distribuito sotto i termini della GNU (General Public License version 2).
Lo sviluppo di Git è iniziato dopo che molti sviluppatori del kernel di Linux sono stati costretti ad abbandonare l'accesso ai sorgenti tramite il sistema proprietario BitKeeper. La possibilità di utilizzare BitKeeper gratuitamente era stata ritirata dal detentore dei diritti d'autore Larry McVoy dopo aver sostenuto che fosse stato effettuato il reverse engineering dei protocolli, violandone la licenza d'uso. Torvalds voleva un sistema distribuito che potesse usare come BitKeeper, ma nessuno dei sistemi disponibili gratuitamente soddisfaceva i suoi bisogni, particolarmente il suo bisogno di velocità. Tra i criteri di progettazione, Linus richiedeva che il nuovo sistema di versioning fosse esattamente l'opposto di CVS, che non era ben visto nel suo funzionamento. Inoltre, era necessario un flusso di lavoro distribuito, simile a BitKeeper, che non usasse un server centralizzato. Erano richieste inoltre la salvaguardia dei dati, sia accidentale che intenzionale e altissime prestazioni.
Nessuno dei sistemi gratuiti disponibili soddisfava questi requisiti e nell'aprile 2005 è iniziato lo sviluppo di Git.
Le primitive di Git non costituiscono un vero e proprio controllo di versione. Ad esempio, Git non fornisce una numerazione progressiva delle revisioni del software.
"In molti modi si può considerare git come un filesystem — è indirizzabile in base al contenuto, e include il concetto di versione, ma io in realtà l'ho progettato guardando il problema dal punto di vista di una persona esperta di filesystem (beh, i kernel sono quello che faccio), e effettivamente ho assolutamente zero interesse nel creare un sistema tradizionale di gestione della configurazione software." (Linus Torvalds).
Git ha due strutture dati, un indice modificabile che mantiene le informazioni sul contenuto della prossima revisione, e un database di oggetti a cui si può solo aggiungere e che contiene quattro tipi di oggetti:

- Un oggetto blob è il contenuto di un file. Git memorizza ogni revisione di un file come un oggetto blog distinto.
- Un oggetto albero è l'equivalente di una directory: contiene una lista di nomi di file, ognuno con alcuni bit di tipi e il nome di un oggetto blob o albero che è il file, il link simbolico, o il contenuto di directory.

- Un oggetto commit (revisione) collega gli oggetti albero in una cronologia. Contiene il nome di un oggetto albero, data e ora, un messaggio di archiviazione, e i nomi di zero o più oggetti di commit genitori.

- Un oggetto tag è un contenitore che contiene riferimenti a un altro oggetto può contenere meta-dati aggiuntivi riferiti a un altro oggetto. L'uso più comune è memorizzare una firma digitale di un oggetto commit corrispondente a un particolare rilascio dei dati gestiti da Git.

Git è pensato per funzionare in ambiente Linux, ma può essere portato su altri sistemi operativi Unix-like, tra cui BSD, Solaris e Darwin. Git è estremamente veloce su sistemi basati su POSIX.
Git può essere portato anche in ambiente Windows utilizzando l'ambiente Cygwin che è un'emulazione di POSIX.

3.2 Organizzazione del codice

Lo scopo di questo capitolo è delineare alcuni elementi strategici nel processo di sviluppo del software e i dettagli di implementazione dell'applicazione realizzata. Inizieremo con una breve panoramica sul linguaggio UML (Unified Modeling Language) e i suoi costrutti architetturali.
Presenteremo i diagrammi di classe e approfondiremo soluzioni e aspetti implementativi esaminando il codice sorgente.

3.2.1 UML

"Il linguaggi UML (Unified Modeling Language) è un linguaggio visuale di modellazione per scopi generali che è usato per specificare, costruire e documentare gli elaborati di un sistema software." (Rumbaugh et al., 1999). UML fu sviluppato dalla Rational Software Corp. per unificare le migliori caratteristiche delle notazioni e metodi esistenti e, nel 1997, il comitato OMG (Object Management Group) lo approvò come linguaggio standard di modellazione. Da allora, UML è stato ulteriormente sviluppato ed è ampiamente adottato nel settore IT.
UML è un linguaggio indipendente da qualsiasi processo di sviluppo del software. Un processo che adotta UML deve seguire un approccio object-oriented allo sviluppo del software. UML non è appropriato agli approcci strutturali di vecchio tipo che forniscono un supporto per sistemi implementati con i tradizionali linguaggi di programmazione procedurali, come il COBOL o il C.
UML è indipendente inoltre dalle tecnologie utilizzate per l'implementazione e per questo è talvolta carente come supporto alla fase di progetto dettagliato nel ciclo di sviluppo ma, per lo stesso motivo, ha il vantaggio di risultare piuttosto flessibile rispetto alle modifiche nelle piattaforme di implementazione.
I costrutti del linguaggio UML permettono la modellazione della struttura statica e del comportamento dinamico di un sistema. Questo è rappresentato come un insieme di oggetti che collaborano (componenti software) i quali reagiscono a eventi esterni per eseguire attività a beneficio dei utilizzatori finali (clienti). Alcuni modelli UML enfatizzano certi aspetti del sistema e ignorano aspetti che sono oggetto di studio per altri modelli. Insieme, questi modelli forniscono una descrizione completa del sistema e possono essere classificati in tre gruppi:

- modelli dello stato - descrivono le strutture statiche dei dati

- modelli del comportamento - descrivono le collaborazioni tra oggetti

- modelli del cambiamento di stato - descrivono gli stati permessi dal sistema nel tempo

UML contiene anche alcuni costrutti architetturali che permettono la modularizzazione del sistema ai fini dello sviluppo iterativo e incrementale.

3.3 Diagrammi

“Un buon giudizio viene dall’esperienza. L’esperienza viene da un giudizio negativo” (Jim Horning).
L’analisi dei requisiti e il progetto di sistema riguardano soprattutto conoscenze pratiche. Una tecnica di modellazione formale è stata usata nella costruzione di questi diagrammi, ma nessuna conoscenza tecnica è necessaria per poter comprendere e usare questi diagrammi.

3.3.1 Database

Il diagramma seguente fornisce una panoramica visuale del database di Wordpress e tutte le relazioni tra le tabelle create durante l’installazione standard di Wordpress. La panoramica tabellare invece, fornisce dettagli aggiuntivi riguardo le tabelle e le colonne del database.
E’ importante notare che durante l’installazione standard di Wordpress non viene forzato nessun tipo di integrità tra le tabelle, ad esempio tra i post e i commenti. Quando si sviluppa un plugin o un’estensione che manipola il database di Wordpress, lo sviluppatore deve tenere presente che eventuali operazioni per garantire l’integrità del database devono essere effettuate via codice, ad esempio rimuovendo i record da altre tabelle del database utilizzando una serie di query SQL quando si opera su chiavi esterne.
Per una panoramica di tutte le tabelle costruite durante l’installazione di Wordpress seguite da informazioni specifiche per ogni tabella consultare il documento originale, disponibile all’indirizzo http://codex.wordpress.org/Database_Description

3.3.2 Use Case

In UML, gli UCD (Use Case Diagram) sono diagrammi dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso. Sono impiegati soprattutto nel contesto della Use Case View di un modello, e in tal caso si possono considerare come uno strumento di rappresentazione dei requisiti funzionali di un sistema. Durante la progettazione, ad esempio, potrebbero essere usati per modellare i servizi offerti da un determinato modulo o sottosistema ad altri moduli o sottosistemi. In molti modelli di sviluppo software basati su UML, la Use Case View e gli Use Case Diagram rappresentano la panoramica più importante, attorno a cui si sviluppano tutte le altre attività del ciclo di vita del software.
Le entità (model element) principali coinvolte in uno Use Case Diagram sono tre: system, actor e use case:

- System - Il sistema può essere rappresentato come un rettangolo vuoto. Questo simbolo viene messo in relazione con gli altri nel senso che i model element che rappresentano caratteristiche del sistema verranno posizionati all’interno del rettangolo, mentre quelli che rappresentano entità esterne (appartenenti al dominio o al contesto del sistema) sono posizionati all’esterno. In molti diagrammi UML il simbolo per il sistema viene omesso in quanto la distinzione fra concetti relativi al sistema e concetti relativi al suo contesto si può in genere considerare implicita.

- Actor - sono rappresentati graficamente nel diagramma da un’icona che rappresenta un uomo stilizzato (stickman). Formalmente, un attore è una classe con stereotipo “actor”. Praticamente, un attore rappresenta un ruolo coperto da un certo insieme di entità interagente col sistema (inclusi utenti umani, altri sistemi software, dispositivi hardware, ecc...). Un ruolo corrisponde a una certa famiglia di interazioni correlate che l’attore intraprende col sistema.

- Use Case - è rappresentato graficamente come un’ellisse contenente il nome del caso d’uso. Formalmente, lo Use Case è una classe di comportamenti correlati. Praticamente, uno use case rappresenta una funzione o servizio offerto dal sistema a uno o più attori. La funzione deve essere completa e significativa dal punto di vista degli attori che vi partecipano.

3.3.3 Activity

L'activity Diagram è un diagramma definito all'interno dell'UML (Unified Modeling Language) che definisce le attività da svolgere per realizzare una data funzionalità. Può essere utilizzato durante la progettazione del software per dettagliare un determinato algoritmo. Più in dettaglio, un activity diagram definisce una serie di attività o flusso, anche in termini di relazioni tra le attività, i responsabili per le singole attività e i punti di decisione. L'activity diagram è spesso usato come modello complementare allo Use Case Diagram per descrivere le dinamiche con cui si sviluppano i diversi use case.

3.3.4 Sequence

Un Sequence Diagram è un diagramma previsto dall'UML utilizzato per descrivere una determinata sequenza di azioni in cui tutte le scelte sono state già effettuate. Nel diagramma non compaiono scelte, nè flussi alternativi. Normalmente da ogni activity Diagram sono derivati uno o più Sequence Diagram. Se l'Activity Diagram descrive due flussi di azioni alternativi, se ne potrebbero ricavare due scenari e quindi due Sequence Diagram alternativi. Questo tipo di diagramma descrive le relazioni che intercorrono, in termini di messaggi, tra attori, oggetti di business, oggetti o entità del sistema.

3.4 Esempi d'uso

In questa sezione, mostriamo alcuni screenshot della piattaforma illustrando i principali casi d'uso.
Con la piattaforma Matchmed è possibile consultare una mappa interattiva che mostra i reperti catalogati. E' possibile creare nuovi reperti, modificarli o eliminarli. in Figura 3.4.1 mostra la home page della piattaforma Matchmed.

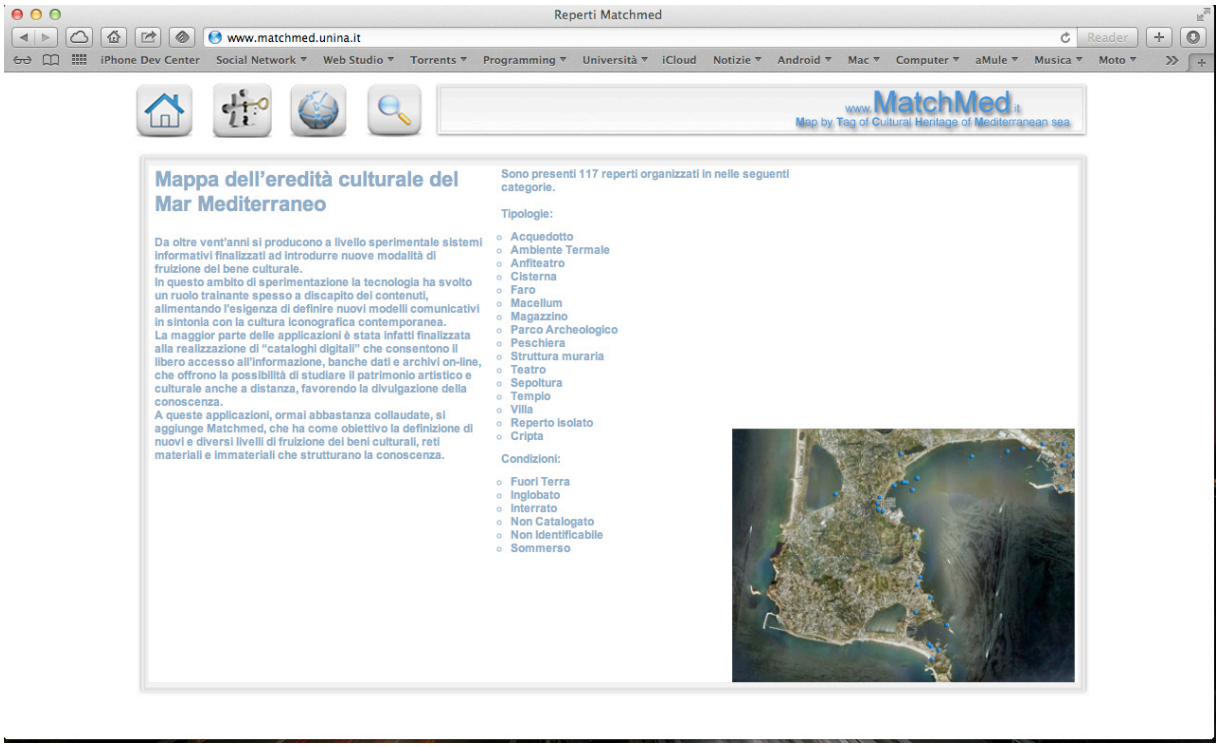

Figura 3.4.1

Per consultare la mappa interattiva, clicchiamo sul tasto "visualizza reperti", come mostrato in Figura 3.4.2. Sul lato sinistro dell'interfaccia utente è possibile filtrare i reperti presenti sulla mappa in base alla tipologia o allo stato di conservazione. E' possibile inoltre selezionare uno specifico reperto dalla lista in alto a sinistra. Selezionando un reperto possiamo visualizzare un'anteprima della scheda di dettaglio direttamente sulla mappa interattiva, come mostrato in Figura 3.4.3. Cliccando sul pulsante nell'anteprima di dettaglio, visualizziamo la scheda di dettaglio del reperto scelto, come mostrato in Figura 3.4.4. Per poter inserire un nuovo reperto o modificare quelli esistenti, è necessario autenticarsi sulla piattaforma Matchmed effettuando il login, come mostrato in Figura 3.4.5. Una volta effettuato il login, l'utente ottiene i permessi per gestire i reperti presenti sulla piattaforma. E' possibile creare un nuovo reperto tramite l'apposita schermata di inserimento, come mostrato in Figura 3.4.6. Per ogni reperto, è possibile definire l'ambito: Campi Flegrei o Napoli; il comune: Napoli, Miseno, Pozzuoli, Bacoli, Baia, Monte di Procida, Cuma, Posillipo; la tipologia: acquedotto, ambiente termale, anfiteatro, cisterna, faro, macellum, magazzino, parco archeologico, peschiera, struttura muraria, sepoltura, teatro, tempio, villa; la condizione: fuori terra, inglobato, interrato, non catalogato, non identificabile e sommerso.
Selezionando un reperto esistente, la maschera di inserimento appare già popolata con le informazioni relative al reperto selezionato, come mostrato in Figura 3.4.7. Il funzionamento è esattamente lo stesso, tutte le informazioni possono essere modificate e aggiornate cliccando sul pulsante "salva modifiche". Durante la fase di inserimento e modifica è possibile caricare e gestire anche i file multimediali associati ad ogni reperto. E' possible caricare diversi tipi di file per ogni reperto: immagini, video, file di testo, file dxf o dwg, come mostrato in Figura 3.4.8. In Matchmed è prevista un'altra modalità di ricerca e consultazione dei reperti, con la possibilità di filtrare l'elenco esistente con dei criteri di ricerca.

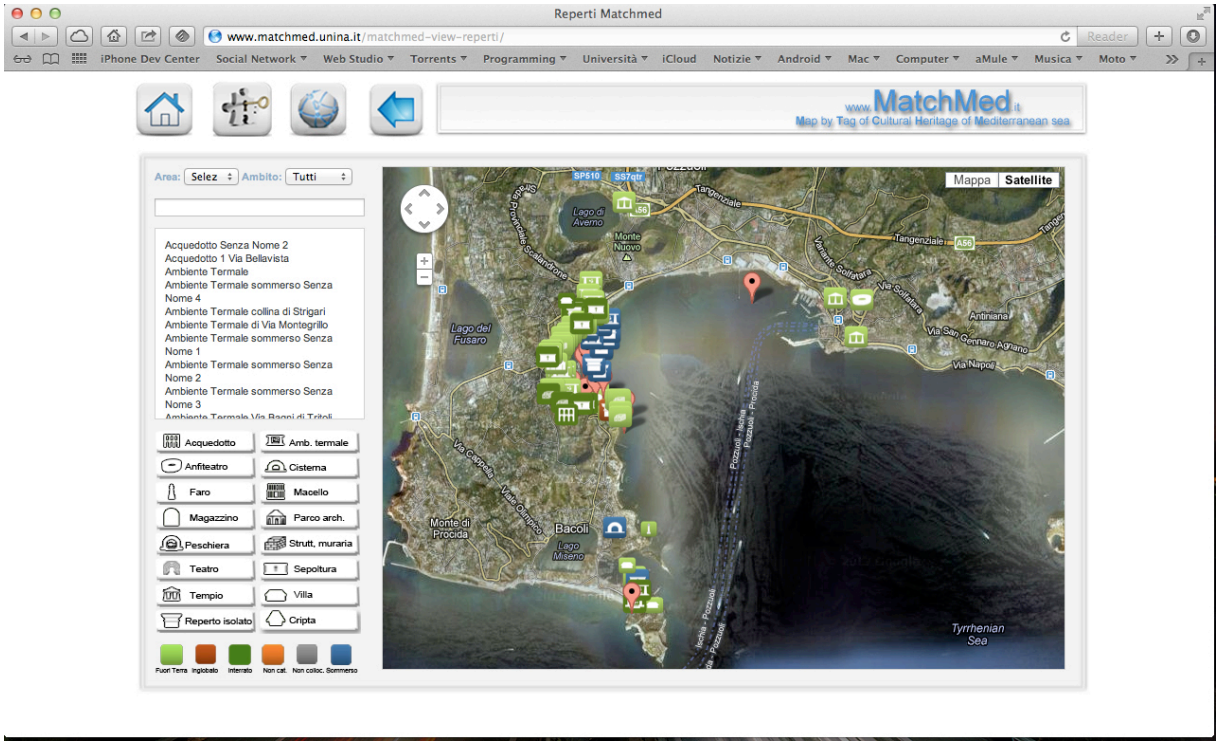

Figura 3.4.2

La ricerca può essere effettuata selezionando i campi di interesse: id specifico, ambito, comune, nome, ubicazione, tipologia, condizione, proprietà, accessibilità, visitabile, presenza di vegetazione.

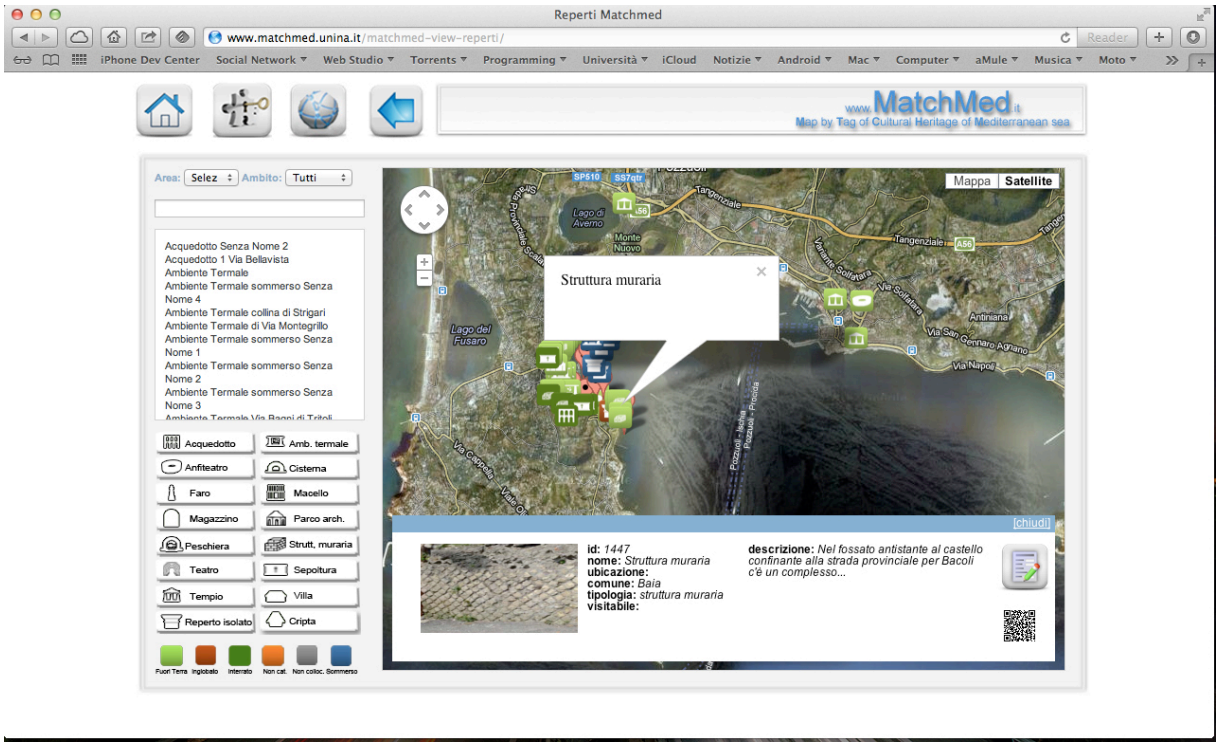

Figura 3.4.3

Questo tipo di ricerca è mostrato in Figura 3.4.8. Una volta ottenuti i risultati della ricerca, è possibile esportarli in formato Excel cliccando sul pulsante “Excel” che appare di fianco al tasto “cerca”.

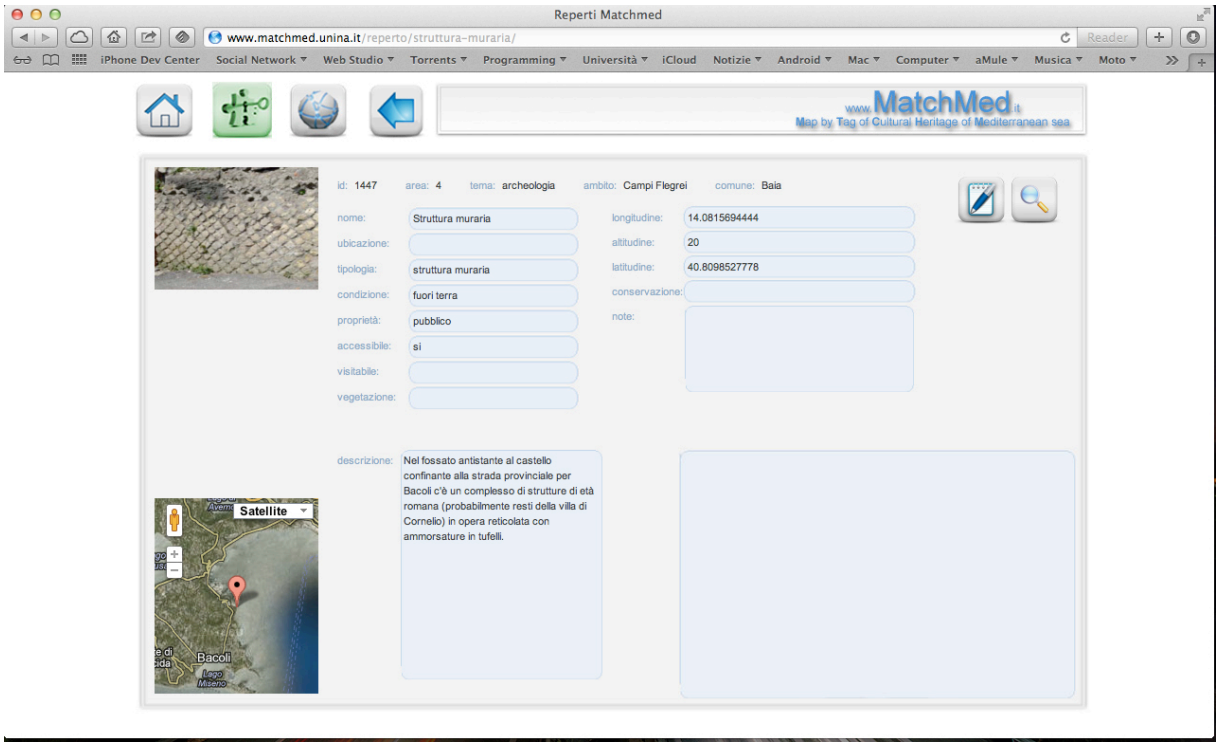

Figura 3.4.4

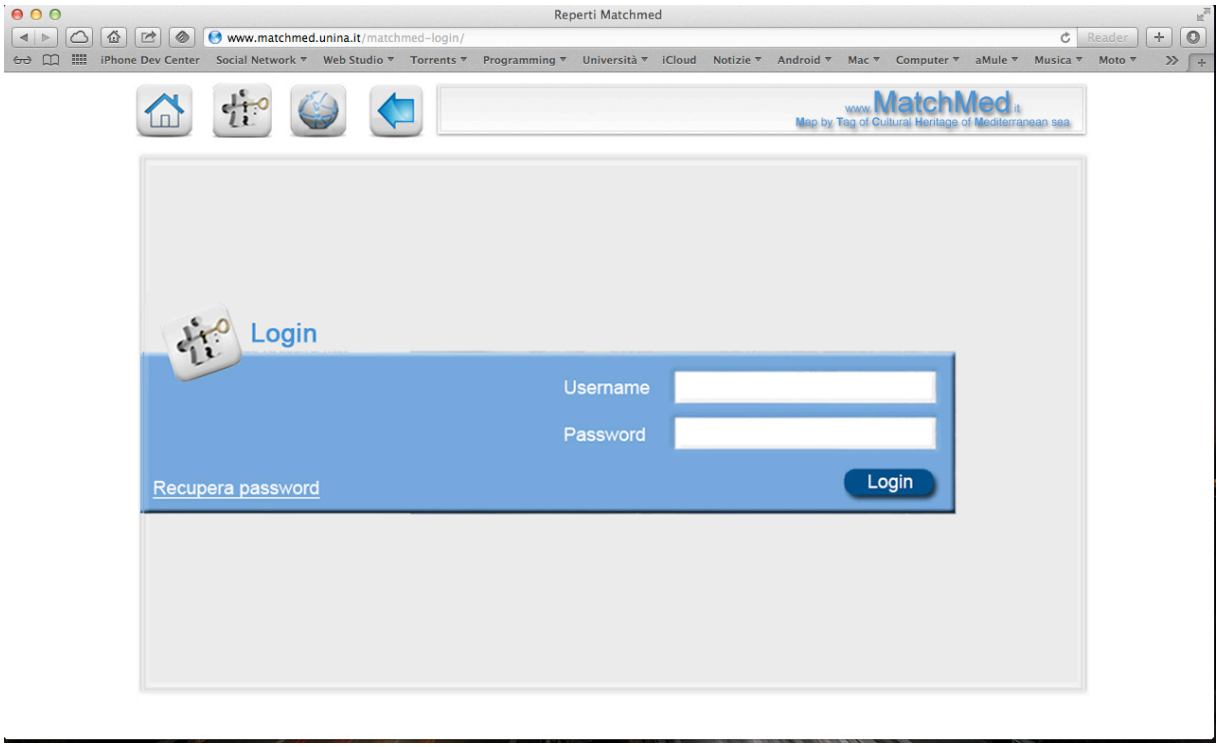

Figura 3.4.5

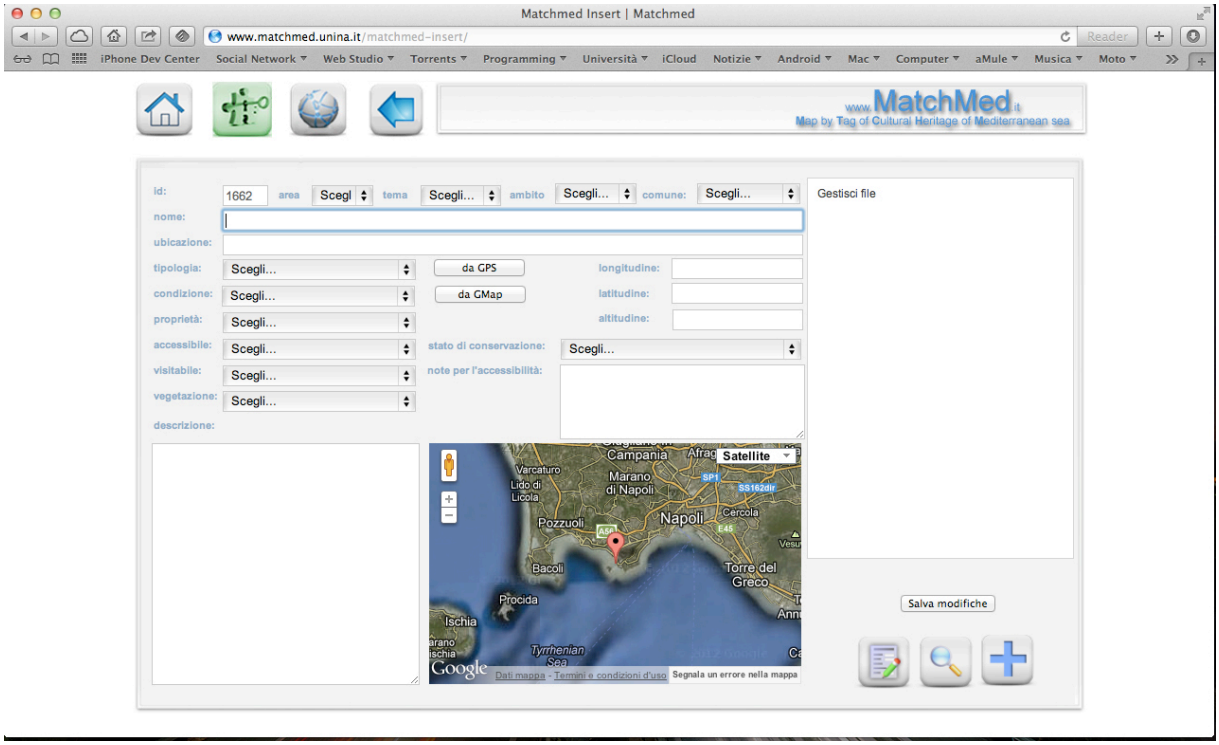

Figura 3.4.6

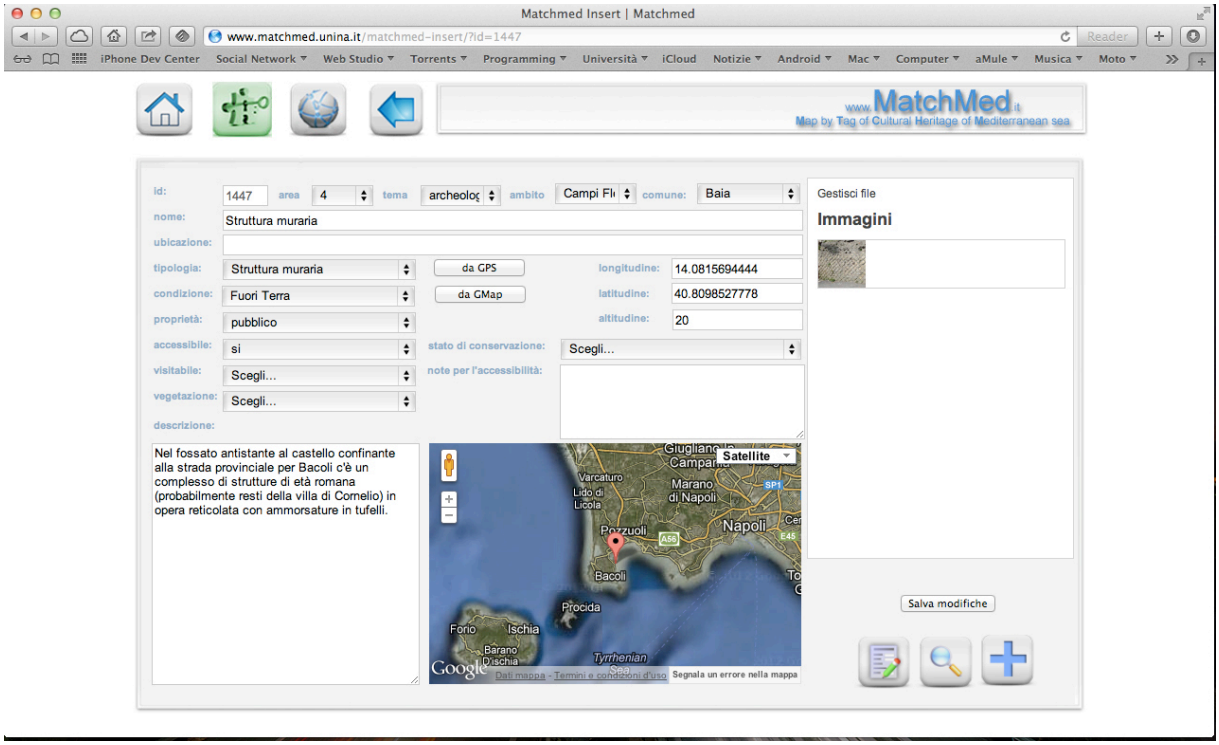

Figura 3.4.7

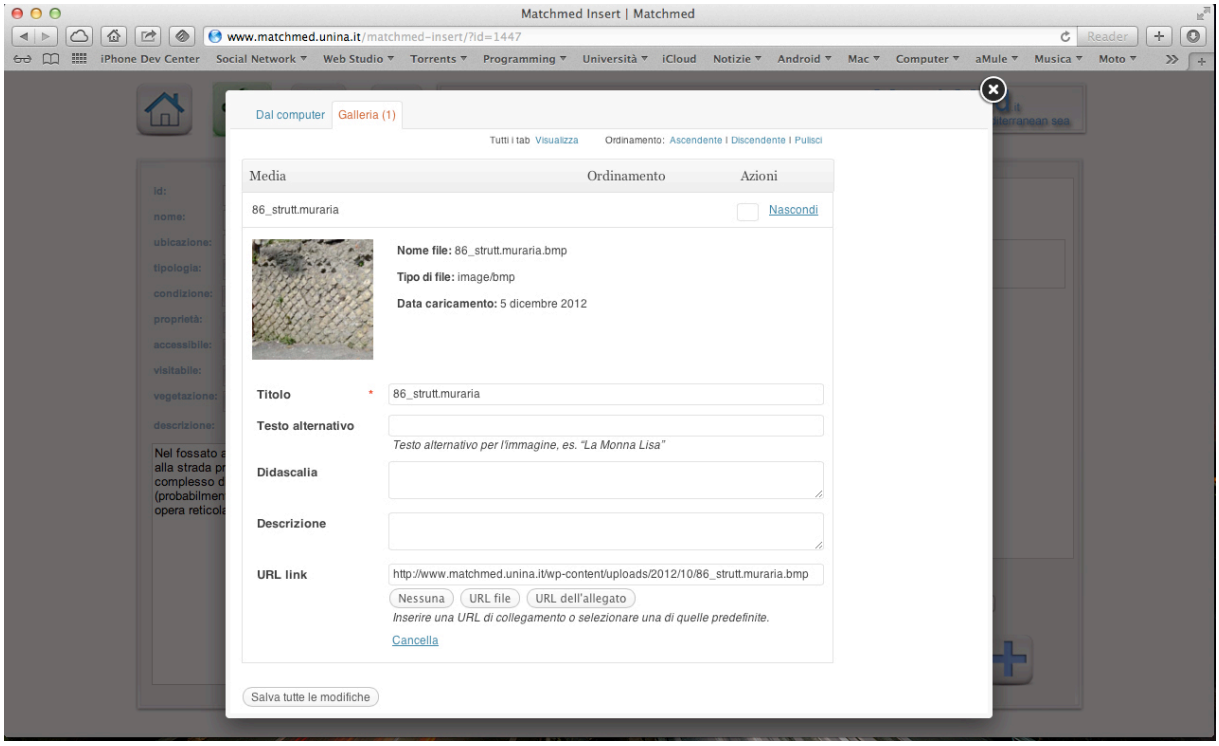

Figura 3.4.8

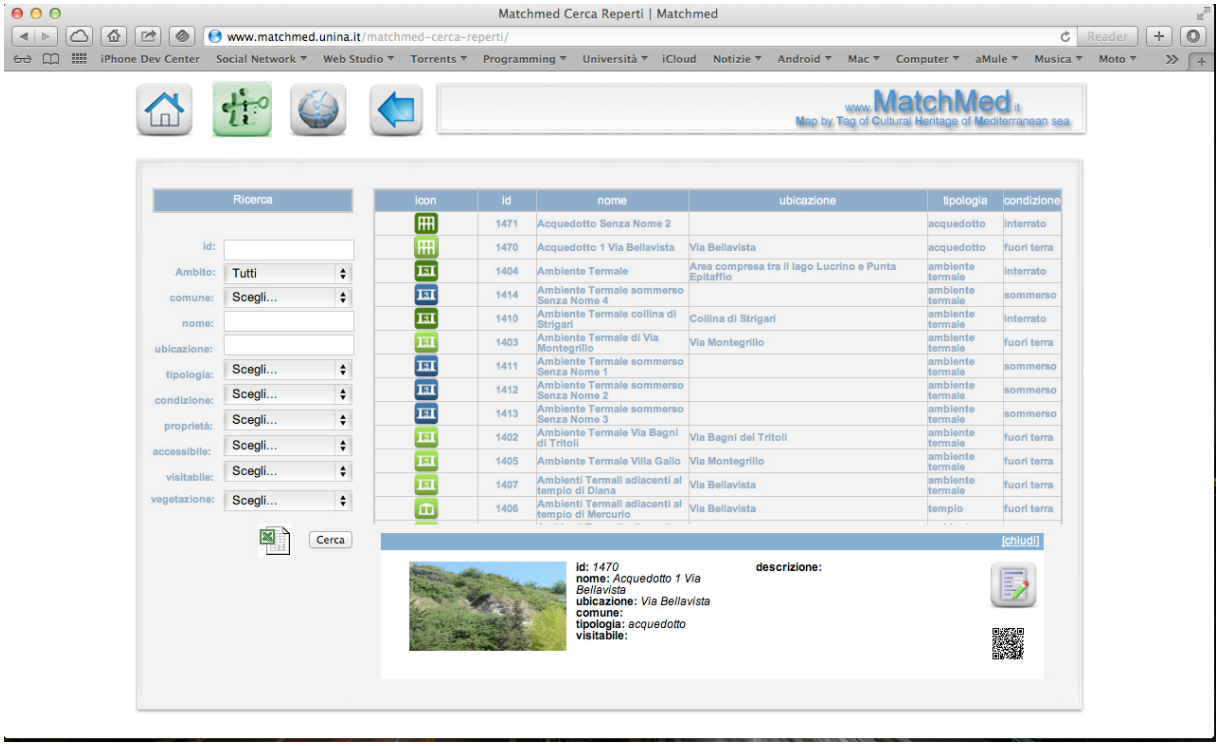

Figura 3.4.9

Bibliografia

Massimiliano Bigatti,
UML e Design Pattern, Notazioni grafiche e soluzioni per la progettazione,
HOEPLI Informatica 2010

Dean Leffingwell, Don Widrig
Managing Software Requirements, a Unified approach
Addison Wesley 2000

Nicholas C. Zakas, Jeremy McPeak, Joe Fawcett
Ajax, Guida per lo sviluppatore, seconda edizione
HOEPLI Informatica 2011

Tim Converse, Joyce Park
Guida a PHP, seconda edizione
McGraw-Hill 2003

Bogdan Brinzarea, Cristian Darie Filip, Chereches-Tosa, Mihai Bucica
Ajax e PHP, Sviluppare applicazioni web dinamiche
HOEPLI Informatica 2011

Paolo Azteni, Stefano Ceri, Stefano Paraboschi, Riccardo Torlone
Basi di dati, Modelli e linguaggi di interrogazione, seconda edizione
McGraw-Hill 2006

Elizabeth Castro
HTML, XHTML e CSS per il world wide web
HOPS tecniche nuove 2007

Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
Design Patterns, Elements of Reusable Object-Oriented Software
Addison Wesley Professional Computing Series 2002

Wordpress, http://wordpress.org

Drupal, http://drupal.org

Joomla, http://www.joomla.org

PHP.net, http://php.net

MySQL, http://www.mysql.com

Apache, http://www.apache.org

“Ajax: a New Approach to Web Applications”, www.adaptivepath.com/pubblications/essays/archives/000385.php

Michael Mahemoff, www.mahemoff.com

http://mashable.com, https://accounts.google.com, https://accounts.google.com, http://w3techs.com, http://www.socialtechnologyreview.com, http://deviousmedia.com

Siege, http://www.joedog.org/siege-home/

Squid, http://www.squid-cache.org

Aptana, http://www.aptana.com

gitHub, https://github.com

Wikipedia, http://it.wikipedia.org

Fabio Armani, http://www2.mokabyte.it/cms/article.run?permalink=mb165_metodoagile-1