Corso di laurea in Ingegneria Informatica – a.a. 2009/2010

TECNOLOGIE INFORMATICHE APPLICATE – PROGETTO DIDATTICO

La realizzazione del progetto, da svolgersi singolarmente da parte degli studenti, ha lo scopo di applicare le conoscenze acquisite durante il corso allo sviluppo di servizi che, seppur nella loro limitata complessità, implementino delle funzionalità logicamente ben definite e realistiche[1]. Nel suo insieme, il servizio realizzato deve essere inedito[2], composto da funzionalità coese[3], non ridondanti[4], modulari e leggibili[5].

Le tecnologie[6] che si possono adoperare sono *HTML 4.01*, *CSS 2.1*, *Javascript 1.3 con DOM Level 2*, *PHP 5.1*, *MySQL 5*. Il progetto deve potersi eseguire correttamente con i due browser *Mozilla FireFox 3.x* e *Microsoft Internet Explorer 8.*x. Tra le tipologie di applicazione compatibili con i contenuti del corso vi sono le seguenti. Scegliere prevalentemente una sola tipologia. Eventuali proposte alternative potranno essere concordate con il docente.

a) Interfaccia web orientata ai **contenuti ed ai fogli di stile**, in cui vengono adoperati tre stili diversi, uno per ogni mezzo di visualizzazione: PC desktop, stampante[7] o palmare[8]. In tal caso la documentazione dovrà specificare chiaramente le caratteristiche dei diversi dispositivi, il contenuto dovrà provenire da una o più pagine cartacee di una pubblicazione del mondo reale (da mostrare ai docenti in fase di verifica), contenere una certa varietà strutturale (elenchi, tabelle, immagini), ed avere anche delle funzionalità aggiuntive consentite dalle tecnologie adoperate (es. mappe immagini, ancore).

b) Interfaccia web orientata alla **verifica dei contenuto di moduli** prima dell'invio al server[9]. Anche in tal caso il contenuto dovrà provenire da un modulo del mondo reale (da mostrare ai docenti in fase di verifica) di cui non esista il formato HTML su web. Ad esempio partire da un formato pdf o cartaceo di un modulo per servizi di pubbliche amministrazioni (moduli di riduzione tasse, di registrazione, …) e controllare adeguatamente il corretto inserimento dei campi. I campi dovranno avere dei valori di default o di esempio per consentire un veloce test del sistema con input regolari.

c) Interfaccia web orientata alla **creazione automatica di contenuti lato client**, sulla base di algoritmi non troppo complessi e sempre relativi a contesti del mondo reale. Ad esempio il calcolo di imposte, di indici di redditività, di dati statistici, ecc.

d) Interfaccia web **orientata agli eventi,** in cui l'utente interagisce, tramite il mouse o la tastiera, con semplici oggetti grafici in movimento o fissi e ne provoca lo spostamento o la alterazione secondo regole stabilite da un gioco reale. Esempi: dama, filetto, tiro a segno (di immagini in movimento come volatili), gioco del ping-pong, etc.. Non occorre implementare giocatori

---

[1] Le funzionalità sono *realistiche* se implementano un servizio utile in qualche contesto del mondo reale. Ad esempio, una procedura che controlla il formato della data senza tenere conto degli anni bisestili, non è realistica. In tal senso, è buona norma partire da contesti reali dei quali si ha esperienza. Se il servizio da realizzare è logicamente ben definito, si può fare un test funzionale, ossia provare se funziona correttamente. Se ad esempio si realizza un gioco, devono essere chiare le regole. Se si realizza un modulo da compilare, deve essere chiaro come compilarlo correttamente. Pertanto occorre inserire nella pagina index.html (o in una pagina ad essa collegata) un **mini manuale utente (15-20 righe)** che spiega quali senza tecnicismi come si usa il servizio.

[2] Un servizio è *inedito* se non ne esiste già uno praticamente equivalente in termini di codice. Si consiglia di verificare ciò con una semplice ricerca su web. Eventuali frammenti minori di codice presi dal web devono essere studiati e ristrutturati secondo i criteri di programmazione e di validazione del corso. Ad esempio, il calcolo del codice fiscale è un servizio già ampiamente presente sul web, pertanto non può costituire la parte predominante di un progetto. Nel caso si adoperi del codice preso dal web senza alcuna modifica, occorre inserire nel codice il sito di provenienza ed evidenziarlo bene tramite dei commenti, dal momento che non sarà preso in considerazione per la valutazione.

[3] Due o più funzionalità sono *coese* se appartengono al medesimo servizio e possibilmente interagiscono tra loro, come ad esempio le diverse funzioni di un gioco. Una pagina web che include una collezione di tante piccole procedure logicamente indipendenti (ad es. l'ora corrente, le previsioni del tempo, il gioco del tris ed il calcolo del codice fiscale) non presenta funzionalità coese. **È buona norma fissare alcuni servizi di una sola tipologia di progetto, e svilupparli in profondità**, piuttosto che fare un progetto con "un po' di tutto, ma abbozzato".

[4] Due o più funzionalità sono *ridondanti* se contengono parti di codice di controllo o di dati che sono molto simili. È buona norma cercare di accorpare tante procedure simili in un'unica procedura parametrica e condivisa. Ad esempio, in un progetto di verifica dei campi inseriti in form, è buona norma fare procedure condivise per la presenza/assenza di contenuto nei campi, ed anche per la verifica del contenuto specifico (una stringa contenente una espressione regolare è il parametro che contiene il tipo di controllo da attuare). Anche sui contenuti può esserci ridondanza. Ad esempio, è inutile inserire una legge con molti articoli con la medesima struttura che si ripete. Oppure molte immagini in un elenco. Ridurre i contenuti a pochi esempi rappresentativi, specialmente se gestiti con tecnologia lato client. Se gli elementi da inserire devono apparire tutti per questioni di integrità del servizio (ad esempio un menu HTML con tutti i comuni d'Italia), è buona norma suddividere tali elementi in categorie e consentirne il caricamento dinamico (ad esempio, un menu che consente di selezionare la provincia, e poi un secondo menu che si genera dinamicamente con 2-3 comuni di quella provincia, inserendo solo esempi per 2-3 province.

[5] La *modularità* è una importante caratteristica di qualità dei programmi, che misura la estensione di quanto sono composti in parti separate chiamate moduli. Nella fattispecie un servizio è modulare se, a livello di codice, è implementato a funzioni o ad oggetti, e ciascuna funzione (o metodo) è abbastanza corta da potersi leggere grosso modo in una sola videata. Vi sono inoltre un numero minimo di variabili globali. Il codice html, js, css è separato in diversi file. I file sono suddivisi in cartelle per tipologia (img, css, js, php, mid,…), il codice è inserito in file separati rispetto ai dati, vi sono vari file js, css, php ciascuno con funzioni logicamente separate ed indipendenti. La *leggibilità* del codice è, accanto alla modularità, una caratteristica cruciale nei linguaggi di alto livello per la riusabilità del codice. La leggibilità nella fattispecie riguarda il fatto che le variabili e gli oggetti abbiano come nomi dei sostantivi (eventualmente con aggettivi, quindi non nomi incomprensibili come "a", "x"), e le funzioni siano nominate con dei verbi (o frasi verbali). Il codice è indentato. Si studino i laboratori come esempi in tal senso.

[6] I progetti sono discussi in una qualsiasi delle aule didattiche o laboratori di facoltà adibiti a sede di esame, dove non si dispongono dei diritti di amministrazione per installare nuovi pacchetti applicativi o creare utenti sotto ambiente Linux o Windows. Le prove dei progetti avvengono sotto l'utente *studenti* nella piattaforma Windows XP, con i pacchetti di installazione scaricabili da http://tweb.ing.unipi.it/tools/ e configurati come nei laboratori. Anche se si dispone di un portatile, è necessario consegnare un progetto che si può eseguire il pacchetto "TIA all-in-one". Eventuali esigenze specifiche possono essere concordati preliminarmente.

[7] Come terminale stampante si adoperi la *stampante virtuale pdf* presente su http://tweb.ing.unipi.it/tools/.

[8] Come terminale palmare si adoperi il *PDA simulator* presente su http://tweb.ing.unipi.it/tools/.

[9] Per progetti a tecnologia esclusivamente lato client, la ricezione di dati da tabelle di un database può essere simulata con l'accesso a matrici javascript (inserite in un file separato, e contenenti solo alcuni esempi rappresentativi), mentre l'invio di dati ad un server può simularsi tramite l'apertura di una finestra di console in cui si stampa in chiaro la sequenza di termini "variabile = valore".

virtuali o algoritmi intelligenti, sono sufficienti i componenti grafici ed i meccanismi per permettere a giocatori umani di operare secondo i vincoli del gioco. Fanno parte di questa categoria anche meccanismi visuali di invio di informazione (es. la pianta di un teatro o di uno stadio in cui i posti sono prenotabili con un click, un'interfaccia che consente di spostare mobili in una stanza).

e) **Generazione dinamica di pagine lato *client*** per creare nuovi documenti a partire da documenti reali presi dal web (riferire la URL). In tal caso non si dovranno fare ipotesi sul documento da analizzare, mantenendo una certa generalità, se non che esso sia composto da una sola pagina HTML. Ad esempio, creazione automatica del sommario di una pagina HTML di una normativa, sulla base della ricerca delle intestazioni H1,…,H6; generazione di un elenco di frammenti di testo, sulla base di una ricerca di occorrenze di una medesima parola (o gruppo di parole usando semplici espressioni regolari) presente tante volte in un documento (una sorta di motore di ricerca di contenuti semplificato ad una sola pagina); trasformazione di una pagina HTML in un'altra con una alterazione di contenuti (es. togliere tutte le immagini ed i suoni ed inserirvi una semplice descrizione testuale) del codice (es. sostituire i tag <img> o <embed> con i corrispondenti <object>), o dello stile; generazione di una pagina riassuntiva del contenuto di 3-4 pagine indipendenti[10], ecc.

f) **Le tipologie b) d) e) ed f) possono avere una componente lato server**, ossia possono essere presenti dei moduli che registrano profili utente, consultano, memorizzano, eliminano o aggiornano dati su database. Oltre a queste possono essere sviluppate applicazioni lato server come bacheche elettroniche, sistemi di gestione delle discussioni on-line, sistemi di noleggio e di e-commerce.

Si raccomanda una distribuzione dei componenti che sia portabile[11], ed organizzata nel seguente modo. Tutti i file dovranno essere inclusi in una cartella principale nominata con *matricola_cognome* dello studente; a partire dalla pagina principale, nominata come *index.html* (o *index.php*) dovrà essere possibile la navigazione in tutte le pagine del progetto inclusa la documentazione (esclusivamente in formato html) che dovrà spiegare in modo semplice e non tecnico i servizi sviluppati e le modalità di uso. È gradita una organizzazione del codice ad oggetti e l'uso del DOM, soprattutto se si riduce la ridondanza.

Il codice HTML 4.01 strict e CSS 2.01 (sia statico che dinamicamente generato, estratto tramite apposita bookmarklet su Firefox) dovrà contenere dei commenti laddove è stato necessario adoperare dei tag di tipo *deprecated*[12] o in *disuso*. È vietato l'uso di innerHTML e simili, tranne in casi concordati. Per eventuali componenti eseguibili con *plug-in* aggiuntivi (ad esempio *Java Applet* ©, *Flash Macromedia* ©) occorrerà riferire l'*URL* da cui possono essere scaricati tali applicativi; la parte relativa allo sviluppo di componenti lato client al di fuori di *HTML-CSS-Javascript* non verrà comunque valutata e pertanto non dovrà essere inserita nella documentazione o nei commenti, anche se potrà essere inclusa nel progetto. A titolo indicativo, le dimensioni adeguate di un progetto sono pari a tre volte[13] la dimensione tipica dei laboratori svolti durante il corso.

Modalità di assistenza, di consegna e di valutazione dei progetti

Per **assistenza** nella fase di sviluppo, rivolgersi all'esercitatore (http://www2.ing.unipi.it/~o1553499/,m.cimino@iet.unipi.it)

Per **consegnare** il progetto in versione definitiva, comprimere[14] la cartella principale ed inviarla come allegato di posta elettronica all'esercitatore, **non oltre il terzo giorno solare prima del giorno dell'esame** (es. esame 17, consegna nel giorno 14).

Il progetto è presentato[15] il giorno dell'esame, mediante un colloquio orale nel quale sono valutati i seguenti aspetti:

01	SERVIZIO LOGICAMENTE DEFINITO
02	TEST FUNZIONALE SUI DUE BROWSER
03	VALIDITA' HTML/CSS
04	ASSENZA ERRORI JAVASCRIPT/PHP
05	DIMENSIONI PROGETTO
06	CORRETTEZZA RISPOSTE SU CODICE

07	COESIONE FUNZIONALITÀ
08	CODICE NON RIDONDANTE
09	CODICE INEDITO
10	LEGGIBILITÀ E MODULARITÀ
11	REALISTICITÀ
12	PORTABILITÀ

La presentazione del progetto prevede anche la discussione delle soluzioni realizzative adottate, e una verifica su argomenti di carattere teorico/metodologico trattati nel corso. Il progetto, una volta discusso, è valido per tutto l'anno accademico e, in caso di esito negativo di un appello, rimane valido per gli appelli successivi.

---

[10] Per fare in modo che lo script possa operare su pagine web provenienti da altri domini, salvare staticamente una copia di tali pagine in locale, e nominarle secondo uno standard (es. *page01.html*, *page02.html*,…) in un percorso predefinito, che consenta allo script di caricarne di nuove senza essere modificato.

[11] Un modulo è portabile se è in grado di funzionare da diversi punti del file system, su diverse piattaforme di sistema. A tale scopo, non deve esserci alcun percorso assoluto nei riferimenti ai componenti all'interno di tutte le pagine (quindi nessun riferimento a percorsi al di fuori della cartella principale) a meno di ancore ad indirizzi web per riferire le fonti adoperate. Inoltre, gli applicativi (quali browser, *Apache*, *PHP*, *MySQL*) devono essere considerati con i percorsi e le impostazioni di default, con i settaggi adoperati nei laboratori. Eseguire un test funzionale del proprio progetto adoperando il pacchetto "TIA all-in-one". Eventuali impostazioni specifiche dovranno essere concordate. Occorre fornire lo script SQL per generare il db, le tabelle ed i dati di partenza.

[12] Anche per pagine intrinsecamente di tipo HTML *transitional* (es. dove si adoperano *iframe*) la validazione tentata in sede di esame è HTML 4.01 *strict*., per verificare che i tag *deprecated* siano solo il minimo necessario. È necessario abolire anche i *warning* generati dal validatore CSS o, in alternativa, giustificare con un commento la loro presenza. Gli strumenti di validazione sono disponibili su http://tweb.ing.unipi.it/tools/.

[13] Considerando codice e dati non ridondanti. Il codice deve essere indentato in modo da evidenziarne la struttura. Per dubbi sulle dimensioni andare a ricevimento.

[14] In formato ".rar" o ".zip".(ottenendo es. *12345_rossi.rar*, o *12345_rossi.zip*). Eliminare ogni frammento di codice o di file non adoperato. A seguito della consegna, si riceverà entro 48 ore una e-mail di conferma della ricezione. Un progetto consegnato deve essere discusso nel primo appello disponibile, altrimenti viene rimosso dall'archivio delle consegne.

[15] Il candidato dovrà portare una copia dei file consegnati, preparandosi all'occorrenza a caricarli personalmente sulla macchina disposta a tale scopo.