Tecnologie Web T
17 Gennaio 2018 – Compito

Tempo a disposizione: 3 ore

La soluzione comprende la **consegna elettronica** dei seguenti file mediante l'apposito applicativo Web **esamix** (**http://esamix.labx**):

Quasi-IM.zip	file zip contenente il sorgente java/class e pagine Web per punto 1
Download.zip	file zip contenente il sorgente java/class e pagine Web per punto 2
Prenotazione.zip	file zip contenente il sorgente java/class e txt per punto 3

Ogni file .zip consegnato DEVE CONTENERE TUTTI e SOLI i file creati/modificati e/o ritenuti importanti in generale ai fini della valutazione (ad esempio, descrittori, risorse statiche o dinamiche, codice Java e relativi .class, ecc.) e NON dell'intero progetto

N.B. Per superare la prova scritta di laboratorio ed essere ammessi all'orale, è necessario totalizzare almeno 18 punti (su un totale disponibile di 33), equamente distribuiti sui tre esercizi, ovvero almeno 6 punti sul primo esercizio, 6 punti sul secondo esercizio e 6 punti sul terzo esercizio

---

Studenti in debito di Tecnologie Web L-A
Viene richiesto lo svolgimento dei soli esercizi 1 (17 punti) e 2 (16 punti). Tempo a disposizione: 2 ore.
I 18 punti necessari per l'ammissione all'orale sono così distribuiti: almeno 10 punti sul primo esercizio e almeno 8 punti sul secondo

---

ESERCIZIO 1 (12 punti)
Si realizzi un'applicazione Web, principalmente basata su tecnologie **JSP**, Java **servlet** e **Javascript**, per la realizzazione di un servizio di **(quasi-)instant messaging**.
L'applicazione deve consentire ad ogni utente di aprire una propria sessione, di essere riconosciuto e di selezionare un gruppo per lo scambio di messaggi (identificato da una stringa, ad es. gruppo "Tecnologie Web T"); come ipotesi semplificativa, non è possibile per uno stesso utente avere più sessioni correntemente attive e ad ogni sessione è associato un solo gruppo di messaggistica; per cambiare gruppo è necessario chiudere la sessione corrente.
Una volta aperta la sessione, l'utente può scrivere e ricevere messaggi per il gruppo associato (scritti da quel momento in avanti). I messaggi sono semplici sequenze di caratteri separati da
 e visualizzati uno di seguito all'altro. Il servizio di (quasi)-instant messaging dovrà:

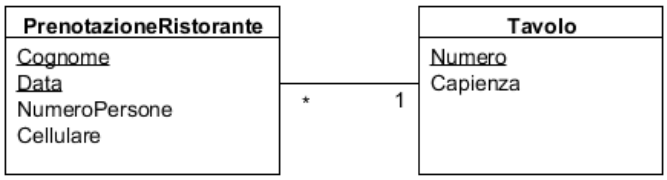
- permettere il refresh (pulsante "aggiorna") dei messaggi ricevuti e visualizzati;
- consentire "l'invio" di un nuovo messaggio a tutti gli utenti del gruppo di messaggistica con sessioni attive;
- visualizzare in basso a destra la stringa "x nuovi messaggi disponibili. Fai il refresh!", con x aggiornato ogni 30 secondi in modo automatico.

A causa di operazioni di invio, sono possibili modifiche concorrenti allo stato del servizio di messaggistica che possano portare a inconsistenze per i vari utenti con sessioni attive? Si indichi la risposta come commento nel codice sorgente consegnato.
Inoltre, deve essere data la possibilità all'amministratore del sistema (username=admin; password=admin) di interrompere i) qualsiasi sessione attiva scegliendo l'utente corrispondente e ii) tutte le sessioni relative a un gruppo del servizio di messaggistica.

Tecnologie Web T
17 Gennaio 2018 – Compito

ESERCIZIO 2 (10 punti)
Si realizzi un'applicazione Web, che usi prevalentemente tecnologie **Javascript**, Java **servlet**, **AJAX** e **JSON** per lo **scaricamento concorrente di x pagine Web** da URL differenti.
In particolare, l'applicazione Web deve essere costituita da una pagina **home.html** che consente all'utente di inserire un numero naturale **x** che indichi il numero di pagine da scaricare; da home.html si deve essere ri-diretti x volte a una pagina per l'inserimento di una stringa, ciascuna delle quali rappresenta uno degli x URL da scaricare. Al termine dell'inserimento stringhe, l'applicazione dovrà effettuare lo scaricamento concorrente di tutte le x pagine Web.
Problemi in caso di accesso concorrente allo stesso URL da parte di più clienti? Vantaggi/svantaggi nell'uso del modello di concorrenza deprecated per servlet? Si risponda a tali domande tramite commenti nel file sorgente.

ESERCIZIO 3 (11 punti)
Partendo dalla realtà illustrata nel **diagramma UML** di seguito riportato, si fornisca una soluzione alla gestione della persistenza basata su metodologia **Forza Bruta** in grado di "mappare" il modello di dominio rappresentato dai **JavaBean** del diagramma UML con le corrispondenti **tabelle relazionali derivata dalla progettazione logica** del diagramma stesso.



Si consideri inoltre la presenza del vincolo: "*Uno stesso tavolo può essere prenotato più volte solo se in date diverse*".

Nel dettaglio, dopo aver creato da applicazione Java lo schema della tabella nel proprio schema nel database **TW_STUD** di **DB2** (esplicitando tutti i vincoli derivati dal diagramma UML) e implementato **JavaBean** e metodi necessari per la realizzazione delle **operazioni CRUD**, si richiede la definizione del metodo principale di richiesta prenotazione `boolean RichiestaPrenotazione(String Cognome, Date data, Int numeroPersone, String Cellulare)`. Tale metodo, mediante l'uso del metodo di supporto `String NumeroTavolo DisponibilitaTavolo(Date data, Int numeroPersone)`, verifica la disponibilità di almeno un tavolo di `capienza` >= `numeroPersone` per la data richiesta e, in caso di esito positivo, restituisce il codice numerico (`NumeroTavolo`) di uno di questi. Tale codice è usato dal metodo `RichiestaPrenotazione` per procedere all'inserimento persistente della prenotazione nel DB e alla restituzione del valore di verità `true` attestante l'accettazione della prenotazione. In caso di esito negativo di disponibilità tavolo invece, il metodo `DisponibilitaTavolo` restituisce `null`, mentre il metodo principale `RichiestaPrenotazione` restituisce direttamente il valore di verità `false` attestante il rifiuto della prenotazione.
Si richiede quindi di realizzare una classe di prova in grado di:

- inserire diversi tavoli e diverse prenotazioni nelle tabelle corrispondenti;
- utilizzare correttamente i metodi `RichiestaPrenotazione` e `DisponibilitaTavolo` per verificare la disponibilità o meno di un tavolo rispetto a una determinata richiesta (si contempli sia il caso di risposta positiva che negativa);
- produrre una stampa completa, opportunamente formattata, delle prenotazioni (complete di numero tavolo) presenti nel DB **prima e dopo** l'inserimento al punto precedente sul file **Prenotazione.txt**.