Tecnologie Web T
25 Luglio 2016 – Compito

Tempo a disposizione: 3 ore

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

Wiki.zip	file zip contenente il sorgente java/class e pagine Web per punto 1
Prefetch.zip	file zip contenente il sorgente java/class e pagine Web per punto 2
Ospedale.zip	file zip contenente il sorgente java/class e file XML 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 (11 punti)
Si realizzi un'applicazione Web, principalmente basata su tecnologie **JSP**, **Java servlet** e **Javascript**, per la redazione collaborativa di capitoli di un libro da parte di autori multipli.
L'applicazione deve consentire ad ogni utente di selezionare il capitolo (numero naturale fra 1 e 10) al quale intende contribuire nella sessione di lavoro; dopo tale selezione, l'utente deve ricevere, via **JSON**, il contenuto corrente del capitolo, che verrà visualizzato nel browser Web, dando anche la possibilità all'utente di modificarlo; le modifiche saranno salvate e rese visibili agli altri utenti solo dopo la pressione esplicita di un tasto di "Invio modifiche".
Si noti che utenti multipli potranno operare anche sullo stesso capitolo concorrentemente: nel caso in cui utenteX abbia ricevuto una versione precedente del capitoloY rispetto a quella modificata lato servitore da utenteZ, la richiesta di "Invio modifiche" di utenteX deve fallire e a tale utente deve essere inviata la nuova versione del capitolo, sempre tramite JSON.
Inoltre, lato servitore, con periodicità 10 minuti, si dovrà controllare se l'intero libro contiene al momento più di 10 occorrenze della parola proibita "aaaaaaaa"; in caso positivo, il libro deve essere interamente cancellato perché non osserva le regole di censura in vigore.

ESERCIZIO 2 (11 punti)

Si realizzi un'applicazione Web, principalmente basata su tecnologie **Javascript e AJAX**, per il prefetching probabilistico di informazioni geolocalizzate su un gioco di realtà aumentata tipo Pokémon Go.

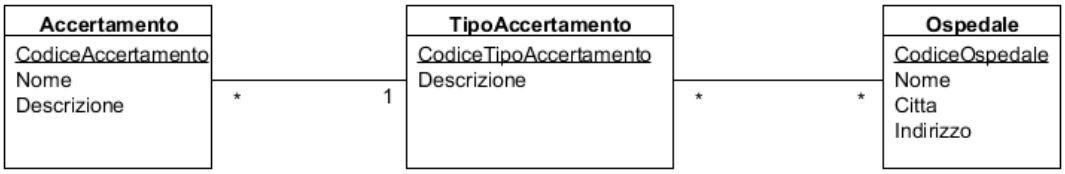
L'applicazione Web deve consentire ad ogni utente giocatore di indicare la sua posizione corrente e di vedersi visualizzare, in risposta, le info relative a tutti gli eventuali personaggi/avatar visibili dalla sua posizione corrente. Si suggerisce di mantenere le info sui personaggi memorizzate lato servitore in una semplice tabella persistente con coordinate cartesiane della posizione del personaggio, nome del personaggio, testo di descrizione.

Si supponga che l'informazione di posizione dell'utente giocatore sia nota in ogni momento all'applicazione (viene inserita tramite form; coordinate cartesiane (x,y) espresse in metri); ogni personaggio è visibile se e solo se la sua distanza dal giocatore è minore di 10 metri.

Per ottenere un comportamento fortemente responsive dell'applicazione Web, come in esempi di gaming reali, si faccia in modo che il browser Web faccia il download in background, in modo asincrono e concorrente, delle informazioni relative ai personaggi che saranno eventualmente visibili nella posizione futura più probabile per il giocatore; per semplicità, si supponga che la posizione futura più probabile sia (x2,y2) = (x1+50, y1).

ESERCIZIO 3 (11 punti)

Partendo dalla realtà illustrata nel **diagramma UML** di seguito riportato, si fornisca una soluzione alla gestione della persistenza basata su **Hibernate** in grado di "mappare" efficientemente e con uso di ID surrogati il modello di dominio rappresentato dai **JavaBean del diagramma UML** con le corrispondenti **tabelle relazionali derivate dalla progettazione logica del diagramma** stesso.



Nel dettaglio, dopo aver creato da applicazione Java gli schemi delle tabelle all'interno del proprio schema nel database **TW_STUD** di DB2 (esplicitando tutti i vincoli derivati dal diagramma UML), implementato i **JavaBean**, definiti i **file XML di mapping** e il **file XML di properties**, si richiede la realizzazione di una classe di prova facente uso delle **API Hibernate** in grado di:

- inserire due o più tuple nelle tabelle di interesse;
- determinare i) elenco dei nomi degli accertamenti di tipo "Analisi di Laboratorio" erogati presso l'ospedale Policlinico S.Orsola-Malpighi di Bologna; ii) per ogni ospedale, nome, città, indirizzo e numero totale di accertamenti erogabili presso la struttura;
- stampare i risultati ottenuti al punto precedente sul file **Ospedale.txt**;

il tutto, mediante opportuna gestione delle **transazioni**.

N.B. La soluzione deve **sfruttare i mapping M-N e 1-N specificati nel diagramma UML**. Ogni ulteriore scelta fatta dallo studente deve essere opportunamente giustificata mediante commenti nel codice.