NOTE:
LISTA eventi attivi — ogni evento ha una LISTA selezionato l'evento vede i messaggi inseriti dall'addettocon i nomi utenti che lo hanno visualizzato — al termine dell'avento viene tolto dalla lista attivi ed eliminata la sua lista visitatori

e alla Programmazione Web del 21 luglio 2020

a un servizio di webcasting di notizie sportive: con una
ıi minuto, o meno) viene dato un aggiornamento su un dato
io,
che nea un servizio analogo.

NOTE:
invece di fare la lista visidatori dell'evento potrei fare una lista per utente con i nomi degli eventi che ha visualizzato

PAGINA B:
pagina di login dell'utente che ne chiede lo username
FATTO

PAGINA A:
addetto crea evento (lo inserisce nella lista eventi attivi), e ne inserisce i diversi messaggi in cosa, chiude poi l'evento una volta terminato (lo elimina dalla lista eventi attivi)

FATTO

PAGINA C:
utente vede la lista degli eventi disponibili in diretta — vede gli eventi che non ha ancora visto, quelli gia visti sono nascosti finchè non lo decide tramite pulsante — se non ci sono eventi o se li ha visti tutti allora messaggio di errore

PAGINA D:
visualizza i messaggi in tempo reale dell'evento selezionato — modifica il tempo di refresh dei messaggi

1) Il servizio ha una componente di feeding: un addetto usa una pagina web per:
- inserire in una coda dei brevi messaggi di testo identificati da un numero progressivo (il primo messaggio inserito crea la coda, se necessario).
- chiudere il webcasting dell'evento tramite uno speciale messaggio "END"
Il sistema è in grado di gestire un solo webcasting alla volta.

2) La componente vista dagli utenti è una pagina web che mostra i nuovi eventi, dall'ultimo visualizzato: se vi sono 10 eventi e l'utente ne ha visti 7, la pagina mostrerà gli ultimi 3. Se l'utente ha già visto tutti gli eventi, gli verrà dato il messaggio "Niente di nuovo al momento".
Ovviamente ciascun utente avrà la propria vista individualizzata. La pagina utente dovrà avere una funzione che permetta di ripristinarne lo stato "dimenticando" gli eventi già visti, e quindi tornando allo stato iniziale.
Se non vi è un webcast attivo, la pagina mostra il messaggio: "Nessun evento disponibile ora".

3) Si curi la formattazione delle pagine.
Si evidenzino con un particolare stile le righe che contengono delle parole chiave (es: gol, rigore, espulso).

4) Per la condivisione di dati tra la parte di feeding e quella vista dall'utente si scelga la soluzione più efficiente (es., File System, Cookie, Session, Database, ServletContext, Variabili di istanza...). Si motivi brevemente la scelta fatta in un file HTML chiamato "motivazione.html" incluso nel progetto (la formattazione del file non è importante).

5) Si aggiunga un refresh automatico della pagina che venga effettuato ogni x secondi (default x=30), dove x è il valore presente in un campo numerico editabile presente nella pagina utente.
Quando il valore di x viene cambiato, la pagina deve cambiare la frequenza di aggiornamento rispettando tale valore.
Quando la pagina "dimentica" gli eventi già visti e torna allo stato iniziale, resetta anche il refresh automatico al valore di default.

Si consiglia di usare due diversi browser per testare le viste di più utenti.

Richiami sulla ServletContext:

ServletContext is an object from Servlet world. It is a shared repository for all servlets belonging to a web application. There is only one ServletContext per web application (so it is shared by all servets and all users).

The ServletContext object is the same as the "application" object in JSPs belonging to the same web application. (https://www.tutorialspoint.com/jsp/jsp_implicit_objects.htm)

It can be acquired by calling

```
getServletContext();
```

in the init, destroy, doGet, doPost, doFilter and all other doX methods. (https://docs.oracle.com/javaee/5/tutorial/doc/bnagl.html)

Once you have the ServletContext object, you can add to/retrieve from it objects like you do in an HttpSession, by using the methods

void setAttribute(String name, Object object)
Object getAttribute(String name)