VECCHI ESAMI TECNOLOGIE WEB

Esercizio 1 (11 punti) 10 settembre 2024

Si realizzi una applicazione Web per la gestione della coda in un ufficio comunale, basata su tecnologie Javascript, Java Servlet, JSP e Websocket.

L'applicazione Web deve permettere a un utente autenticato di aggiungersi alla coda di cittadini in attesa e di specificare se è interessato al rinnovo della sua carta di identità o alla pratica di cambio di residenza; in risposta deve ricevere una pagina Web con l'indicazione del presunto tempo di attesa. Il tempo di attesa dipenderà dal numero di cittadini già in coda secondo la seguente formula:

$$T_{attesa} = 5\ min * \#identita + 8\ min * \#residenza$$

dove #identita e #residenza sono ovviamente i numeri di cittadini già in attesa per le pratiche, rispettivamente, di rilascio carta di identità e cambio di residenza. Ogni volta che una pratica è stata terminata (tempo trascorso), la coda deve essere aggiornata server-side, senza necessariamente aggiornare i tempi di attesa mostrati ai clienti.

Nel caso in cui arrivi un cliente VIP (username=professore, password=vip), la sua pratica deve essere evasa immediatamente per prima; gli altri cittadini in attesa dovranno essere informati immediatamente con un aggiornamento del loro tempo di attesa.

Inoltre, visto che l'ufficio comunale chiude alle 12:30, si faccia in modo che quando la somma fra ora attuale e massimo tempo di attesa supera le 12:45, l'applicazione Web notifichi tutti i cittadini già in coda che non sarà accodata nessuna nuova richiesta ed eventuali nuovi cittadini in arrivo saranno informati di ritornare il giorno successivo.

Esercizio 2 (11 punti) 10 settembre 2024

Si realizzi una applicazione Web per il conteggio concorrente dei caratteri maiuscoli e minuscoli inclusi in due file txt di grande dimensione. L'applicazione deve essere basata esclusivamente sulle tecnologie Javascript, Ajax, Java Servlet e JSP (no Websocket).

In particolare, l'applicazione Web deve permettere a un utente non autenticato di inserire i nomi dei due file txt di input già presenti client-side. Vista la considerevole lunghezza dei due file di testo, il conteggio dei caratteri maiuscoli e minuscoli deve essere eseguito server-side e concorrentemente da tre thread separati: il cliente deve dividere ciascun file di testo in tre parti di dimensione analoga e inviare la prima parte dei due file al primo thread, la seconda parte dei due file al secondo thread, e così via; ovviamente ogni thread server-side determinerà localmente il numero di maiuscoli e minuscoli sulla propria parte. Al termine, tutti e sei i conteggi "locali" saranno inviati al cliente, che dovrà calcolare il numero totale dei caratteri processati, il numero totale dei maiuscoli contati e il numero totale dei minuscoli contati.

Inoltre, in ogni momento, tramite una servlet di amministrazione, previa autenticazione, l'amministratore di sistema deve poter visualizzare il numero totale dei caratteri processati, il numero totale dei maiuscoli contati e il numero totale dei minuscoli contati, considerando l'intero mese corrente.

Esercizio 3 (11 punti) 10 settembre 2024

Si realizzi in React un'applicazione Web lato client che simuli il gioco "Pesca al lago". L'applicazione dovrà eseguire interamente sul browser senza interagire con alcun server remoto.

L'interfaccia dell'applicazione sarà composta dalle seguenti sezioni:

- **Sezione Configurazione:** In questa sezione sono presenti due elementi di input per l'inserimento delle dimensioni del lago (rispettivamente, larghezza e lunghezza) e un elemento di input per il numero di lanci che possono essere effettuati dal giocatore in una battuta di pesca. I campi larghezza e lunghezza non possono assumere valori inferiori a 6. Il campo numero di lanci non potrà assumere un valore superiore a 4. Acquisiti tali dati, l'utente potrà iniziare a giocare.
- **Sezione Lago:** Tale sezione deve contenere una griglia rettangolare (larghezza x lunghezza) di celle di colore grigio. Ogni cella nasconde un numero casuale di pesci compreso tra 0 e 5. Il giocatore effettua un lancio cliccando su una determinata cella. Effettuato il lancio, sulla cella in questione (cella target) e su tutte le celle immediatamente adiacenti appaiono, e rimangono visibili per tutto il resto della battuta, i numeri corrispondenti di pesci contenuti (e quindi pescati), la cui somma costituirà il punteggio da attribuire al lancio. La cella target assumerà un colore verde, mentre le celle adiacenti assumeranno un colore giallo. Sia la cella target che le celle adiacenti non potranno più essere obiettivo di ulteriori lanci, pertanto, il giocatore potrà effettuare i successivi lanci su celle in cui non ha ancora pescato. Il gioco terminerà quando il giocatore avrà esaurito il numero di lanci a disposizione.
- **Sezione Punteggio:** In questa sezione, occorre visualizzare una lista aggiornata in tempo reale con i punteggi ottenuti dal giocatore per ciascun lancio e, solo alla fine della battuta di pesca, il punteggio complessivo.

Esercizio 1 (11 punti) 15 Luglio 2024

Si realizzi una applicazione Web per la gestione del tavolo da ping pong di uno stabilimento balneare, basata su tecnologie Javascript (anche Ajax), Java Servlet e JSP, senza fare uso di Websocket.

L'applicazione Web deve permettere a un utente autenticato di specificare se vuole solo vedere (lettura) lo stato delle prenotazioni del tavolo da ping pong oppure se intende avviare una nuova prenotazione (scrittura). In caso di lettura, l'applicazione deve semplicemente visualizzare lo stato corrente delle prenotazioni completate per il ping pong, aggiornandolo automaticamente ogni 3 secondi. In caso di scrittura, invece, un solo utente per volta deve avere il diritto di avviare una nuova prenotazione (indicandone l'orario di inizio; ogni prenotazione si intende per l'uso del tavolo per mezzora); la prenotazione si considererà completata e valida se e solo se entro 5 minuti un altro utente autenticato si aggiungerà come secondo giocatore per quella prenotazione. Si osservi che l'aggiornamento delle info visualizzate dai lettori può non essere effettuato immediatamente, bensì al prossimo intervallo "naturale" di aggiornamento.

Inoltre, si faccia in modo che, quando il numero di prenotazioni valide per partite non ancora svolte raggiunge il valore di 5, l'applicazione automaticamente consideri le successive nuove prenotazioni per un altro campo da ping pong (campo 2). Una volta sceso di nuovo il numero di prenotazioni a 2 per il campo 1, allora si potrà di nuovo convergere a usare un solo campo (campo 1), spostando le prenotazioni del campo 2 sull'unico campo 1 in utilizzo nei primi slot liberi disponibili.

Esercizio 2 (11 punti) 15 Luglio 2024

Si realizzi una applicazione Web per il calcolo concorrente del MAX e del min di una grande matrice quadrata di numeri reali i cui elementi sono contenuti all'interno di un file JSON di grande dimensione. L'applicazione deve essere basata esclusivamente sulle tecnologie Javascript, Ajax, Java Servlet e JSP (no Websocket).

In particolare, l'applicazione Web deve permettere a un utente non autenticato di inserire la dimensione della grande matrice da considerare (n, dove n sia maggiore di 100) e di fornire in input un file JSON contenente gli n^2 elementi reali della matrice, secondo il formato e l'ordine che si preferiscono. Inoltre, sempre a propria preferenza, per il file JSON si supporti o l'inserimento manuale in input o il caricamento da un file già salvato presso il cliente.

Vista la considerevole lunghezza del file JSON, la somma deve essere eseguita server-side e concorrentemente: il cliente deve dividere il file JSON in due parti di dimensione analoga e inviarle a una servlet per il calcolo concorrente server-side da parte di due thread distinti, ciascuno dei quali determinerà il MAX e min nel proprio subset di elementi. Tali MAX e min saranno inviati al cliente, che visualizzerà i soli MAX e min definitivi e globali, dopo avere ovviamente ricevuto i 4 risultati delle due esecuzioni concorrenti.

Inoltre, in ogni momento, tramite una JSP di amministrazione, previa autenticazione, l'amministratore di sistema deve poter visualizzare quanti numeri reali totali sono stati processati da tutte le richieste già servite nel giorno corrente.

Esercizio 3 (11 punti) 15 Luglio 2024

Si realizzi in React un'applicazione Web lato client che simula la ricerca di una sequenza ordinata di numeri interi. L'applicazione dovrà eseguire interamente sul browser senza interagire con alcun server remoto.

L'interfaccia dell'applicazione sarà composta dalle seguenti sezioni:

- **Sezione Gioco:** La sezione conterrà una griglia 3x3 di caselle di testo. In fase di avvio, la griglia verrà inizializzata riempiendo a caso le 9 caselle di testo con numeri interi da 1 a 9, senza ripetizioni. Il giocatore inizialmente non vedrà alcun numero contenuto nelle caselle, le quali assumeranno quindi il colore grigio (stato di casella coperta). Obiettivo del gioco è quello di scoprire, ad una ad una, tutte le caselle secondo la sequenza crescente dei numeri da 1 a 9. La scopertura di una casella avviene cliccando su di essa. Ovviamente, all'avvio del gioco, l'utente dovrà cercare in quale casella si nasconde il numero 1, e poi cercare in ordine il 2, il 3, etc. Quando l'utente clicca su una casella coperta contenente un numero in essa sequenza, la casella si colorerà di verde e visualizzerà il numero in essa contenuto. Se invece l'utente clicca su una casella coperta contenente un numero non in sequenza, la casella non cambierà di stato ed apparirà un messaggio di alert che mostra il numero contenuto della casella e invita l'utente a riprovare. Nulla accade se l'utente clicca su una casella già scoperta. Il gioco termina quando l'utente avrà scoperto tutte le caselle. Infine, la sezione contiene un bottone per il reset del gioco nonché le informazioni contenute nella sezione Punteggio.
- **Sezione Punteggio:** La sezione dovrà riportare i seguenti dati aggiornati in tempo reale: numero complessivo di tentativi di scopertura di caselle falliti; la sequenza di fallimenti consecutivi più lunga.

Esercizio 1 (11 punti) 24 giugno 2024

Si realizzi una applicazione Web per la gestione del tavolo da ping pong di uno stabilimento balneare, basata su tecnologie Javascript (anche Ajax), Java Servlet e JSP, senza fare uso di Websocket.

L'applicazione Web deve permettere a un utente autenticato di specificare se vuole solo vedere (lettura) lo stato delle prenotazioni del tavolo da ping pong oppure se intende avviare una nuova prenotazione (scrittura). In caso di lettura, l'applicazione deve semplicemente visualizzare lo stato corrente delle prenotazioni completate per il ping pong, aggiornandolo automaticamente ogni 3 secondi. In caso di scrittura, invece, un solo utente per volta deve avere il diritto di avviare una nuova prenotazione (indicandone l'orario di inizio; ogni prenotazione si intende per l'uso del tavolo per mezzora); la prenotazione si considererà completata e valida se e solo se entro 5 minuti un altro utente autenticato si aggiungerà come secondo giocatore per quella prenotazione. Si osservi che l'aggiornamento delle info visualizzate dai lettori può non essere effettuato immediatamente, bensi al prossimo intervallo "naturale" di aggiornamento.

Inoltre, si faccia in modo che, quando il numero di prenotazioni valide per partite non ancora svolte raggiunge il valore di 5, l'applicazione automaticamente consideri le successive nuove prenotazioni per un altro campo da ping pong (campo 2). Una volta sceso di nuovo il numero di prenotazioni a 2 per il campo 1, allora si potrà di nuovo convergere a usare un solo campo (campo 1), spostando le prenotazioni del campo 2 sull'unico campo 1 in utilizzo nei primi slot liberi disponibili.

Esercizio 2 (11 punti) 24 Giugno 2024

Si realizzi una applicazione Web per il calcolo concorrente della somma di due matrici di numeri interi i cui elementi sono contenuti all'interno di un file JSON di grande dimensione. L'applicazione deve essere basata esclusivamente sulle tecnologie Javascript, Ajax, Java Servlet e JSP (no Websocket).

In particolare, l'applicazione Web deve permettere a un utente non autenticato di inserire le dimensioni delle matrici considerate (a*b) e di fornire in input un file JSON contenente i (2*a*b) elementi interi delle due matrici, secondo il formato e l'ordine che si preferiscono. Inoltre, sempre a propria preferenza, per il file JSON si supporti o l'inserimento manuale in input o il caricamento da un file già salvato presso il cliente.

Vista la considerevole lunghezza del file JSON, la somma deve essere eseguita server-side e concorrentemente: il cliente deve dividere il file JSON in tre parti, circa della stessa dimensione e inviarle a una servlet per il calcolo concorrente server-side da parte di tre thread distinti. Solo dopo che tutti i risultati della matrice somma saranno ricevuti lato cliente, l'applicazione dovrà visualizzare tutta insieme l'intera matrice somma.

Inoltre, in ogni momento, tramite una servlet di amministrazione, previa autenticazione, l'amministratore di sistema deve poter visualizzare quante richieste di calcolo matriciale sono state già completate nel giorno corrente e la durata temporale di ciascuna di esse.

Esercizio 3 (11 punti) 24 Giugno 2024

Si realizzi in React un'applicazione Web lato client che simula una gara dei 100 metri. L'applicazione dovrà eseguire interamente sul browser senza interagire con alcun server remoto. L'interfaccia dell'applicazione sarà composta dalle seguenti sezioni:

- **Sezione Configurazione:** In questa sezione è presente un campo per l'inserimento del numero di corridori che gareggeranno. Occorre verificare che il numero inserito sia compreso tra 4 e 6;
- **Sezione Pista:** Questa sezione conterrà tante corsie orizzontali e rettilinee quanti sono i corridori che gareggeranno. Ogni corsia sarà composta da undici caselle di forma rettangolare e contigue. In ciascuna corsia, sulla prima casella apparirà la scritta "Partenza", sull'ultima casella la scritta "Traguardo", sulle caselle intermedie i numeri da "1" a "9"). All'inizio di ogni gara, le caselle "Partenza" di tutte le corsie sono idealmente occupate dai corridori in gara. Ogni corridore sarà identificato da un colore assegnato al corridore. L'utente avvia la gara tramite la pressione di un bottone. Avviata la gara, ogni 4 secondi i corridori progrediscono sulla propria corsia di un numero random di posizioni compreso fra 1 e 3. Quando, in seguito ad una progressione, un corridore si posiziona su una casella, lo sfondo di quest'ultima assume il colore assegnato al corridore. La gara terminerà quando l'ultimo dei corridori avrà tagliato il traguardo.
- **Sezione Classifica:** Questa sezione riporta la classifica finale della gara. Se due o più corridori tagliano il traguardo contemporaneamente, essi occuperanno ex aequo la stessa posizione in classifica. Infine, è presente un bottone per resettare il gioco.

Esercizio 1 (11 punti) 19 Aprile 2024 versione A

Si realizzi un'applicazione Web per la gestione di una lavagna condivisa basata su tecnologie Javascript (anche Ajax), Java Servlet e JSP, senza fare uso di Websocket.

L'applicazione Web deve permettere a un utente autenticato di specificare se in questa fase vuole leggere o scrivere sulla lavagna condivisa. In caso di lettura, l'applicazione deve semplicemente visualizzare il contenuto corrente della lavagna condivisa, aggiornandolo automaticamente ogni 10 secondi. In caso di scelta di scrittura, invece, un solo utente per volta deve averne il diritto: dopo avere superato tale controllo, l'utente può scrivere in qualsiasi punto della lavagna condivisa; alla pressione del pulsante "Invia" il contenuto della lavagna condivisa sarà aggiornato e l'utente tornerà a dover scegliere se è interessato a leggere o a scrivere. L'aggiornamento dei lettori può non essere effettuato immediatamente, bensì al prossimo intervallo "naturale" di aggiornamento.

Inoltre, si faccia in modo che quando il numero di utenti autenticati con sessioni attive raggiunge il valore di 50, l'applicazione automaticamente li divida in due gruppi da 25, ciascuno con la sua lavagna condivisa separata; le lavagne devono partire con valore iniziale uguale all'ultimo valore della iniziale lavagna condivisa, subito prima del group splitting.

Esercizio 2 (11 punti) 19 Aprile 2024 versione A

Si realizzi un'applicazione Web per la ricerca concorrente del cognome "Di Modica" all'interno di un file JSON di grande dimensione. L'applicazione deve essere basata esclusivamente sulle tecnologie Javascript, Ajax, Java Servlet e JSP (no Websocket).

In particolare, l'applicazione Web deve permettere a un utente non autenticato di fornire in input un file JSON. A propria preferenza, si supporti o l'inserimento manuale in input o il caricamento da un file già salvato presso il cliente. Il file JSON contiene una lunga sequenza di dati relativi a persone, con classica struttura per contenere i dati anagrafici di una persona; in questa struttura deve esistere un campo "Cognome", che ovviamente risulterà quello sui cui valori deve essere effettuata la ricerca.

Vista la considerevole lunghezza del file, la ricerca deve essere eseguita server-side e concorrentemente: il cliente deve dividere il file JSON in tre parti, circa della stessa dimensione e inviarle a una servlet per la ricerca concorrente server-side. Il risultato della ricerca concorrente deve essere aggregato lato cliente e, alla fine, l'applicazione deve visualizzare all'utente: "Sono state trovate x+ y + z = TOT occorrenze di Di Modica", dove x, y, z sono ovviamente i conteggi parziali di ciascuna delle esecuzioni concorrenti.

Inoltre, in ogni momento, tramite una JSP di amministrazione, previa autenticazione, l'amministratore di sistema deve poter visualizzare tutte le richieste in corso e il numero di richieste già completate per ogni sessione attiva.

Esercizio 3 (11 punti) 19 Aprile 2024 versione A

Si realizzi in React un'applicazione Web lato client per la gestione della rubrica telefonica. L'applicazione dovrà eseguire interamente sul browser Web, senza interagire con alcun server remoto.

L'interfaccia dell'applicazione sarà composta dalle seguenti sezioni:

- **Sezione "Ricerca"**: Qui sono presenti un campo di testo e un bottone che avvia la ricerca di tutti i contatti in rubrica il cui nome inizia per la stringa inserita nel campo di testo;
- **Sezione "Display"**: Qui compare l'elenco ordinato delle entry individuate in seguito a una ricerca. Ogni entry è composta da due campi di testo, che conterranno rispettivamente nome e numero di telefono, e da un bottone che l'utente potrà utilizzare per cancellare il nome dalla rubrica;
- **Sezione "Aggiunta nome"**: Questa sezione contiene due campi di testo e un bottone che il giocatore dovrà cliccare. All'azione del bottone dell'aggiunta, occorre verificare se il contatto non sia già presente in rubrica. Qualora il contatto fosse presente, l'applicazione dovrà chiedere all'utente conferma se intende sovrascrivere o meno il nuovo numero.

All'avvio dell'applicazione, la rubrica verrà inizializzata con un elenco statico di dieci contatti, direttamente cablati nel codice sviluppato. Inoltre, il display deve apparire vuoto.

Esercizio 1 (11 punti) 19 Aprile 2024 versione B

Si realizzi una applicazione Web per la gestione di un carrello condiviso basata su tecnologie Javascript (anche Ajax), Java Servlet e JSP, senza fare uso di Websocket.

L'applicazione Web deve permettere a un utente autenticato di specificare se in questa fase vuole modificare o semplicemente visualizzare il carrello condiviso. In caso di sola visualizzazione, l'applicazione deve semplicemente scaricare il contenuto corrente del carrello (in formato JSON), aggiornandolo automaticamente ogni 10 secondi. In caso di scelta di modifica carrello, invece, un solo utente per volta deve averne il diritto: dopo avere superato tale controllo, l'utente può inserire/rimuovere item dal carrello; alla pressione del pulsante "Invia" il contenuto del carrello sarà ufficialmente aggiornato e l'utente tornerà a dover scegliere se è interessato a modificare o visualizzare. L'aggiornamento degli utenti "visualizzatori" può non essere effettuato immediatamente, bensì al prossimo intervallo "naturale" di aggiornamento.

Inoltre, si faccia in modo che quando il numero di utenti autenticati con sessioni attive raggiunge il valore di 50, l'applicazione automaticamente li divida in due gruppi casuali da 25, ciascuno con il suo carrello condiviso separato; i carrelli devono partire con valore iniziale complessivo uguale all'ultimo valore dell'iniziale carrello condiviso subito prima del group splitting, ovvero dividendosi gli item presenti in modo circa uguale.

Esercizio 2 (11 punti) 19 Aprile 2024 versione B

Si realizzi una applicazione Web per la ricerca concorrente della via "Ugo Bassi" all'interno di un file JSON di grande dimensione. L'applicazione deve essere basata esclusivamente sulle tecnologie Javascript, Ajax, Java Servlet e JSP (no Websocket).

In particolare, l'applicazione Web deve permettere a un utente non autenticato di fornire in input un file JSON. A propria preferenza, si supporti o l'inserimento manuale in input o il caricamento da un file già salvato presso il cliente. Il file JSON contiene una lunga sequenza di dati relativi a persone, con classica struttura per contenere i dati anagrafici di una persona; in questa struttura deve esistere un campo "Indirizzo", che ovviamente risulterà quello sui cui valori deve essere effettuata la ricerca.

Vista la considerevole lunghezza del file, la ricerca deve essere eseguita server-side e concorrentemente: il cliente deve dividere il file JSON in due parti, circa della stessa dimensione, e inviarle a una servlet per la ricerca concorrente server-side. Il risultato della ricerca concorrente deve essere aggregato lato cliente e, alla fine, l'applicazione deve visualizzare all'utente: "Sono state trovate a+b= TOT occorrenze di Ugo Bassi negli indirizzi", dove a, b sono ovviamente i conteggi parziali di ciascuna delle esecuzioni concorrenti.

Inoltre, in ogni momento, tramite una JSP di amministrazione, previa autenticazione, l'amministratore di sistema deve poter visualizzare, per tutte le sessioni attive, quante richieste sono state già completate e quanto tempo è durata ciascuna di esse.

Esercizio 3 (11 punti) 19 Aprile 2024 versione B

Si realizzi una applicazione Web per la disseminazione probabilistica di informazioni in un gruppo, basata su tecnologia Websocket.

L'applicazione Web deve consentire a un utente non autenticato di visualizzare tutti i messaggi ricevuti dall'inizio della sua sessione di interazione e di inviare un messaggio agli altri utenti correntemente connessi. Nel caso di invio, il messaggio inviato deve essere propagato solo al 10% degli utenti connessi, scelti in modo casuale. I messaggi scambiati siano relativi a risultati di partite di Champions League e rappresentati in formato JSON.

Inoltre, in ogni momento, tramite una JSP di amministrazione, previa autenticazione, si consenta all'amministratore di sistema di poter effettuare il push di tutti i messaggi gestiti dall'applicazione dall'inizio della sua esecuzione a tutti gli utenti correntemente connessi, per riallineare le informazioni già probabilisticamente disseminate in precedenza.

Esercizio 1 (11 punti) 14 Febbraio 2024 - Versione B

Si realizzi una applicazione Web per il conteggio di lettere e numeri in un file server-side, basandosi esclusivamente sulle tecnologie Javascript (no Ajax), Java Servlet e JSP.

L'applicazione Web deve permettere a un utente autenticato di specificare il nome di un file server-side e se intende contarne il numero di caratteri alfabetici o numerici. Terminato l'inserimento dei dati di input con la stringa speciale "£££" in fondo al nome del file, senza pressione di alcun pulsante, i dati di input devono essere trasferiti al server in formato JSON.

Lato servitore, una servlet S1 deve controllare se quell'utente non ha già effettuato più di 100 richieste nella sua sessione di interazione; in caso positivo, deve girare la richiesta a una JSP J1 in caso di conteggio di lettere o a una JSP Jn nel caso di conteggio di cifre numeriche. Ogni JSP deve conteggiare il suo tipo di caratteri nel file, se non fatto precedentemente (si veda in seguito); effettuato il conteggio, tale risultato deve essere restituito al cliente e all'inizio del file dovrà essere aggiunta la riga "Il numero di caratteri alfabetici (o numerici) del file è uguale a x". Quindi, in apertura di ogni file potrebbero essere contenute 0, 1 o 2 righe con gli eventuali conteggi effettuati in precedenza; nel caso le righe esistano, la JSP non deve effettuare il conteggio; ovviamente si noti che diversi clienti possono richiedere di lavorare sul medesimo file concorrentemente.

Inoltre, si faccia in modo che l'amministratore dell'applicazione Web possa cancellare in ogni istante una riga di ogni file server-side specificando la posizione n-esima della riga stessa; ovviamente tale cancellazione renderà non più validi gli eventuali conteggi effettuati in precedenza dalle servlet per quel file.

Esercizio 2 (11 punti) 14 Febbraio 2024 - Versione B

Si realizzi una applicazione Web per il calcolo concorrente della differenza fra due matrici quadrate 4*4 di numeri naturali inferiori a 500, basandosi esclusivamente sulle tecnologie Javascript, Ajax, Java Servlet e JSP.

L'applicazione Web deve permettere a un utente non autenticato di inserire tutti gli elementi delle due matrici A e B. Dopo gli opportuni controlli locali sui dati inseriti, il cliente Web deve inviare le matrici A e B al servitore per il calcolo concorrente della matrice differenza: lato server, un thread separato deve occuparsi della differenza fra le matrici A e B considerando una sola riga di propria competenza; inoltre, si deve calcolare il tempo complessivo necessario per il processamento server-side. La matrice differenza risultante deve essere trasmessa al cliente insieme al tempo di processamento server-side; questi risultati saranno mostrati nella pagina Web del cliente e trasferiti server-to-client in formato JSON.

Inoltre, in ogni momento, tramite una JSP di amministrazione, previa autenticazione, l'amministratore di sistema deve poter visualizzare tutte le sessioni in corso ed eventualmente selezionarne una per comandarne la sua interruzione immediata.

Esercizio 3 (11 punti) 14 Febbraio 2024 - Versione B

Si realizzi una applicazione Web per il gioco della tombola, basandosi principalmente sulle tecnologie Javascript, Java Servlet e Websocket.

L'applicazione Web deve permettere a un utente non autenticato di chiedere di partecipare a un tavolo di tombola natalizio; l'applicazione deve supportare l'esecuzione di un tavolo alla volta e a ogni tavolo possono partecipare al massimo 5 giocatori. Dopo essere entrato con successo in un tavolo, il giocatore dovrà attendere fino a che il tavolo non sarà completo; il tavolo si considera completo se sono presenti già 5 giocatori ammessi al tavolo oppure se sono trascorsi 5 minuti dalla richiesta di accesso dell'ultimo giocatore ammesso.

A tavolo completo, la partita di tombola comincia e il server dovrà estrarre un nuovo numero random ogni 10 secondi, comunicandolo a tutti i partecipanti. Ogni partecipante avrà a disposizione un bottone per segnalare di avere fatto tombola, interrompendo così la sua partita, e un bottone per ritirarsi (in questo caso la partita proseguirà con gli altri giocatori rimasti). Non sono previsti controlli automatici sulla veridicità della dichiarazione del giocatore che afferma di avere fatto tombola.

Al termine della partita di tombola si preveda che il server avvii una nuova partita, che dovrà collezionare nuovi giocatori prima di cominciare.

Inoltre, in ogni momento, tramite una JSP di amministrazione, previa autenticazione, l'amministratore di sistema deve poter inviare messaggi di update ai giocatori di una partita selezionata, ad esempio per notificarli del cambiamento del risultato di un match del campionato di calcio ("aggiornamento push in diretta").

Esercizio 1 (11 punti) 30 gennaio 2024 versione C

Si realizzi un'applicazione Web per la gestione della stampa di file da parte di gruppi diversi di utenti, basandosi esclusivamente sulle tecnologie Javascript (no Ajax), Java Servlet e JSP.

L'applicazione Web deve permettere a un utente autenticato di scegliere quanti nomi di file inserire (da 1 a 4) e poi di inserire tali nomi; i file si suppongano memorizzati lato servitore, dove devono essere mandati in stampa. Terminato l'inserimento dei nomi di file previsti (si eseguano lato cliente gli opportuni controlli), senza pressione di alcun pulsante, i dati di input devono essere trasferiti al server in formato JSON.

Gli utenti appartengono a uno dei seguenti gruppi: 1) gruppo professori, che può stampare fino a 1000 pagine al giorno per l'intero gruppo; 2) gruppo dottorandi, che può stampare fino a 100 pagine al giorno per l'intero gruppo. Lato server, una JSP J1 deve calcolare la lunghezza di ogni file in termini di numero di pagine (si supponga per semplicità che 100 byte di lunghezza del file equivalgano a una pagina); se il numero di pagine di un file è superiore a 10, allora il file deve essere stampato fronte e retro, e lo svolgimento dell'operazione deve essere delegato alla servlet S1; negli altri casi, invece, il file deve essere stampato solo fronte, e lo svolgimento dell'operazione deve essere delegato alla servlet S2. Ogni servlet emulerà la stampa semplicemente restituendo direttamente al cliente la risposta "File stampato: è lungo pagine", con che rappresenta la lunghezza del file sopra definita.

Inoltre, si faccia in modo che l'amministratore della stampante possa in ogni istante cambiare la configurazione del numero di pagine limite per le due categorie; in tal caso, il cambiamento si deve applicare a partire da 24 ore dopo la modifica di configurazione.

Esercizio 2 (11 punti) 30 gennaio 2024 versione C

Si realizzi una applicazione Web capace di emulare il comportamento di un Web server con modello push, tramite schema di interazione "Forever Response" (secondo la terminologia usata a lezione). L'applicazione sarà basata esclusivamente sulle tecnologie Javascript, Ajax, Java Servlet e JSP; NON potrà utilizzare Websocket.

In particolare, l'applicazione Web deve emulare il comportamento di un sito Web di un giornale online: dopo la prima richiesta di un utente non autenticato, lo stesso utente dovrà vedere aggiornata la propria single page application di visualizzazione ogniqualvolta ci sarà una nuova news; le nuove news possono essere inserite dai giornalisti, previa autenticazione, tramite un semplice form che permetta l'input di un testo libero.

Inoltre, in ogni momento, tramite una JSP di amministrazione, il direttore del giornale può intervenire e cancellare l'ultima news inserita da un giornalista perché ritenuta non compatibile con le policy editoriali del giornale.

Esercizio 3 (11 punti) 30 gennaio 2024 versione C

Si realizzi una applicazione Web basata su tecnologia Websocket per il monitoraggio, il controllo e la gestione di dispositivi IoT.

In particolare, ogni dispositivo IoT sarà un cliente Web che deve inviare, ogni 60 secondi, un dato di monitoraggio della temperatura (un numero reale estratto a caso nell'intervallo [15, 25]) e il valore medio delle temperature misurate dall'inizio della sua esecuzione.

L'applicazione Web lato server dovrà invece occuparsi di due funzionalità differenti su due canali logici distinti. La prima funzionalità sarà di inviare ogni ora ai dispositivi IoT collegati la media globale di tutte le temperature misurate da tutti i dispositivi collegati nell'ora appena trascorsa. La seconda funzionalità sarà quella di permettere a un amministratore autenticato di selezionare un dispositivo specifico e di inviargli un comando con urgenza, ad esempio un comando di spegnimento o di disconnessione. Alla ricezione del comando il dispositivo IoT deve interrompere immediatamente il suo comportamento "normale" per eseguire con priorità il comando ricevuto (per semplicità si supponga che la ricezione del comando debba comportare la fine della sessione di interazione per quel dispositivo).