Diario di lavoro

|  |  |
| --- | --- |
| Luogo | Scuola d’arti mestieri Trevano |
| Data | 08.10.2020 |

|  |
| --- |
| Lavori svolti |
| Oggi ho aggiustato il problema che era legato ai file SQL (file di backup) che venivono salvati nella cartelle sbagliate. In effetti ho modificato il metodo runBackup del models backup\_model.  Immagine che contiene testo  Descrizione generata automaticamente  La prima linea di codice è il percorso della cartella nel quale vengono salvati i file SQL del collegamento al db. Il percorso è relativo alla cartella di root del progetto (gestione\_backup). Se la cartella non esiste viene creata un nuova cartella con il nome del collegamento al db. Per creare un cartella viene usato la funzione mkdir. Nella variabile $path\_file viene salvato il percorso del file, cioè dove deve essere salvato il file. Il percorso è però relativo alla cartella corrente. Ad esempio se creo un nuovo collegamento al datbase accesso, la cartella accesso in ./application/sources/backup non esiste e quindi viene creata. Inseguito viene eseguito il backup del database e il file SQL verrà salvato nella cartella ./application/sources/backup/accesso.  Invece il ciclo whiel che esegue i backup per tutti i collegamenti non era un problema,ma mi sono reso conto che l’utente Admin (l’unico utente aministratore del database), non aveva alcuni permessi DDL sul database e quindi non poteva creare, modificare o eliminare dei database. Per questo motivo nel file SQL non c’era tutto il codice del mysqldump. Per risolvere ho aperto MySQL workbench e nelle opzioni di Management/Users and Privileges posso modificare i permessi per l’utente Admin.  Immagine che contiene screenshot, monitor, sedendo, lato  Descrizione generata automaticamente  Poi mi sono reso conto che quando caricavo la pagina gestione i backup venivono eseguiti automaticamente, questo perché questa la funzione linkDBBackup viene eseguito all’interno dell’index. Quindi ho creato un nuovo controller e la views che si chiama backup. Nel controller backup ho copiato il metodo linkDBBackup e quest’ultimo lo richiamo del metodo index. Inseguito ho modificato il mio cron in modo che esegue ogni giorno (tranne sabato e domenica) a mezzanotte il backup giornaliero rispetto al sito http://localhost/gestione\_backup/backup/index.php.  Il comando crontab è il seguente:  0 0 \* \* 1-5 /usr/bin/curl –silent –compressed http://localhost/gestione\_backup/backup/index.php  Inseguito ho modificato il codice del metodo checkAggiungiUtenti del controller aggiungi\_utente. In effetti quando cliccavo il button submit del form della views aggiungi\_utente non mi inseriva ancora nella tabella utenti il nuovo utente. Un altro problema era anche il metodo getUserByUsername del models utente\_model che non funzionava del tutto.  Immagine che contiene testo  Descrizione generata automaticamente  Nella prima condizione, controllo che tutti i campi del form non siano vuoti. Se i campi non sono vuoti posso salvo nelle variabili $nome, $cognome, $email, $pswd e $tipo il valore che ha inserito l’utente. Inseguito mi collego al database gestione\_backup per potere andere a leggere la tabella utenti, in effetti richiamo il metodo getUserByUsername del models utente\_model che va a leggere i dati della tabella utenti filtrando per username. Se trova un utente che ha lo stesso username ricarica la pagina aggiungi\_utente, altrimenti salva nella variabile $username il valore che ha inserito l’utente. Per finire inserisco nella tabella utenti il nuovo utente, in effetti richiamo il metodo addUser del models utente\_model passandoli anche tutti i parametri (nome, cognome, username, password, email, tipo).  Oltre a questi problemi ho modificato la pagina utente dove ho aggiunto un nuovo campo nella tabella utenti che si chiama elimina. Per ogni utenti (tranne quello che ha eseguito il login) viene creato un pulsante che permette di eliminare l’utente.  Immagine che contiene testo  Descrizione generata automaticamente    Quando clicco il pulsante elimina mi richiama il metodo eliminaUtente del controller utente. Viene anche passato al metodo il parametro id che corrisponde all’id dell’utente.  Immagine che contiene testo  Descrizione generata automaticamente  Al suo interno richiamo il metodo deleteUser del models utente\_model che esegue una query che permette di eliminare un utente rispetto al suo id. |

|  |
| --- |
| Problemi riscontrati e soluzioni adottate |
| Oggi non ho riscontrato dei problemi che mi hanno fatto perdere del tempo. In effetti ho risolto tutti i problemi che avevo mercoledì. |

|  |
| --- |
| Punto della situazione rispetto alla pianificazione |
| Sono ad un buon punto della pianificazione, perché ho ricoperato l’attività nella quale dovevo creare il sistema di backup. In effettio ho risolto tutti i problemi che avevo ieri. Come ad esempio il percorso della cartella nel quale vengono salvati i file SQL e i permessi DDL per potere creare, eliminare o aggiornare un database. Questi permessi sono obbligatori se devo fare delle operazioni su un database. L’ultima attività del GANTT che devo recuperare sono la creazione dei file log. |

|  |
| --- |
| Programma di massima per la prossima giornata di lavoro |
| Per la prossima giornata di lavoro devo terminare la creazione dei file log. Ogni volta che creo un file SQL (file di backup) devo anche creare il suo file log corrispettivo. Una volta che ho terminato questa attività posso creare la pagina visualizza\_backup nel quale un utente può vedere tutti i backup che sono stati eseguiti. Nella pagina vengono visualizzati i file SQL + file log. L’utente deve avere la possibilità di crere un file report cliccando su un button. Nella pagina viene anche creato un segnale che permette di definire lo stato del backup. |