Diario di lavoro

|  |  |
| --- | --- |
| Luogo | Scuola d’arti mestieri Trevano |
| Data | 09.10.2020 |

|  |
| --- |
| Lavori svolti |
| Oggi mi sono reso conto che per eseguire i backup giornalieri non conviene utilizzare crontab perché è un utility esterno al sito ed è solamente accesibile dai sistemi unix. Quindi questo sistema non è multipiattaforma. Ho crecato su internet altre soluzioni che permettesse di realizzare un semplice task scheduler in php. Un task scheduler è un utility che permette di eseguire dei programmi o dei script in periodi predefiniti o dopo intervali di tempo specificati. Ad esempio crontab permette di realizzare un task scheduler su sistemi operativi unix. Una soluzione è realizzare in SQL un trigger che permetta di realizzare questo task. I trigger sono degli oggetti di SQL Sever molto simili alle stored procedures, una sorta di procedura particolare che si attiva automaticamente dopo un determinato evento. Dopo quasi due ore di ricerca per capire come creare i trigger, mi sono reso conto che i trigger non potevo realizzare questo task scheduler, perché il comando mysqldump viene eseguito via web dal metodo php exec (il metodo exec esegue dei comandi shell). Nell’applicativo vengono creati questi file SQL che vengono poi utilizzati per fare il restore di un database in caso di perdita di dati o del stesso database. Visto che non ho trovato una soluzione ho smesso la ricerca e ho iniziato la pagina visualizza\_backp. In questa pagina un utente può visualizzare il contenuto dei file SQL e dei file log di ogni collegamento. L’utente ha anche la possibilità di cliccare un pulsante che permette di realizzare un report del collegamento in formato pdf. Per finire nel sito per ogni collegamenti viene mostarto un segnale che indica lo stato dl backup. Prima di tutto ho aggiornato il metodo runBackup del model backup\_model. Ho aggiunto al comando mysqldump l’opzione --databases  che permette di aggiungere il codice che crea il database, in modo che se elimino il database c’è la possibilità di realizzarlo. Ho aggiunto anche l’opzione --verbose che permette di visualizzare le informazioni durante il processo di backup, in modo che possa scrivere quest’ultime su un file log. Dopo il segno >, ho aggiunto il percorso del file log, così durante il processo di backup viene scritto tutti i pasaggi che permettono di realizzare il file SQL. Oltre a creare la cartella nel quale ci sono i file SQL creao anche un’altra cartella nella quale vengono salvati i file log. Per ogni collegamenti c’è un cartella che contiene i file SQL e un’altra cartella che contiene i file log.  Immagine che contiene testo  Descrizione generata automaticamente  In effetti nella condizione se non esiste la directory del collegamento, oltre a creare la directory del collegamento creo al su interno la directory log. All’interno della variabile $path\_log salvo il percorso completo dei file log. Per verificare che il metodo funzioni ho aperto la pagina http://localhost/gestione\_backup/backup in modo che potesse creare i backup. Poi ho realizzato sempre nel model backup\_model un nuovo metodo che si chiama viewBackupFile che permette di visaulizazre il contenuto della cartella di backup del collegamento. Ad esempio posso vedere il contenuto dell cartella accesso (collegamento del db accesso).  Immagine che contiene testo  Descrizione generata automaticamente  Nlla prima condizione verifico se rieco aprire la directory (magari la directory non esiste). Per aprire una cartella in php viene utilizzato il metodo opendir. Con il metodo scandir in php posso vedere il contenuto di una cartella. Questo metodo mi serve per sapere quali file ci sono all’interno della directory. Con il metodo array\_diff posso eliminare un elemento che è presente all’interno della cartella. Non viene cancellato fisicamente gli elementi nella directory, ma scandir ritorna un array di elementi e con il metodo array\_diff posso mantenere hli elementi che mi interessano. In questo caso tengo solo i file all’interno dell’array. Creo du variabili che sono $row e $row\_log. La prima variabile è un array che contiene i file SQL. Invece la seconda variabile è un array che contiene i file log. Per recuperare l’ultimo elemento dell’array $row utilizzo il metodo count che conta il numero di elementi di un array. |

|  |
| --- |
| Problemi riscontrati e soluzioni adottate |
| Per eseguire i backup giornalieri non conviene utilizzare crontab perché è un utility esterno al sito ed è solamente accesibile dai sistemi unix. Quindi questo sistema non è multipiattaforma. Ho perso molto tempo su internet e non ho trovato una soluzione che potrebbe funzionare. Per ora utilizzo ancora un crontab. |

|  |
| --- |
| Punto della situazione rispetto alla pianificazione |
| Oggi ho realizazto parzialmente le attività che avevo previsto di realizzare. Per esempio ho realizzato i file log che vengono generati durante il backup. Non potuto realizzare la views visualizza\_backup perché ho perso del tempo nel cercare su internet una soluzone che permette di realizazre un sistema di task scheduler. Ma per finire ho già implementato un metodo che mi serve per questa pagina. |

|  |
| --- |
| Programma di massima per la prossima giornata di lavoro |
| Per la prossima giornata di lavoro devo trovare una soluzione che permette di implementare un task scheduler in php. 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. Se resta del tempo posso andare avanti con la pagina di gestione degli utenti. |