Diario di lavoro

|  |  |
| --- | --- |
| Luogo | Scuola d’arti mestieri Trevano |
| Data | 10.11.2020 |

|  |
| --- |
| Lavori svolti |
| Oggi ho modificato il models password. All’interno di questa classe ho creato due metodi che si chiamano verify e encode.  Immagine che contiene testo  Descrizione generata automaticamente  Il metodo encode permette di codificare la password in hash. Al metodo viene passato come parametro $pswd che corrisponde alla password in chiaro. La funzione password\_hash permette di generare una nuova password hash utilizzando un algoritmo di hashing unidirezionale. Questa funzione accetta due parametri:   * Il primo parametro corrisponde al password in chiaro * Il secondo parametro corrisponde al tipo di algoritmo. Come algoritmo utilizzo l’agoritmo bcrypt di default. Questo algoritmo può espandersi oltre i 60 caratteri. Quest’ultimo e momorizzato nella costante PASSWORD\_DEFAULT.   Immagine che contiene testo  Descrizione generata automaticamente  Il metodo verify permette di verificare se la password passata in chiaro corrisponde alla password hash. Questa funzione viene realizzata dal metodo password\_verify.  Quando l’utente aggiunge un nuovo utente viene richiamato il metodo checkAggiungiUtente del models aggiungi\_utente. All’interno di questo metodo viene richiamato il metodo addUser del models utente. In effetti al metodo viene anche passato la password codificata in hash.    Quando un utente esegue il login viene richiamato il metodo checkLogin, quest’ultimo richiama il metodo getUtente del models login\_model. In effetti ho modificato il metodo getUtente aggiungendo anche il controllo che mi permette di verifcare che la password inserita corrisponde alla password hash. Per realizzarlo ho utilizzato il metodo verify del models password.    Oggi ho anche terminato il sistema che permette di cambiare le password provvisorie. Quando un utente esegue il login viene anche letto dalla tabella utenti lo stato del login. Ho modificato la tabella utenti aggiungendo un nuova colonna che si chiama statusLogin. Quando viene creato un nuovo utente lo stato del login è -1 (ho impostato come default -1). Per aggiungere una nuova colonna che momorizza lo stato del login alla tabella utenti ho realizzato il seguente codice SQL.    Sempre nel metodo checkLogin viene controllato lo stato del login dell’utente. Se lo stato è -1 vengono viene richiamato la views modifica\_password, altrimenti la views visualizza\_backup.    La views modifica\_password viene solamente visualizzata se sono state create le sessioni status e id-utente, altrimenti viene richiamato sempre la pagina di login. All’interno della sessione id-utente viene salvato l’id dell’utente corrente (ogni utente è identificato da un id). Nel form l’utente può inserire la nuova password e deve anche confermare la password. Quando viene cliccato il pulsante submit viene richiamato il metodo modifiyPassword del models modifica\_password.  Immagine che contiene testo  Descrizione generata automaticamente  Allinterno verifico che la password di conferma corrisponde alla nuova password con semplice selettore. All’interno richiamo il metodo updatePasswordUtente del models login\_model. Quest’ultimo permette di cambiare lo stato del login di un determinato utente (ogni utente è identificato da un id) e di cambiare la password provissoria. |

|  |
| --- |
| Problemi riscontrati e soluzioni adottate |
| Oggi non ho riscontrato dei problemi. |

|  |
| --- |
| Punto della situazione rispetto alla pianificazione |
| Sono ad un buon punto della situazione rispetto alla pianificazione. In effetti mi manca solamente il sistema di email se il sistema di ricerca. |

|  |
| --- |
| Programma di massima per la prossima giornata di lavoro |
| Per la prossima giornata di lavoro devo realizzare il sistema di ricerca. Quest’ultimo permette di scegliere la migliore soluzione per risolvere il problema dei backup. |