Diario di lavoro

|  |  |
| --- | --- |
| Luogo | Canobbio, SAM Trevano |
| Data | 03/12/2019 |

|  |
| --- |
| Lavori svolti |
| Per ogni pagina ho aggiunto dei bottoni che possono eseguire le informazioni principali sui dati del database. Le azioni principali che si possono eseguire con i bottoni sono la modifica, il salvataggio e l’eliminazione dei dati, invece per aggiungere i dati è presente un’interfaccia apposita all’inizio di ogni pagina.  Come esempio prendo la pagina per la gestione degli utenti, in questa pagina è presente un’interfaccia per aggiungere un nuovo utente e subito sotto la tabella con la lista degli utenti memorizzati sul database.  Sopra alla tabella sono presenti i bottoni che permetto di salvare e modificare tutti insieme i dati presenti nel database. Invece in parte ad ogni riga sono presenti i bottoni per modificare, salvare e eliminare un utente.  Se si schiaccia il bottone per modificare e dopo lo si rischiaccia i dati modificati non saranno salvati e i dati torneranno allo stato in cui erano prima della modifica.  Se si schiaccia il bottone salva dopo la modifica i dati non si potranno più modificare fino a quando non si schiaccia di nuovo il bottone modifica.  Per precauzione quando il bottone elimina viene schiacciato comparirà una piccola interfaccia, in cui si chiede la conferma per cancellare l’utente.    Queste azioni che vengono fatte sui dati sono state implementate in modo tale che quando vengano eseguite non si debba ricaricare la pagina, questo è stato possibile con l’utilizzo di AJAX per eseguire le azioni.  Le funzioni con AJAX sono state fatte in un file esterno apposito per ogni pagina. E in questo esempio  permette di salvare le informazioni di un utente.  Questa funzione prende le informazioni che sono presenti all’interno della pagina, e li manda sotto forma di JSON ad un metodo presente nel controller della stessa view con un POST.  let URL = "<?= URL ?>";  /\*\*  \* Update a user from its ID  \*  \* @param id The id of the user  \*/  function updateUser(id) {  var user = createUserArrayFromId(id);  var jsonUser = JSON.stringify(user);  $.ajax({  type: "POST",  url: URL + "users/updateUser",  data: {  user: jsonUser  },  success: function(response) {  if (response) {  console.log(response);  $(".user-" + id + "-field").prop("disabled", true);  }  }  });  }  Questo metodo aiuta il metodo precedente, permette di ricavare le informazioni di un certo utente dal suo ID, e le ritorna sotto forma di file JSON.  /\*\*  \* Get the user representation with an JSON,  \* the JSON is mapped name => value.  \*  \* @param id The id of the user  \*/  function createUserArrayFromId(id) {  var user = {};  $(".user-" + id + "-field").each(function(i, obj) {  user[this.name] = this.value;  });  user["enabled"] = $("#enabled" + id).is(":checked");  return user;  } |
|  |

|  |
| --- |
| Problemi riscontrati e soluzioni adottate |
| nessuno |

|  |
| --- |
| Punto della situazione rispetto alla pianificazione |
| Un po’ in ritardo rispetto alla pianificazione. |

|  |
| --- |
| Programma di massima per la prossima giornata di lavoro |
| Continuare l’implementazione, iniziare gestione fatturazioni e finire di sistemare le altre pagine. |