Diario di lavoro

|  |  |
| --- | --- |
| Luogo | Canobbio |
| Data | 02.03.2020 |

|  |
| --- |
| Lavori svolti |
| Il modo in cui ho implementato il menu non mi è piaciuto tanto quindi ho fatto un po’ di ricerca su quale è il modo migliore.    Esplorando il sito della documentazione del framework Vaadin ho scoperto che ci sono dei esempi di applicazioni che hanno un aspetto simile a quello che serve a me.  La loro implementazione si trova sul seguente sito: <https://vaadin.com/start/lts/simple-ui>  Nella loro implementazione utilizzano l’AppLayout come base della loro GUI. L’AppLayout consente di avere un “cassetto” (menu di lato) al di fuori delle view.  Screenshot from 2020-03-02 08-44-03  Figura 1 Menu di lato aperto  Screenshot from 2020-03-02 08-44-14  Figura 2 Menu di lato chiuso  Ho proseguito ad adattare il mio progetto alla loro soluzione. Dovevo apportare poche modifiche  La loro soluzione implementa un metodo migliore per il controllo dell’autenticazione grazie ad una classe che implementa l’interfaccia VaadinServiceInitListener. Questa classe sovrascrive un metodo che viene chiamato ogni volta che viene caricata una View, in questo metodo viene eseguito il controllo se l’utente ha fatto l’accesso. Se l’utente non è autenticato, esso viene reindirizzato verso la LoginView.  Le modifiche maggiori sono riportate qua di seguito:   * Per indicare a Vaadin la classe che implementa l’interfaccia VaadinServiceInitListener bisogna creare il file com.vaadin.flow.server.VaadinServiceInitListener sotto la cartella resources/Meta-INF/services/ contenente il nome canonico (package.className secondo le specifiche del linguaggio di Java)   Screenshot from 2020-03-02 09-01-41  Figura 3 File che indica la classe che implementa il VaadinServiceInitListener  La classe DeduplicatorGUIInitListener contiene il seguente codice che grazie all’AccessControlInterface controlla se l’utente ha eseguito l’accesso in questa sessione:  @Override  public void serviceInit(ServiceInitEvent initEvent) {  final AccessControlInterface accessControl = AccessControlFactory.getInstance()  .createAccessControl();  initEvent.getSource().addUIInitListener(uiInitEvent -> {  uiInitEvent.getUI().addBeforeEnterListener(enterEvent -> {  if (!accessControl.isUserSignedIn() && !LoginView.class  .equals(enterEvent.getNavigationTarget()))  enterEvent.rerouteTo(LoginView.class);  });  });  }   * Per l’effettivo controllo se l’utente è autenticato viene usato il metodo isUserSignedIn della classe AccessControl e la memoria della sessione dell’applicazione.  Se il metodo get della classe CurrentUser ritorna null, questo sta a indicare che nella memoria dell’applicazione non è presente il nome del utente, quindi non è stato eseguito l’accesso perciò l’utente viene reindirizzato alla pagina di Login.   String currentUser = (String) UI.getCurrent().getSession()  .getAttribute(Resources.CURRENT\_USER\_SESSION\_ATTRIBUTE\_KEY);   * Le view non hanno subito nessuna maggiore modifica   Ho finito al view dei rapporti che adesso è da testare. |

|  |
| --- |
| Problemi riscontrati e soluzioni adottate |
| Quando cercavo di aggiornare i valori dello stato della scansione ho riscontrato un problema dove veniva tirata un’eccezione indicando che volevo modificare la sessione senza bloccarla prima.  Caused by: java.lang.IllegalStateException: java.lang.IllegalStateException: Cannot access state in VaadinSession or UI without locking the session.  Ho trovato la soluzione sul seguente sito: <https://vaadin.com/docs/v14/flow/advanced/tutorial-push-access.html>  Ho risolto il problema il metodo access della UI che esegue il codice contenuto nel metodo execute della classe Command gestendo l’accesso alla sessione:  Command command = new Command() {  @Override  public void execute() {  updateStatus(true, Integer.parseInt(response.get("fileCount").toString()),  Long.parseLong(response.get("timestamp").toString()),  Float.parseFloat(response.get("progress").toString()));  }  };  getUI().get().access(command); |

|  |
| --- |
| Punto della situazione rispetto alla pianificazione |
| Sono indietro di 2 ore rispetto alla pianificazione. |

|  |
| --- |
| Programma di massima per la prossima giornata di lavoro |
| Finire i test della schermata dei rapporti |
| Documentare le nuove modifiche |
|  |