

## Java - User Story 4 - PDF

- Come Lorenzo
- vorrei poter cercare tra gli articoli
- in modo tale da visualizzare subito quello che mi interessa

### ACCEPTANCE CRITERIA:

- Implementazione della ricerca full-text
- Ricerca per titolo
- Ricerca per sottotitolo
- Ricerca per categoria

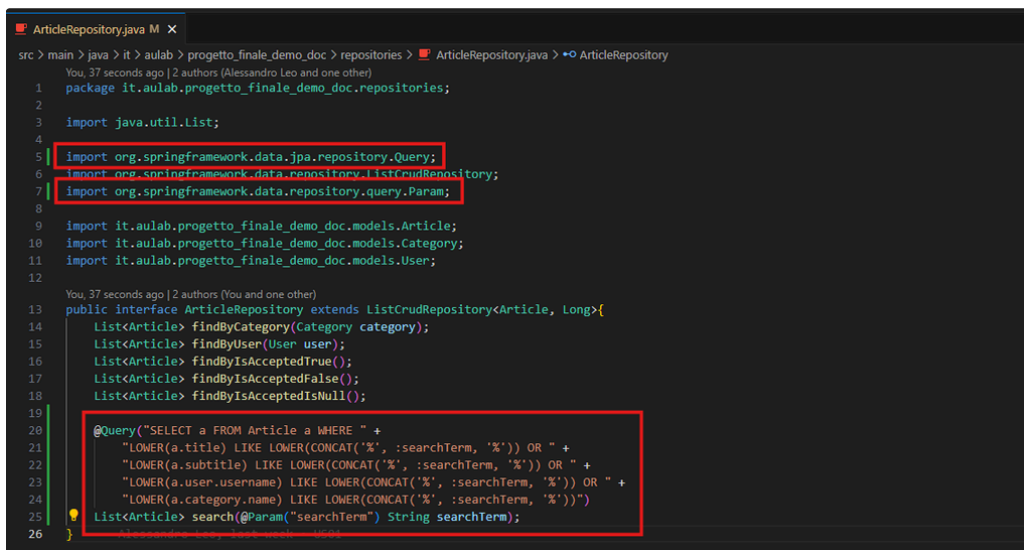
### Svolgimento

Con questa user story procederemo con lo sbloccare il funzionamento del form di ricerca che fino ad ora abbiamo sempre avuto sulla navbar ma mai reso attivo.

### QUERY JPQL

Per la ricerca utilizzeremo una query JPQL (Java Persistence Query Language) molto articolata ma che renderà la ricerca facile da effettuare dato che recupererà solo i dati che corrispondono al parametro di ricerca che invieremo tramite il form

in "src/main/java/it/aulab/progetto\_finale\_docente/repositories/ArticleRepository.java"



```
src > main > java > it > aulab > progetto_finale_demo_doc > repositories > ArticleRepository.java > ArticleRepository

You, 37 seconds ago | 2 authors (Alessandro Leo and one other)
package it.aulab.progetto_finale_demo_doc.repositories;

import java.util.List;

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.ListCrudRepository;
import org.springframework.data.repository.query.Param;

import it.aulab.progetto_finale_demo_doc.models.Article;
import it.aulab.progetto_finale_demo_doc.models.Category;
import it.aulab.progetto_finale_demo_doc.models.User;

You, 37 seconds ago | 2 authors (You and one other)
public interface ArticleRepository extends ListCrudRepository<Article, Long>{
    List<Article> findByCategory(Category category);
    List<Article> findByUser(User user);
    List<Article> findByIsAcceptedTrue();
    List<Article> findByIsAcceptedFalse();
    List<Article> findByIsAcceptedIsNull();

    @Query("SELECT a FROM Article a WHERE " +
        "LOWER(a.title) LIKE LOWER(CONCAT('%', :searchTerm, '%')) OR " +
        "LOWER(a.subtitle) LIKE LOWER(CONCAT('%', :searchTerm, '%')) OR " +
        "LOWER(a.user.username) LIKE LOWER(CONCAT('%', :searchTerm, '%')) OR " +
        "LOWER(a.category.name) LIKE LOWER(CONCAT('%', :searchTerm, '%'))")
    List<Article> search(@Param("searchTerm") String searchTerm);
}
```

La nostra JPQL può essere anche ampliata ma per il momento ci consente la ricerca per titolo, sottotitolo e categoria.

 **EXTRA: AMPLIARE LA JPQL PER MAGGIORI RICERCHE**

### SERVICE

Procediamo ora col creare il metodo nel service che utilizzerà questa query.

in “src\main\java\it\aulab\progetto\_finale\_docente\services\ArticleService.java”

```
ArticleService.java M X
src > main > java > it > aulab > progetto_finale_demo_doc > services > ArticleService.java > ArticleService > setIsAccepted(Boolean, Long)
26 public class ArticleService implements CrudService<ArticleDto, Article, Long>{
110     public List<ArticleDto> searchByAuthor(User user){
111         List<ArticleDto> dtos = new ArrayList<ArticleDto>();
112         for(Article article: articleRepository.findByUser(user)){
113             dtos.add(modelMapper.map(article, destinationType:ArticleDto.class));
114         }
115         return dtos;
116     }
117
118     public void setIsAccepted(Boolean result, Long id){
119         Article article = articleRepository.findById(id).get();
120         article.setIsAccepted(result);
121         articleRepository.save(article);
122     }
123
124     public List<ArticleDto> search(String keyword){
125         List<ArticleDto> dtos = new ArrayList<ArticleDto>();
126         for(Article article: articleRepository.search(keyword)){
127             dtos.add(modelMapper.map(article, destinationType:ArticleDto.class));
128         }
129         return dtos;
130     }
131
132 }
```

In questo metodo utilizziamo la query appena creata nel repository e mappiamo i risultati nei dto.

## CONTROLLER

Costruito il metodo nel service procediamo con l’handler nel controller che agganceremo al form di ricerca nella navbar

in “src\main\java\it\aulab\progetto\_finale\_docente\controllers\ArticleController.java”

```
ArticleController.java
src > main > java > it > aulab > progetto_finale_demo_doc > controllers > ArticleController.java > ...
You, yesterday | 2 authors (Alessandro Leo and one other)
1 package it.aulab.progetto_finale_demo_doc.controllers;
2
3 import java.security.Principal;
4 import java.util.ArrayList;
5 import java.util.Collections;
6 import java.util.Comparator;
7 import java.util.List;
8
9 import org.modelmapper.ModelMapper;
10 import org.springframework.beans.factory.annotation.Autowired;
11 import org.springframework.beans.factory.annotation.Qualifier;
12 import org.springframework.data.repository.query.Param;
13 import org.springframework.stereotype.Controller;
14 import org.springframework.ui.Model;
15 import org.springframework.validation.BindingResult;
16 import org.springframework.web.bind.annotation.GetMapping;
17 import org.springframework.web.bind.annotation.ModelAttribute;
18 import org.springframework.web.bind.annotation.PathVariable;
19 import org.springframework.web.bind.annotation.PostMapping;
20 import org.springframework.web.bind.annotation.RequestMapping;
21 import org.springframework.web.bind.annotation.RequestParam;
22 import org.springframework.web.multipart.MultipartFile;
23 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
24
25 import it.aulab.progetto_finale_demo_doc.dtos.ArticleDto;
26 import it.aulab.progetto_finale_demo_doc.dtos.CategoryDto;
27 import it.aulab.progetto_finale_demo_doc.models.Article;
28 import it.aulab.progetto_finale_demo_doc.models.Category;
29 import it.aulab.progetto_finale_demo_doc.repositories.ArticleRepository;
30 import it.aulab.progetto_finale_demo_doc.services.ArticleService;
31 import it.aulab.progetto_finale_demo_doc.services.CrudService;
32 import jakarta.validation.Valid;
33
34
35 You, yesterday | 2 authors (Alessandro Leo and one other)
36 @Controller
37 @RequestMapping("/articles")
38 public class ArticleController {
```

```

114 }
115
116 //Rotta dedicata all'azione del revisore
117 @PostMapping("/accept")
118 public String articleSetAccepted(@RequestParam("action") String action, @RequestParam("articleId") Long articleId, RedirectAttributes redirectAttributes) {
119
120     if(action.equals(anObject:"accept")){
121         articleService.setIsAccepted(result:true, articleId);
122         redirectAttributes.addFlashAttribute(attributeName:"resultMessage", attributeValue:"Articolo accettato!");
123     }else if(action.equals(anObject:"reject")){
124         articleService.setIsAccepted(result:false, articleId);
125         redirectAttributes.addFlashAttribute(attributeName:"resultMessage", attributeValue:"Articolo rifiutato!");
126     }else{
127         redirectAttributes.addFlashAttribute(attributeName:"resultMessage", attributeValue:"Azione non corretta!");
128     }
129
130     return "redirect:/revisor/dashboard";
131 }
132
133 //Rotta di ricerca di un articolo
134 @GetMapping("/search")
135 public String articleSearch(@Param("keyword") String keyword, Model viewModel) {
136     viewModel.addAttribute(attributeName:"title", attributeValue:"Tutti gli articoli trovati!");
137
138     List<ArticleDto> articles = articleService.search(keyword);
139     viewModel.addAttribute(attributeName:"articles", articles);
140
141     return "article/articles";
142 }
143
144 }
145

```

## NAVBAR

Ultimo step prima di poter effettuare un test è proprio quello di agganciare il controller appena creato con il form nella navbar.

in "src/main/resources/templates/index.html"

```

index.html M X
src > main > resources > templates > index.html > html > body > nav.navbar.navbar-expand-lg.navbar-light.bg-light > form.d-flex
2 <html xmlns:th="http://www.thymeleaf.org">
15 <body>
17 <nav th:fragment="navbar" class="navbar navbar-expand-lg navbar-light bg-light">
18 <div class="container-fluid">
23 <div class="collapse navbar-collapse" id="navbarSupportedContent">
31 <li class="nav-item" sec:authorize="hasRole('ROLE_ADMIN')">
32 <div class="d-flex">
33 <a class="nav-link" href="/admin/dashboard">Dashboard admin</a>
34 <div class="mt-1" th:if="${careerRequests > 0}">
35 <i class="fas fa-bell fa-1 pt-2"></i>
36 <span class="badge rounded-pill bg-danger text-white px-1" th:text="${careerRequests}></span>
37 </div>
38 </div>
39 </li>
40 <li class="nav-item" sec:authorize="hasRole('ROLE_REVISOR')">
41 <div class="d-flex">
42 <a class="nav-link" href="/revisor/dashboard">Dashboard revisor</a>
43 <div class="mt-1" th:if="${articlesToBeRevised > 0}">
44 <i class="fas fa-bell fa-1 pt-2"></i>
45 <span class="badge rounded-pill bg-danger text-white px-1" th:text="${articlesToBeRevised}></span>
46 </div>
47 </div>
48 </li>
49 <li class="nav-item dropdown">
50 <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">
51 Accesso
52 <a>
53 <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
54 <li sec:authorize="isAnonymous"><a class="dropdown-item" aria-current="page" th:href="@{/register}">Register</a></li>
55 <li sec:authorize="isAnonymous"><a class="dropdown-item" aria-current="page" th:href="@{/login}">Login</a></li>
56 <li sec:authorize="isAuthenticated"><a class="dropdown-item" aria-current="page" th:href="@{/logout}">Logout</a></li>
57 </ul>
58 </li>
59 </div>
60 <div sec:authorize="isAuthenticated" th:text="'Benvenuto: ' + ${#authentication.principal.fullname} + ' - ' + ${#authentication.principal.authorities[0].authority.replace('ROLE_', '')}>Benvenuto</div>
61 </div>
62 </div>
63 <form class="d-flex" th:action="@{/articles/search}" method="get">
64 <input class="form-control" type="text" placeholder="Cerca" name="keyword" aria-label="Search">
65 <button class="btn btn-outline-success" type="submit">Search</button>
66 </form>
67 </nav>
68 <!-- fine navbar -->

```

Fatto questo procediamo con un test e vedremo che la ricerca avviene secondo i criteri inseriti.

## SBLOCCO ROTTE IN SECURITY

Ma abbiamo un problema. La ricerca deve essere consentita anche ad utenti non registrati o loggati cosa che attualmente invece non avviene.

La modifica è molto semplice, basterà andare ad aggiungere i permessi nel nostro file di configurazioni di sicurezza

in "src/main/java/it/aulab/progetto\_finale\_docente/config/SecurityConfig.java" modifichiamo

```

SecurityConfig.java X
src > main > java > it > aulab > progetto_finale_demo_doc > config > SecurityConfig.java > SecurityConfig > filterChain(HttpSecurity)
18 @Configuration
19 @EnableWebSecurity
20 public class SecurityConfig{
21
22     @Autowired
23     private CustomUserDetailsService customUserDetailsService;
24
25     @Autowired
26     private PasswordEncoder passwordEncoder;
27
28     @Bean
29     public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
30         http
31             .csrf(csrf -> csrf.disable())
32             .authorizeHttpRequests(authorize -> {
33                 .requestMatchers(...patterns: "/register/**").permitAll()
34                 .requestMatchers(...patterns: "/admin/dashboard", "/categories/create", "/categories/edit/{id}", "/categories/update/{id}", "/categories/delete/{id}").hasRole("ADMIN")
35                 .requestMatchers(...patterns: "/aulab/dashboard", "/aulab/detail/{id}", "/accept").hasRole("VISITOR")
36                 .requestMatchers(...patterns: "/register", "/", "/articles", "/images/**", "/articles/detail/**", "/categories/search/{id}", "/search/{id}", "/articles/search").permitAll()
37                 .anyRequest().authenticated()
38             })
39         .formLogin(form -> {
40             form.loginPage(loginPage: "/login")
41             .loginProcessingUrl(loginProcessingUrl: "/login")
42             .defaultSuccessUrl(defaultSuccessUrl: "/")
43             .permitAll()
44         })
45         .logout(logout -> {
46             .logoutRequestMatcher(new AntPathRequestMatcher(pattern: "/logout"))
47             .permitAll()
48         })
49         .exceptionHandling(exception -> {
50             exception.accessDeniedPage(accessDeniedUrl: "/error/403")
51             .sessionManagement(session -> {
52                 session.creationPolicy(sessionCreationPolicy: SessionCreationPolicy.IF_REQUIRED)
53                 .maximumSessions(maximumSessions: 1)
54                 .expiredUrl(expiredUrl: "/login?session-expired=true")
55             })
56         })
57     }
58     return http.build();
59 }

```

Facciamo ancora un test e vedremo che anche da utenti non loggati ci verranno mostrati i risultati della ricerca.

## MOSTRARE SOLO GLI ARTICOLI ACCETTATI

Abbiamo anche qui un problema. Dobbiamo visualizzare nella ricerca solo gli articoli accettati.

Modifichiamo.

in "src\main\java\it\aulab\progetto\_finale\_doc\controllers\ArticleController.java"

```

ArticleController.java M X
src > main > java > it > aulab > progetto_finale_demo_doc > controllers > ArticleController.java > ArticleController > articleSearch(String, Model)
163 public class ArticleController {
164     public String articleSetAccepted(@RequestParam("action") String action, @RequestParam("articleId") Long articleId, RedirectAttributes redirectAttributes) {
165     }
166     return "redirect:/revisor/dashboard";
167 }
168
169 //Rotta di ricerca di un articolo
170 @GetMapping("/search")
171 public String articleSearch(@Param("keyword") String keyword, Model viewModel) {
172     viewModel.addAttribute(attributeName: "title", attributeValue: "Tutti gli articoli trovati");
173
174     List<ArticleDto> articles = articleService.search(keyword);
175     List<ArticleDto> acceptedArticles = articles.stream().filter(article -> Boolean.TRUE.equals(article.getIsAccepted())).collect(Collectors.toList());
176     viewModel.addAttribute(attributeName: "articles", acceptedArticles);
177     return "article/articles";
178 }
179 }

```