

Java - User Story 2 - PDF

- Come Lorenzo
- vorrei visualizzare gli ultimi articoli sul portale
- in modo tale da informarmi su ciò che succede nel mondo

ACCEPTANCE CRITERIA:

- Manipolazione delle immagini con supabase
- Nella home, solo gli articoli più recenti (scegliete voi numero)
- In home e index, ordine dal più recente al più vecchio
- Pagina dettaglio per ogni articolo
- Click per ricerca per categoria
- Click per ricerca per scrittore

Svolgimento ↗

Procediamo quindi con la configurazione, inserimento e memorizzazione delle immagini.

SUPABASE ↗

⚠ VEDERE MATERIALE RELATIVO

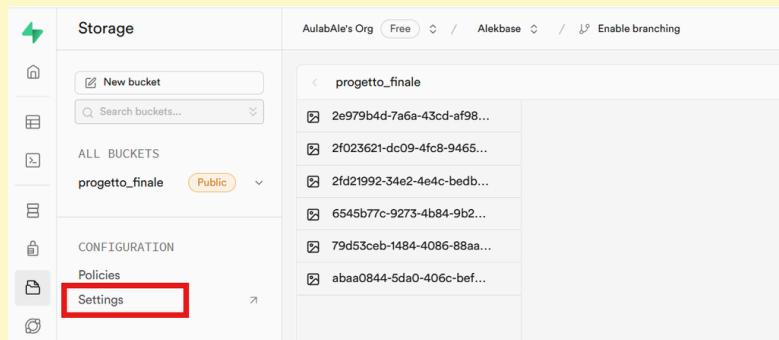
PROGETTO ↗

Creo il nostro storage in supabase colleghiamolo al nostro progetto. Iniziamo con l'inserire all'interno dell' "application.properties" le configurazioni necessarie affinchè avvenga correttamente la chiamata verso supabase, come l'url e la chiave ma anche due attributi che utilizzeremo per la manipolazione degli url delle immagini

in "src\main\resources\application.properties"

```
application.properties M X
src > main > resources > application.properties
You, 1 second ago | 2 authors (You and one other)
1 spring.application.name=progetto_finale_demo_doc
2 spring.datasource.url=jdbc:mysql://localhost:3306/progettofinaledemo
3 spring.datasource.username=root
4 spring.datasource.password=root
5 spring.datasource.driver-class-name=com.mysql.jdbc.Driver
6
7 supabase.url=https://gjeqflttqnrvrzvgxvt.supabase.co
8 supabase.key=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXQyMmFzZSIzInJlZiiI6ImdqZXFmbHR0cHFudnJpenZneHZ0IiI
9
10 supabase.bucket=/storage/v1/object/progetto_finale/
supabase.image=/storage/v1/object/public/progetto_finale/
```

⚠ ATTENZIONE: Le configurazioni dipendono dal vostro storage, trovate tutto quello che vi serve in



In "Url" c'è la stringa da inserire in "supabase.url" del vostro "application_properties" ed in "anon public" la chiave da inserire in "supabase.key"

Per quanto riguarda invece "supabase.bucket" e "supabase.image" inserite:

- supabase.bucket=/storage/v1/object/<NOME DEL VOSTRO BUCKET>/
- supabase.image=/storage/v1/object/public/<NOME DEL VOSTRO BUCKET>/

Aggiungiamo quindi queste 4 configurazione

Fatto questo andiamo a creare tutta l'infrastruttura che accoglierà l'immagine e la memorizzerà all'interno di supabase.

SCRIPT SQL

Per prima cosa dobbiamo creare la tabella che accoglierà al suo interno tutti i link alle immagini e questa verrà messa in relazione con la tabella "articles". La relazione sarà di tipo uno a molti.

Creiamo quindi all'interno di "sql" un nuovo file che chiameremo "createImageTable.sql"

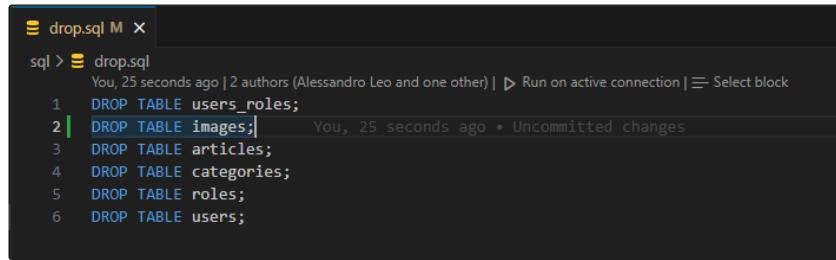
in "sql\createImageTable.sql"

```
createImageTable.sql U X
sql > createImageTable.sql
    ▶ Run on active connection | ⌂ Select block
1  create table images(
2      id BIGINT auto_increment PRIMARY KEY,
3      path VARCHAR(255) not null,
4      article_id BIGINT,
5      FOREIGN KEY (article_id) REFERENCES articles(id)
6  );
```

E procediamo quindi col lanciare lo script sulla nostra connessione.

Abbiamo anche bisogno di modificare il file "drop.sql" per la cancellazione di questa nuova tabella.

in "sql\drop.sql"



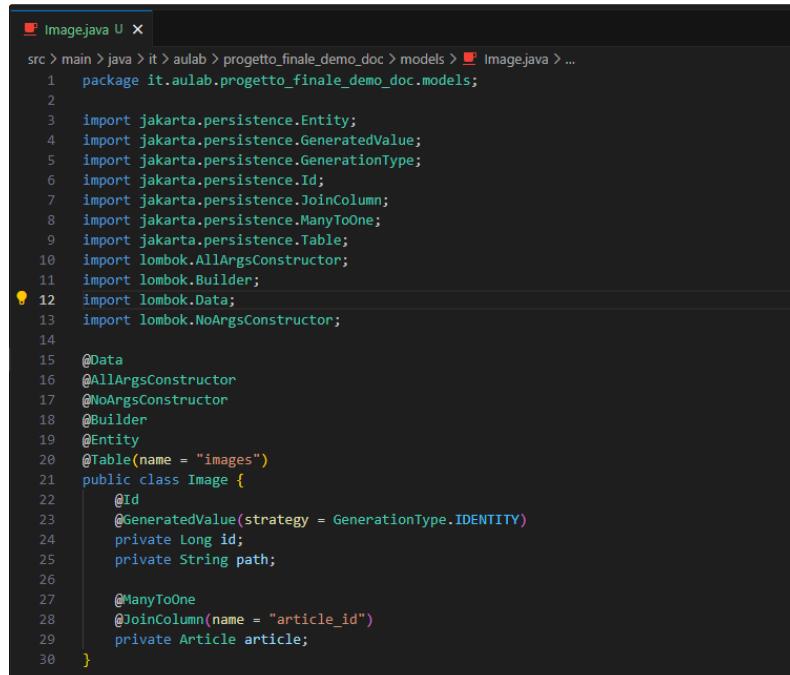
```
drop.sql M X
sql > drop.sql
You, 25 seconds ago | 2 authors (Alessandro Leo and one other) | Run on active connection | Select block
1  DROP TABLE users_roles;
2  | DROP TABLE images;| You, 25 seconds ago • Uncommitted changes
3  DROP TABLE articles;
4  DROP TABLE categories;
5  DROP TABLE roles;
6  DROP TABLE users;
```

Seguiamo esattamente questo ordine

- 💡 Ovviamente se riuscite a gestire nel giusto modo il lancio dello script, potete anche inserirlo nel file "create.sql" senza dover necessariamente creare un nuovo file "createlImageTable.sql"

CREAZIONE DEL MODEL(DAO)

Una volta creata la tabella dobbiamo procedere col creare il modello "Image" all'interno di "src\main\java\it\aulab\progetto_finale_docente\models" in "src\main\java\it\aulab\progetto_finale_docente\models\Image.java"



```
Image.java U X
src > main > java > it > aulab > progetto_finale_demo_doc > models > Image.java > ...
1 package it.aulab.progetto_finale_demo_doc.models;
2
3 import jakarta.persistence.Entity;
4 import jakarta.persistence.GeneratedValue;
5 import jakarta.persistence.GenerationType;
6 import jakarta.persistence.Id;
7 import jakarta.persistence.JoinColumn;
8 import jakarta.persistence.ManyToOne;
9 import jakarta.persistence.Table;
10 import lombok.AllArgsConstructor;
11 import lombok.Builder;
12 import lombok.Data;
13 import lombok.NoArgsConstructor;
14
15 @Data
16@AllArgsConstructor
17@NoArgsConstructor
18@Builder
19@Entity
20@Table(name = "images")
21public class Image {
22    @Id
23    @GeneratedValue(strategy = GenerationType.IDENTITY)
24    private Long id;
25    private String path;
26
27    @ManyToOne
28    @JoinColumn(name = "article_id")
29    private Article article;
30}
```

In cui abbiamo inserito anche la relazione verso il modello Article.

MODIFICA DAO E DTO

Andiamo quindi a modificare il dao dell'Article

in "src\main\java\it\aulab\progetto_finale_docente\models\Article.java"

```
Article.java M ×
src > main > java > it > aulab > progetto_finale_demo_doc > models > Article.java > ...
You, 8 seconds ago | 2 authors (Alessandro Leo and one other)
1 package it.aulab.progetto_finale_demo_doc.models;
2
3 import java.time.LocalDate;
4
5 import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
6
7 import jakarta.persistence.Column;
8 import jakarta.persistence.Entity;
9 import jakarta.persistence.GeneratedValue;
10 import jakarta.persistence.GenerationType;
11 import jakarta.persistence.Id;
12 import jakarta.persistence.JoinColumn;
13 import jakarta.persistence.ManyToOne;
14 import jakarta.persistence.OneToOne; You, 10 seconds ago * Uncommitted changes
15
16 import jakarta.validation.constraints.NotEmpty;
17 import jakarta.validation.constraints.Size;
18 import lombok.Getter;
19 import lombok.Setter;
20 import lombok.NoArgsConstructor;
21
22 You, 8 seconds ago | 2 authors (Alessandro Leo and one other)
23 @Setter
24 @Getter
25 @NoArgsConstructor
26 @Entity
27 @Table(name = "articles")
28 public class Article {
29     @Id
30     @GeneratedValue(strategy = GenerationType.IDENTITY)
31     private Long id;
32
33     @Column(nullable = false, length = 100)
34     @NotEmpty
35     @Size(max = 100)
36     private String title;
37
38     @Column(nullable = false, length = 100)
39     @NotEmpty
40     @Size(max = 100)
41     private String subtitle;
42
43     @Column(nullable = false, length = 1000)
44     @NotEmpty
45     @Size(max = 1000)
46     private String body;
47
48     @Column(nullable = true, length = 8)
49     @NotNull
50     private LocalDate publishDate;
51
52     @ManyToOne
53     @JoinColumn(name = "user_id")
54     @JsonIgnoreProperties({"articles"})
55     private User user;
56
57     @ManyToOne
58     @JsonIgnoreProperties({"articles"})
59     private Category category;
60
61     @OneToOne(mappedBy = "article")
62     @JsonIgnoreProperties({"article"})
63     private Image image;
```

ed ora anche il DTO dell'articolo

in "src\main\java\it\aulab\progetto_finale_docente\dtos\ArticleDto.java"

```

ArticleDto.java M X
src > main > java > it > aulab > progetto_finale_demo_doc > dtos > ArticleDto.java > ArticleDto > category
You, 12 seconds ago | 2 authors (Alessandro Leo and one other)
1 package itaulab.progetto_finale_demo_doc.dtos;
2
3 import java.time.LocalDate;
4
5 import itaulab.progetto_finale_demo_doc.models.Category;
6 import itaulab.progetto_finale_demo_doc.models.Image;
7 import itaulab.progetto_finale_demo_doc.models.User;
8 import lombok.Getter;
9 import lombok.Setter;
10 import lombok.NoArgsConstructor;
11
12 You, 12 seconds ago | 2 authors (Alessandro Leo and one other)
13 @Setter
14 @Getter
15 @NoArgsConstructor
16 public class ArticleDto {
17     private Long id;
18     private String title;
19     private String subtitle;
20     private String body;
21     private LocalDate publishDate;
22     private User user;
23     private Category category; Alessandro Leo, 21 hours ago * US01
24     private Image image;
}

```

ARTICLE SERVICE

Nel codice realizzato fin qui il file immagine viene catturato correttamente ma non utilizzato per la memorizzazione effettiva.

Dobbiamo quindi modificare il metodo create() all'interno dell'ArticleService

in "src\main\java\it\aulab\progetto_finale_docente\services\ArticleService.java"

```

ArticleService.java 2, M X
src > main > java > it > aulab > progetto_finale_demo_doc > services > ArticleService.java > ...
You, 20 seconds ago | 2 authors (Alessandro Leo and one other)
1 package itaulab.progetto_finale_demo_doc.services;
2
3 import java.security.Principal;
4 import org.springframework.core.convert.converter.Converter;
5 import java.util.concurrent.CompletableFuture;
6
7 import org.springframework.security.core.context.SecurityContextHolder;
8 import org.springframework.stereotype.Service;
9 import org.springframework.web.multipart.MultipartFile;
10 import org.modelmapper.ModelMapper;
11 import org.springframework.beans.factory.annotation.Autowired;
12 import org.springframework.security.core.Authentication;
13
14 Alessandro Leo, 21 hours ago * US01
15 import itaulab.progetto_finale_demo_doc.dtos.ArticleDto;
16 import itaulab.progetto_finale_demo_doc.models.Article;
17 import itaulab.progetto_finale_demo_doc.models.User;
18 import itaulab.progetto_finale_demo_doc.repositories.ArticleRepository;
19 import itaulab.progetto_finale_demo_doc.repositories.UserRepository;
20
21 You, 20 seconds ago | 2 authors (Alessandro Leo and one other)
22 @Service
23 public class ArticleService implements CrudService<ArticleDto, Article, Long>{
24
25     @Autowired
26     private UserRepository userRepository;
27
28     @Autowired
29     private ArticleRepository articleRepository;
30
31     @Autowired
32     private ModelMapper modelMapper;
33
34     @Override
35     public List<ArticleDto> readAll() {
36         // TODO Auto-generated method stub
37         throw new UnsupportedOperationException("Unimplemented method 'readAll'");
38     }
39
40     @Override
41     public ArticleDto read(Long key) {
42         // TODO Auto-generated method stub
43         throw new UnsupportedOperationException("Unimplemented method 'read'");
44     }
}

```

```

43
44     @Override
45     public ArticleDto create(Article article, Principal principal, MultipartFile file) {
46         String url = "";
47
48         Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
49         if (authentication != null) {
50             CustomUserDetails userdetails = (CustomUserDetails) authentication.getPrincipal();
51             User user = (userRepository.findById(userDetails.getId())).get();
52             article.setUser(user);
53         }
54
55         if(file.isEmpty()){
56             try {
57                 CompletableFuture<String> futureUrl = imageService.saveImageOnCloud(file);
58                 url = futureUrl.get();
59             } catch (Exception e) {
60                 e.printStackTrace();
61             }
62         }
63
64         ArticleDto dto = modelMapper.map(articleRepository.save(article), destinationType(ArticleDto.class));
65         if(file.isEmpty()){
66             imageService.setImageOnDB(url, article);
67         }
68
69         return dto;
70     }
71
72     @Override
73     public ArticleDto update(long key, Article model, MultipartFile file) {
74         // TODO Auto-generated method stub
75         throw new UnsupportedOperationException("Unimplemented method `update`");
76     }
77
78     @Override
79     public void delete(long key) {
80         // TODO Auto-generated method stub
81         throw new UnsupportedOperationException("Unimplemented method `delete`");
82     }
83
84 }
85

```

Ed importiamo "java.util.concurrent.CompletableFuture"

⚠ Dovete avere problemi con i suggerimenti di import, scrivetelo a mano

IMAGE SERVICE

Notiamo immediatamente che questo metodo ha bisogno di un service per le immagini che effettuerà le varie logiche per la memorizzazione.

Seguiremo sempre la stessa procedura di creazione dell'interfaccia per poi creare l'implementazione.

Creiamo in "src\main\java\it\aulab\progetto_finale_docente\services" la nostra interfaccia "ImageService.java".

in "src\main\java\it\aulab\progetto_finale_docente\services\ImageService.java"

```

  ImageService.java U ✘
src > main > java > it > aulab > progetto_finale_demo_doc > services > ImageService.java > ...
1 package it.aulab.progetto_finale_demo_doc.services;
2
3 import java.io.IOException;
4 import java.util.concurrent.CompletableFuture;
5
6 import org.springframework.web.multipart.MultipartFile;
7
8 import it.aulab.progetto_finale_demo_doc.models.Article;
9
10 public interface ImageService {
11     void saveImageOnDB(String url, Article article);
12     CompletableFuture<String> saveImageOnCloud(MultipartFile file) throws Exception;
13     void deleteImage(String imagePath) throws IOException;
14 }

```

Ed importiamo quello che ci serve

Creata l'interfaccia dedichiamoci all'implementazione. Creiamo all'interno di "src\main\java\it\aulab\progetto_finale_docente\services" il file "ImageServiceImpl.java"

in "src\main\java\it\aulab\progetto_finale_docente\services\ImageServiceImpl.java"

```

1  ImageServiceImpl.java 4.0
src> main> java> it> aulab> progetto_finale_demo_doc> services> ImageServiceImpl.java > supabaseKey
2
3  package it.aulab.progetto_finale_demo_doc.services;
4
5  import java.io.IOException;
6  import java.util.UUID;
7  import java.util.concurrent.CompletableFuture;
8
9  import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.beans.factory.annotation.Value;
11 import org.springframework.scheduling.annotation.Async;
12 import org.springframework.util.LinkedMultiValueMap;
13 import org.springframework.util.MultiValueMap;
14 import org.springframework.web.client.RestTemplate;
15
16 import it.aulab.progetto_finale_demo_doc.models.Article;
17 import it.aulab.progetto_finale_demo_doc.models.Image;
18
19 import org.springframework.http.*;
20
21 import jakarta.transaction.Transactional;
22
23 @Service
24 public class ImageServiceImpl implements ImageService{
25
26     @Autowired
27     private ImageRepository imageRepository;
28
29     @Value("${supabase.url}")
30     private String supabaseUrl;
31
32     @Value("${supabase.key}")
33     private String supabaseKey;
34
35     @Value("${supabase.bucket}")
36     private String supabaseBucket;
37
38     @Value("${supabase.image}")
39     private String supabaseImage;
40
41     private final RestTemplate restTemplate = new RestTemplate();
42
43     public void saveImageOnDB(String url, Article article){
44         url = url.replace(supabaseBucket, supabaseImage);
45         imageRepository.save(Image.builder().path(url).article(article).build());
46     }
47
48
49     @Sync
50     public CompletableFuture<String> saveImageOnCloud(MultipartFile file) throws Exception {
51         if(!file.isEmpty()){
52             try {
53                 String nameFile = UUID.randomUUID().toString() + "_" + file.getOriginalFilename();
54
55                 String extension = StringManipulation.getFileExtension(nameFile);
56
57                 String url = supabaseUrl + supabaseBucket + nameFile;
58
59                 MultiValueMap<String, Object> body = new LinkedMultiValueMap<>();
60
61                 body.add("file", file.getBytes());
62
63                 HttpHeaders headers = new HttpHeaders();
64                 headers.set(headerName:"Content-Type","image/"+ extension);
65                 headers.set(headerName:"Authorization", "Bearer " + supabaseKey);
66
67                 HttpEntity<byte[]> requestEntity = new HttpEntity<byte[]>(file.getBytes(), headers);
68
69                 restTemplate.exchange(url, HttpMethod.POST, requestEntity, responseType:String.class);
70
71                 return CompletableFuture.completedFuture(url);
72             } catch (Exception e) {
73                 e.printStackTrace();
74             }
75         } else {
76             throw new IllegalArgumentException("File is empty");
77         }
78
79         return CompletableFuture.failedFuture(ex: null);
80     }
81
82
83     @Sync
84     @Transactional
85     public void deleteImage(String imagePath) throws IOException {
86
87         String url = imagePath.replace(supabaseImage, supabaseBucket);
88
89         imageRepository.deleteImagePath(imagePath);
90
91         RestTemplate restTemplate = new RestTemplate();
92
93         HttpHeaders headers = new HttpHeaders();
94         headers.set(headerName:"Authorization", "Bearer " + supabaseKey);
95
96         HttpEntity<String> entity = new HttpEntity<String>(headers);
97
98         ResponseEntity<String> response = restTemplate.exchange(url, HttpMethod.DELETE, entity, responseType:String.class);
99
100    }
101}

```

Ed importiamo quello che ci serve.

In questa classe abbiamo creato delle proprietà che “agganciano” e assumono il valore delle chiavi nell’application properties.

Le funzioni di salvataggio nel cloud e cancellazione non fanno altro che gestire le chiavi http verso supabase.

Molto interessante è il metodo “restTemplate.exchange()” che nella chiamata rest effettua uno scambio dell'url dato in pasto alla chiamata con quello effettivo dell’immagine nel cloud.

I due metodi poi sono corredati da annotation “@Async” che come dice la traduzione stessa li rende asincroni, questo vuol dire che non andranno ad appesantire il flusso del nostro progetto, poiché la memorizzazione e la cancellazione delle immagini dipendendo da un altro server. Se fossero sincroni potrebbero dare problemi di rallentamento.

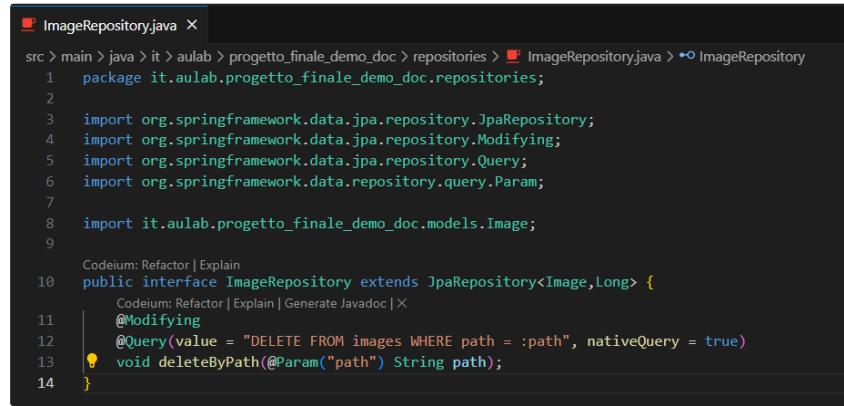
Notiamo anche la necessità di un’injection di “**imageRepository**” che gestisce le immagini e l’utilizzo di un metodo chiamato “getFileExtension()” della classe “**StringManipulation**”.

Entrambe non esistono nel nostro progetto, quindi procediamo con la creazione.

IMAGE REPOSITORY

All’interno di “src\main\java\it\aulab\progetto_finale_docente\repositories” creiamo il nostro “ImageRepository.java”

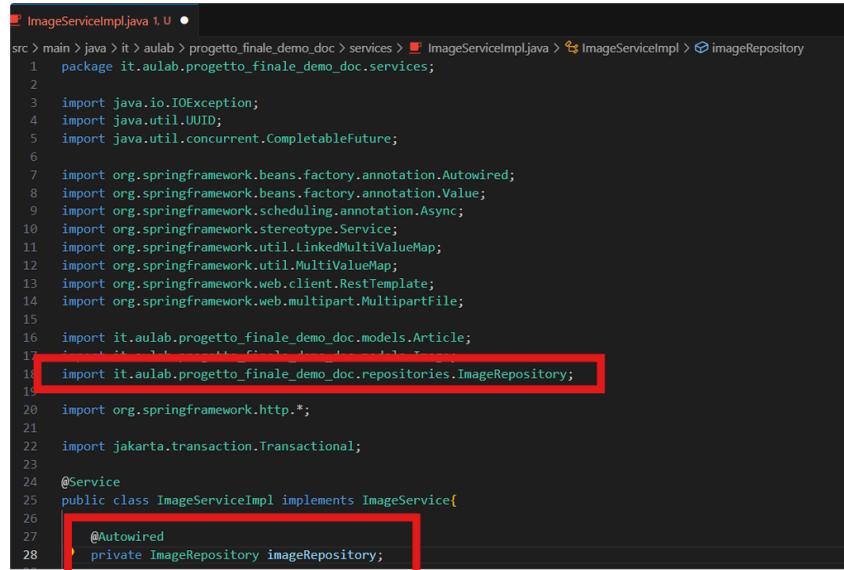
in "src\main\java\it\aulab\progetto_finale_docente\repositories\ImageRepository.java"



```
ImageRepository.java X
src > main > java > it > aulab > progetto_finale_demo_doc > repositories > ImageRepository.java > ImageRepository
1 package it.aulab.progetto_finale_demo_doc.repositories;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.data.jpa.repository.Modifying;
5 import org.springframework.data.jpa.repository.Query;
6 import org.springframework.data.repository.query.Param;
7
8 import it.aulab.progetto_finale_demo_doc.models.Image;
9
10
11
12
13 void deleteByPath(@Param("path") String path);
14 }
```

Dove il metodo "deleteByPath" è correddato con la annotation "@Transactional" poichè deve seguire delle regole "transazionali" in fase di cancellazione dell'immagine. Transazionale poichè fa dei check di avvenuta operazione, in caso contrario torna tutto come se l'operazione non fosse mai avvenuta.

Creata il repository possiamo importarlo all'interno della implementazione del service



```
ImageServiceImpl.java 1.0 •
src > main > java > it > aulab > progetto_finale_demo_doc > services > ImageServiceImpl.java > ImageServiceImpl > imageRepository
1 package it.aulab.progetto_finale_demo_doc.services;
2
3 import java.io.IOException;
4 import java.util.UUID;
5 import java.util.concurrent.CompletableFuture;
6
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.beans.factory.annotation.Value;
9 import org.springframework.scheduling.annotation.Async;
10 import org.springframework.stereotype.Service;
11 import org.springframework.util.LinkedMultiValueMap;
12 import org.springframework.util.MultiValueMap;
13 import org.springframework.web.client.RestTemplate;
14 import org.springframework.web.multipart.MultipartFile;
15
16 import it.aulab.progetto_finale_demo_doc.models.Article;
17 import it.aulab.progetto_finale_demo_doc.repositories.ImageRepository;
18
19 import org.springframework.http.*;
20
21 import jakarta.transaction.Transactional;
22
23
24 @Service
25 public class ImageServiceImpl implements ImageService{
26
27     @Autowired
28     private ImageRepository imageRepository;
```

UTILS STRING MANIPULATION

Per poter memorizzare correttamente un file immagine all'interno di supabase abbiamo bisogno di estrarre l'estensione del file da quello originale inserito nel form e mandarlo nel modo corretto allo storage. Purtroppo non è un'operazione che avviene in automatico quindi abbiamo bisogno di una logica che estrae l'estensione per poi inviarla a supabase come già scritto all'interno del service.

Inserire la logica di estrazione all'interno del nostro service "nascondendola" al resto della nostra implementazione non è corretto. Potrebbe esserci utile in altri momenti. Quindi creiamo una classe di utilità che contiene dei metodi di utilità. Uno fra questi un metodo che riceve in input la stringa del nome del file e ne estrapola l'estensione

Creiamo quindi un nuovo folder chiamato "utils" all'interno di "src\main\java\it\aulab\progetto_finale_docente" ed al suo interno creiamo la classe "StringManipulation.java"

in "src\main\java\it\aulab\progetto_finale_docente\utils\StringManipulation.java"

```

1 package it.aulab.progetto_finale_demo_doc.utils;
2
3 public class StringManipulation {
4
5     public static String getFileExtension(String nameFile){
6         int dotIndex = nameFile.indexOf('.');
7         String extension = nameFile.substring(dotIndex + 1);
8         return extension;
9     }
10}

```

Fatto questo nel nostro ImageServiceImpl possiamo importare la nuova classe

```

1 package it.aulab.progetto_finale_demo_doc.services;
2
3 public class ImageServiceImpl implements ImageService{
4
5     private String supabaseBucket;
6
7     @Value("${supabase.image}")
8     private String supabaseImage;
9
10    private final RestTemplate restTemplate = new RestTemplate();
11
12    public void saveImageOnDB(String url, Article article){
13        url = url.replace(supabaseBucket, supabaseImage);
14        imageRepository.save(Image.builder().path(url).article(article).build());
15    }
16
17    @Async
18    public CompletableFuture<String> saveImageOnCloud(MultipartFile file) throws Exception {
19        if(!file.isEmpty()){
20            try {
21                String nameFile = UUID.randomUUID().toString() + "_" + file.getOriginalFilename();
22
23                String extension = StringManipulation.getFileExtension(nameFile);
24                String url = supabaseUrl + supabaseBucket + nameFile;
25
26                ...
27            } catch (Exception e) {
28                ...
29            }
30        }
31    }
32}

```

e non avremo più errori in questa classe.

Una volta terminati questi sviluppi possiamo procedere con l'injection del service delle immagini all'interno di "src\main\java\it\aulab\progetto_finale_docente\services\ArticleService.java"

in "src\main\java\it\aulab\progetto_finale_docente\services\ArticleService.java"

```

1 package it.aulab.progetto_finale_demo_doc.services;
2
3 import java.security.Principal;
4 import java.util.List;
5 import java.util.concurrent.CompletableFuture;
6
7 import org.springframework.security.core.context.SecurityContextHolder;
8 import org.springframework.stereotype.Service;
9 import org.springframework.web.multipart.MultipartFile;
10 import org.modelmapper.ModelMapper;
11 import org.springframework.beans.factory.annotation.Autowired;
12 import org.springframework.security.core.Authentication;
13
14 import it.aulab.progetto_finale_demo_doc.dtos.ArticleDto;
15 import it.aulab.progetto_finale_demo_doc.models.Article;
16 import it.aulab.progetto_finale_demo_doc.models.User;
17 import it.aulab.progetto_finale_demo_doc.repositories.ArticleRepository;
18 import it.aulab.progetto_finale_demo_doc.repositories.UserRepository;
19
20 ...
21
22 ...
23 ...
24 ...
25 ...
26 ...
27 ...
28 ...
29 ...
30 ...

```

Ultimo step, abbiamo bisogno di configurare la possibilità di lavorare in asincrono e di attivare le transazioni. Lo rendiamo disponibile aggiungendo le annotation:

```

1 @EnableAsync(proxyTargetClass = true)
2 @EnableTransactionManagement

```

all'interno del file principale "ProgettoFinaleDocenteApplication"

in "src\main\java\it\aulab\progetto_finale_docente\ProgettoFinaleDocenteApplication.java"

```

1 You 1 second ago | 2 authors (Alessandro Leo and one other)
2 package it.aulab.progetto_finale_demo_doc;
3
4 import org.modelmapper.ModelMapper;
5 import org.springframework.boot.SpringApplication;
6 import org.springframework.context.annotation.Bean;
7 import org.springframework.scheduling.annotation.EnableAsync;
8 import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
9 import org.springframework.security.crypto.password.PasswordEncoder;
10 import org.springframework.transaction.annotation.EnableTransactionManagement;
11
12 You 1 second ago | 2 authors (Alessandro Leo and one other)
13 @EnableAsync(proxyTargetClass = true)
14 @EnableTransactionManagement
15
16 public class ProgettoFinaleDemoDocApplication {
17
18     Run | Debug
19     public static void main(String[] args) {
20         SpringApplication.run(ProgettoFinaleDemoDocApplication.class, args);
21     }
22
23     @Bean
24     public PasswordEncoder passwordEncoder() {
25         return new BCryptPasswordEncoder();
26     }
27
28     @Bean
29     public ModelMapper instanceModelMapper(){
30         ModelMapper mapper = new ModelMapper();
31         return mapper;
32     }
33

```

Siamo pronti per effettuare un test. Memorizziamo un nuovo articolo selezionando una immagine. Vedremo che l'articolo viene memorizzato correttamente, avremo un nuovo record nella tabella images e all'interno dello storage troveremo la nostra immagine appena inviata.

INDEX ARTICOLI

Nella nostra piattaforma però non abbiamo ancora un modo effettivo per visualizzare tutti gli articoli inseriti nel nostro progetto.

Procediamo quindi con la creazione della pagina di index.

Per prima cosa dovremo creare l'handler all'interno del nostro controller degli articoli

in "src\main\java\it\aulab\progetto_finale_docente\controllers\ArticleController.java"

```

1 You 2 days ago | 2 authors (Alessandro Leo and one other)
2 package it.aulab.progetto_finale_demo_doc.controllers; Alessandro Leo, 2 days ago * US01
3
4 import java.security.Principal;
5 import java.util.Collections;
6 import java.util.Comparator;
7 import java.util.List;
8
9 import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.beans.factory.annotation.Qualifier;
11 import org.springframework.stereotype.Controller;
12 import org.springframework.ui.Model;
13 import org.springframework.validation.BindingResult;
14 import org.springframework.web.bind.annotation.GetMapping;
15 import org.springframework.web.bind.annotation.ModelAttribute;
16 import org.springframework.web.bind.annotation.PostMapping;
17 import org.springframework.web.bind.annotation.RequestMapping;
18 import org.springframework.web.multipart.MultipartFile;
19 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
20
21 import it.aulab.progetto_finale_demo_doc.dtos.ArticleDto;
22 import it.aulab.progetto_finale_demo_doc.dtos.CategoryDto;
23 import it.aulab.progetto_finale_demo_doc.models.Article;
24 import it.aulab.progetto_finale_demo_doc.models.Category;
25 import it.aulab.progetto_finale_demo_doc.services.ArticleService;
26 import jakarta.validation.Valid;
27
28 You 2 days ago | 2 authors (Alessandro Leo and one other)
29 @Controller
30 @RequestMapping("/articles")
31 public class ArticleController {
32
33     @Autowired
34     @Qualifier("categoryService")
35     private CrudService<CategoryDto, Category, Long> categoryService;
36
37
38     @Autowired
39     private ArticleService articleService;
40
41     //Rotta index degli articoli
42     @GetMapping
43     public String articlesIndex(Model viewModel) {
44         viewModel.addAttribute("title", "Tutti gli articoli");
45
46         List<ArticleDto> articles = articleService.readAll();
47
48         Collections.sort(articles, Comparator.comparing(ArticleDto::getPublishDate).reversed());
49         viewModel.addAttribute("articles", articles);
50
51         return "article/articles";
52     }
53
54     //Rotta per la creazione di un articolo
55     @GetMapping("create")
56     public String articleCreate(Model viewModel) {
57         viewModel.addAttribute("title", "Crea un articolo");
58         viewModel.addAttribute("article", new Article());
59         viewModel.addAttribute("categories", categoryService.readAll());
60
61     }

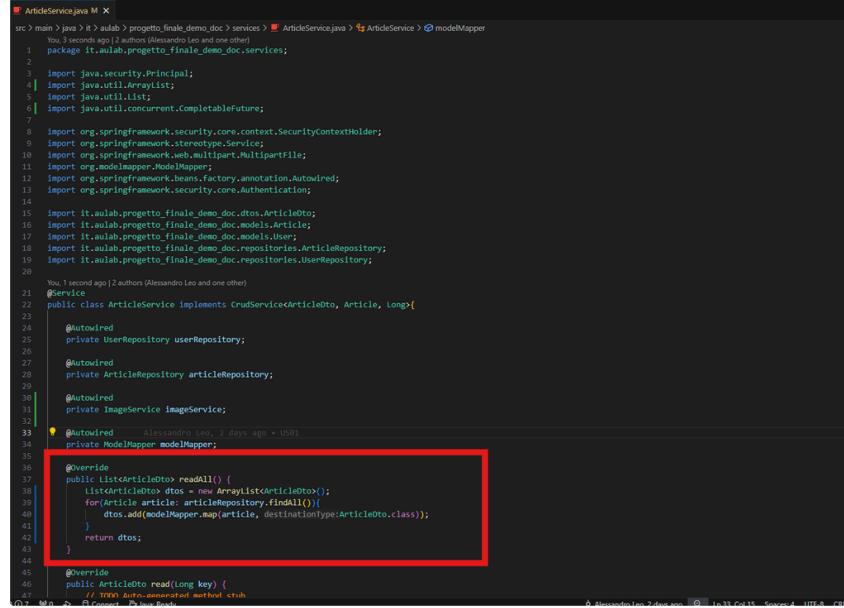
```

Inseriamo il metodo al di sopra del metodo "articlecreate" ed importiamo "java.util.Collections", "java.util.Comparator", "java.util.List"

In questo metodo abbiamo anche inserito una logica sulla collezione che tramite "ArticleDto::getPublishDate" e "comparator" ordina i risultati in base all'articolo più recente che di conseguenza viene visualizzato per primo.

Questo metodo però richiede un metodo specifico chiamato "readAll()" all'interno dell'ArticleService "articleService.readAll()" non ancora implementato, esiste ma possiede la giusta logica. Procediamo allora con l'implementazione

in "src\main\java\it\aulab\progetto_finale_docente\services\ArticleService.java"



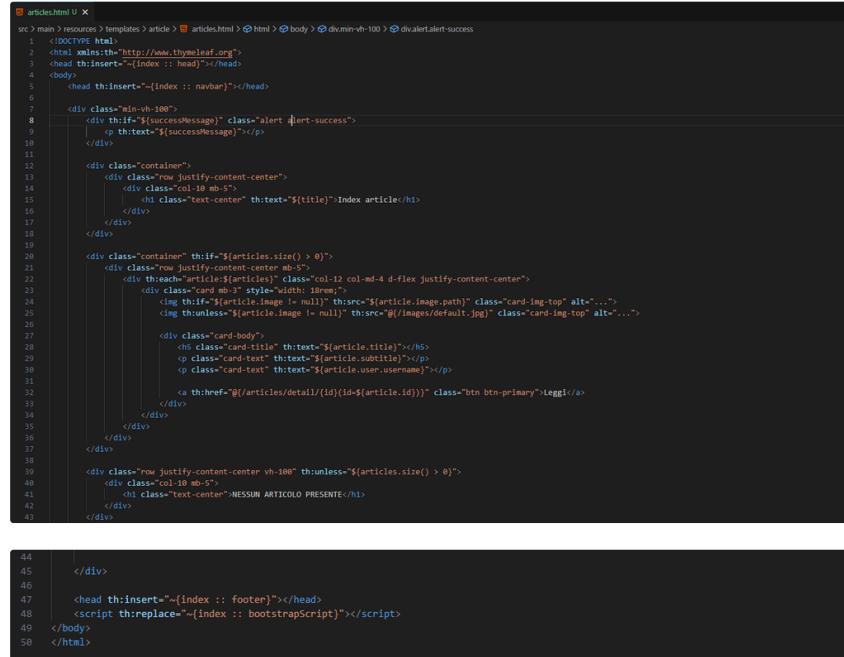
```

1 ArticleService.java M
src > main > java > it > aulab > progetto_finale_demo_doc > services > ArticleService.java > ArticleService.java > modelMapper
You, 3 seconds ago | 2 authors (Alessandro Leo and one other)
1 package it.aulab.progetto_finale_demo_doc.services;
2
3 import java.security.Principal;
4 import java.util.ArrayList;
5 import java.util.List;
6 import java.util.concurrent.CompletableFuture;
7
8 import org.springframework.security.core.context.SecurityContextHolder;
9 import org.springframework.web.multipart.MultipartFile;
10 import org.modelmapper.ModelMapper;
11 import org.springframework.beans.factory.annotation.Autowired;
12 import org.springframework.security.core.Authentication;
13
14 import it.aulab.progetto_finale_demo_doc_dto.ArticleDto;
15 import it.aulab.progetto_finale_demo_doc.models.Article;
16 import it.aulab.progetto_finale_demo_doc.models.User;
17 import it.aulab.progetto_finale_demo_doc.repositories.ArticleRepository;
18 import it.aulab.progetto_finale_demo_doc.repositories.UserRepository;
19
20 You, 3 seconds ago | 2 authors (Alessandro Leo and one other)
21 @Service
22 public class ArticleService implements CrudService<ArticleDto, Article, Long>{
23
24     @Autowired
25     private UserRepository userRepository;
26
27     @Autowired
28     private ArticleRepository articleRepository;
29
30     @Autowired
31     private ImageService imageService;
32
33     @Autowired
34     Alessandro Leo, 2 days ago | User
35     private ModelMapper modelMapper;
36
37     @Override
38     public List<ArticleDto> readAll() {
39         List<ArticleDto> dtos = new ArrayList<ArticleDto>();
40         for(Article article: articleRepository.findAll()){
41             dtos.add(modelMapper.map(article, destinationType:ArticleDto.class));
42         }
43         return dtos;
44     }
45
46     @Override
47     public ArticleDto read(Long key) {
48         // TODO Auto-generated method stub
49     }
50 }

```

Fatto questo creiamo il template che questo handler restituirà. All'interno di "src\main\resources\templates\article" creiamo il nostro html "articles.html"

in "src\main\resources\templates\article\articles.html"

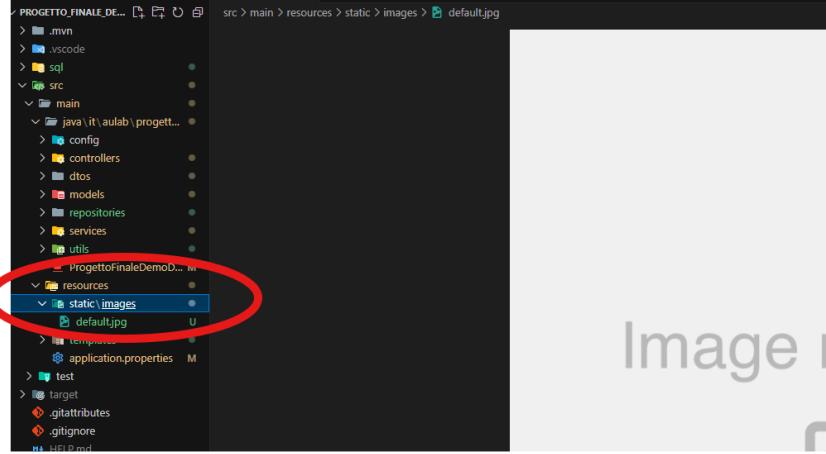


```

1 articles.html U
src > main > resources > templates > article > articles.html > HTML > body > div.min-vh-100 > div.alert.alert-success
2 <!DOCTYPE html>
3 <html xmlns="http://www.thymeleaf.org">
4     <head th:insert="~{index :: head}"></head>
5     <body>
6         <head th:insert="~{index :: navbar}"></head>
7
8         <div class="min-vh-100">
9             <div th:if="${successMessage}" class="alert alert-success">
10                 <p th:text="${successMessage}"></p>
11             </div>
12
13             <div class="container">
14                 <div class="row justify-content-center">
15                     <div class="col-10 mb-5" th:unless="${articles.size() > 0}">
16                         <div class="card mb-3" style="width: 18rem;">
17                             <div th:each="article:$articles" class="col-12 col-md-4 d-flex justify-content-center">
18                                 <div class="card mb-3" style="width: 18rem;">
19                                     
20                                     
21
22                                     <div class="card-body">
23                                         <h5 class="card-title" th:text="${article.title}"></h5>
24                                         <p class="card-text" th:text="${article.subtitle}"></p>
25                                         <p class="card-text" th:text="${article.user.username}"></p>
26
27                                         <a href="#" th:href="@{/articles/detail/10/(id=${article.id})}" class="btn btn-primary">Leggi</a>
28                                     </div>
29                                 </div>
30                             </div>
31                         </div>
32                     </div>
33                 </div>
34             </div>
35
36             <div class="row justify-content-center vh-100" th:unless="${articles.size() > 0}">
37                 <div class="col-10 mb-5" th:unless="${articles.size() > 0}">
38                     <h1 class="text-center">NESSUN ARTICOLO PRESENTE</h1>
39                 </div>
40             </div>
41         </div>
42     </div>
43
44     </div>
45
46     <head th:insert="~{index :: footer}"></head>
47     <script th:replace="~{index :: bootstrapScript}"></script>
48
49 </body>
50 </html>

```

Dove con "th:unless="\${article.image != null}"" abbiamo inserito la possibilità di visualizzare un'immagine di default in caso l'articolo venisse creato senza immagine. Però questa immagine non l'abbiamo effettivamente inserita. Prendiamone una a scelta ed [inseriamola all'interno del percorso "src\main\resources\static\images\default.jpg"](#) dove **"images"** è un nuovo folder che creiamo all'interno di "src\main\resources\static" e "default.jpg" è il nome che daremo alla nostra immagine scelta. Tutto questo nel nostro codice viene indicato con "/images/default.jpg" all'interno del "src" del tag.



Ultimo step inseriamo il collegamento sulla navbar che ci riporterà al nostro handler per poi mostrarci tutti gli articoli

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

```

index.html N
src > main > resources > templates > index.html > body > nav.navbar.navbar-expand-lg.navbar-light.bg-light > div.container-fluid > a.navbar-brand
You, 2 seconds ago [2 authors (Alessandro Lioi and one other)
1 <!doctype html>
2 <html xmlns="http://www.thymeleaf.org">
3
4 <!-- fragment head -->
5 <head th:fragment="head">
6   <meta charset="UTF-8">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QfIk2yjP6J1S5hM0b2xgPhs4uIX+LJA9JZPKZSQIAj61a86w+fbzTkmFyXqBv0cvQoQ1kQeSCm3W6n--" crossorigin="anonymous">
9
10  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.2/css/all.min.css" integrity="sha512-SmHSt+b2xp0eRpo6Y7cOQHs4uIX+LJA9JZPKZSQIAj61a86w+fbzTkmFyXqBv0cvQoQ1kQeSCm3W6n--" crossorigin="anonymous" referrerPolicy="no-referrer" />
11
12  <title th:text="${title}">Title</title>
13 </head>
14 <!-- fine fragment head -->
15 <body>
16   <div class="container-fluid">
17     <nav class="navbar navbar-expand-lg navbar-light bg-light">
18       <div class="container-fluid">
19         <a class="navbar-brand" href="#">Aulab Chronicle</a> You, 1 second ago + Uncommitted changes
20         <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
21           <span class="navbar-toggler-icon"></span>
22         </button>
23         <div id="navbarSupportedContent" class="collapse navbar-collapse" style="margin-left: auto; margin-right: 0;">
24           <ul class="navbar-nav me-auto mb-2 mb-lg-0">
25             <li sec:authorize="isAuthenticated" class="nav-item">
26               <a class="nav-link" href="#">Crea articolo</a>
27             <li class="nav-item">
28               <a class="nav-link" href="#">Tutti gli articoli</a>
29             </li>
30           </ul>
31           <ul class="nav-item dropdown">
32             <a class="nav-link" href="#" id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false" style="color: #007bff; font-weight: bold; font-size: 1.2em; padding-bottom: 0.5em; border-bottom: 1px solid black; border-radius: 0.25em; background-color: transparent; text-decoration: none; text-decoration: underline; text-decoration-color: transparent; text-decoration-style: none; text-decoration-width: 0; text-align: center; width: fit-content; margin: 0 auto; margin-top: 10px; margin-bottom: 10px; font-family: inherit; font-size: inherit; line-height: inherit; border: none; outline: none; transition: all 0.3s ease-in-out; position: relative; z-index: 1000; ">
33               Accesso
34             </a>
35             <ul class="dropdown-menu" aria-labelledby="navbarDropdown" style="background-color: #f9f9f9; border: 1px solid #ccc; border-radius: 0.25em; padding: 10px; margin: 0; position: absolute; top: 100%; left: 0; width: fit-content; font-size: 0.9em; font-weight: normal; font-family: inherit; font-size: inherit; line-height: inherit; border: none; outline: none; transition: all 0.3s ease-in-out; z-index: 1001; ">
36               <li sec:authorize="isAnonymous"><a class="dropdown-item" aria-current="page" th:href="@{/register}"/>Register</a></li>
37               <li sec:authorize="isAnonymous"><a class="dropdown-item" aria-current="page" th:href="@{/login}"/>Login</a></li>
38               <li sec:authorize="isAuthenticated"><a class="dropdown-item" aria-current="page" th:href="@{/logout}"/>Logout</a></li>
39             </ul>
40           </ul>
41         </div>
42       </div>
43     </nav>
44   </div>
45 </body>
46 </html>

```

Facciamo ora un test inserendo articoli con immagini ma anche senza (se non creati già in precedenza) e vedremo che dove non abbiamo inserito l'immagine ci verrà mostrata quella inserita come default.

Cosa molto interessante da notare è che se non abbiamo fatto la login o non ci siamo registrati non possiamo avere accesso alla pagina index con tutti gli articoli e questa cosa non è esattamente giusta.

Sblocchiamo quindi questa funzionalità.

La soluzione è molto semplice, dobbiamo modificare le nostre configurazioni di sicurezza

in "src\main\java\it\aulab\progetto_finale_docente\config\SecurityConfig.java"

```

  SecurityConfig.java M X
src > main > java > it > aulab > progetto_finale_demo_doc > config > SecurityConfig.java > SecurityConfig > filterChain(HttpSecurity)
11 import org.springframework.config.http.SessionCreationPolicy;
12 import org.springframework.security.crypto.password.PasswordEncoder;
13 import org.springframework.security.web.SecurityFilterChain;
14 import org.springframework.security.web.util.matcher.AntPathRequestMatcher;
15
16 import it.aulab.progetto_finale_demo_doc.services.CustomUserDetailsService;
17
18 You, 3 minutes ago | 2 authors (Alessandro Leo and one other)
19 @Configuration
20 @EnableWebSecurity
21 public class SecurityConfig{
22
23     @Autowired
24     private CustomUserDetailsService customUserDetailsService;
25
26     @Autowired
27     private PasswordEncoder passwordEncoder;
28
29     @Bean
30     public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
31         http
32             .csrf(csrf -> csrf.disable())
33             .authorizeHttpRequests(authorize) -> Alessandro Leo, 3 days ago + USE1
34                 authorize.requestMatchers(...,patterns:"/register**").permitAll()
35                 .requestMatchers(...,patterns:"/register", "/", "/articles").permitAll()
36                 .anyRequest().authenticated()
37             .formLogin(form ->
38                 form.loginPage("/login")
39                 .loginProcessingUrl("/loginProcessingUrl:/login")
40                 .defaultSuccessUrl(defaultSuccessUrl:/)
41                 .permitAll()
42             ).logout(logout -> logout
43                 .logoutRequestMatcher(new AntPathRequestMatcher(pattern:"/logout"))
44                 .permitAll()
45             ).exceptionHandling(exception -> exception.accessDeniedPage(accessDeniedUrl:/error/403))
46             .sessionManagement(session -> session
47                 .sessionCreationPolicy(SessionCreationPolicy.IF_REQUIRED)
48                 .maximumSessions(maximumSessions:1)
49                 .expiredUrl(expiredUrl:/login?session-expired=true)
50             );
51         }
52     }

```

Dove oltre all'uri “/register” abbiamo aggiunto anche “ “/”, “/articles” “ ai permessi delle request, in soldoni permetti l'accesso da non loggato non solo alla pagina di index ma anche quella di home che fino ad ora risultava anche essa bloccata.

Ma se facessimo un test nella pagina di index , [se presente un articolo con immagine di default](#) vedremmo che questa non ci verrà mostrata, questo perchè nelle configurazioni va aggiunto un ulteriore permesso

```
.requestMatchers(...,patterns:"/register", "/", "/articles", "/images/**").permitAll() You
```

abbiamo aggiunto un nuovo uri “ /images/** ” per consentire i permessi di visualizzazione anche a qualsiasi risorsa statica presente all'interno del folder “images”.

DETTAGLIO ARTICOLO

Abilitiamo ora il tasto “leggi” nelle card, banalmente dobbiamo implementare il dettaglio degli articoli.

Iniziamo col creare il nuovo handler sempre all'interno del controllore “ArticleController”

in “src\main\java\it\aulab\progetto_finale_docente\controllers\ArticleController.java”

```

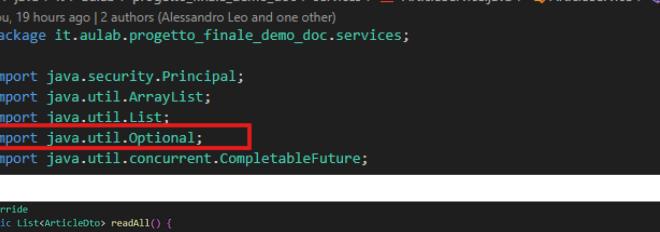
  ArticleController.java M X
src > main > java > it > aulab > progetto_finale_demo_doc > controllers > ArticleController.java > ArticleController > detailArticle(Long, Model)
You, 27 seconds ago | 2 authors (Alessandro Leo and one other)
1 package it.aulab.progetto_finale_demo_doc.controllers;
2
3 import java.security.Principal;
4 import java.util.Collections;
5 import java.util.Comparator;
6 import java.util.List;
7
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.beans.factory.annotation.Qualifier;
10 import org.springframework.stereotype.Controller;
11 import org.springframework.ui.Model;
12 import org.springframework.validation.BindingResult;
13 import org.springframework.web.bind.annotation.GetMapping;
14 import org.springframework.web.bind.annotation.ModelAttribute;
15 import org.springframework.web.bind.annotation.PathVariable;
16 import org.springframework.web.bind.annotation.PostMapping;
17 import org.springframework.web.bind.annotation.RequestMapping;
18 import org.springframework.web.multipart.MultipartFile;
19 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
20
21 import it.aulab.progetto_finale_demo_doc.dtos.ArticleDto;
22 import it.aulab.progetto_finale_demo_doc.dtos.CategoryDto;
23 import it.aulab.progetto_finale_demo_doc.models.Article;
24 import it.aulab.progetto_finale_demo_doc.models.Category;
25 import it.aulab.progetto_finale_demo_doc.services.ArticleService;
26 import it.aulab.progetto_finale_demo_doc.services.CrudService;
27 import jakarta.validation.Valid;
```

```
63 //Rotta per lo store di un articolo
64
65 @PostMapping
66 public String articleStore(@Valid @ModelAttribute("article") Article article,
67     BindingResult result,
68     RedirectAttributes redirectAttributes,
69     Principal principal,
70     MultipartFile file,
71     Model viewModel) {
72
72 //Controllo degli errori con validazioni
73 if (result.hasErrors()) {
74     viewModel.addAttribute("title", "Crea un articolo");
75     viewModel.addAttribute("article", article);
76     viewModel.addAttribute("categories", categoryService.readAll());
77     return "article/create";
78 }
79
80 articleService.create(article, principal, file);
81 redirectAttributes.addFlashAttribute(attributeName:"successMessage", attributeValue:"Articolo aggiunto con successo!");
82
83 return "redirect:/";
84 }
85
86 //Rotta di dettaglio di un articolo      You, 2 seconds ago * Uncommitted changes
87 @GetMapping("detail/{id}")
88 public String detailArticle(@PathVariable("id") Long id, Model viewModel) {
89     viewModel.addAttribute("title", "Articolo detail");
90     viewModel.addAttribute("article", articleService.read(id));
91     return "article/detail";
92 }
```

Ed importiamo "org.springframework.web.bind.annotation.PathVariable"

L'handler appena creato utilizza il metodo `read()` dell'article service che attualmente non possiede la giusta logica, quindi procediamo con l'implementazione corretta.

in "src\main\java\it\aulab\progetto_finale_docente\services\ArticleService.java"



```
src > main > java > it > aulab > progetto_finale_demo_doc > services > ArticleService.java > ArticleService > read(Long)

    You, 19 hours ago | 2 authors (Alessandro Leo and one other)
1 package it.ulab.progetto_finale_demo_doc.services;
2
3 import java.security.Principal;
4 import java.util.ArrayList;
5 import java.util.List;
6 import java.util.Optional;
7 import java.util.concurrent.CompletableFuture;

39
40 @Override
41 public List<ArticleDto> readAll() {
42     List<ArticleDto> dtos = new ArrayList<ArticleDto>();
43     for(Article article: articleRepository.findAll()){
44         dtos.add(modelMapper.map(article, destinationType:ArticleDto.class));
45     }
46     return dtos;
47 }
48
49 @Override Alessandro Leo, 3 days ago + US01
50 public ArticleDto read(Long key) {
51     Optional<Article> optArticle = articleRepository.findById(key);
52     if (optArticle.isPresent()) {
53         return modelMapper.map(optArticle.get(), destinationType:ArticleDto.class);
54     } else {
55         throw new ResponseStatusException(HttpStatus.NOT_FOUND, "Author id=" + key + " not found");
56     }
57 }
```

ed importiamo “java.util.Optional”.

Fatto questo procediamo col creare il template "detail.html" all'interno di "src\main\resources\templates\article"

in "src\main\resources\templates\article\detail.html"

```
src/main/resources/templates/article.html > article > static.html > body > div.container.my-5 > div.row.justify-content-center > div.col-12.col-md-8.d-flex-column > div.text-center > div.text-muted.my-3 > p > pre> detail.html M X
```

You, a guest (Sign in) | authors (Alessandro Leo and one other)

1 <html>

2 <head>

3 <meta charset="UTF-8" />

4 <meta name="viewport" content="width=device-width, initial-scale=1.0" />

5 <title>\${article.title}</title>

6 <link href="https://www.thymeleaf.org/doc/tutorials/3.0/using-thymeleaf.html" rel="stylesheet" type="text/css" />

7 <script src="https://code.jquery.com/jquery-3.5.1.min.js" type="text/javascript" />

8 <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js" type="text/javascript" />

9 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.js" type="text/javascript" />

10 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" type="text/css" />

11 | <div class="container-fluid p-5 bg-secondary-subtle text-center">

12 <div class="row justify-content-center">

13 <div class="col-12">

14 \${article.title}

15 <h2 th:text="\${article.title}">Titolo:</h2>

16 </div>

17 </div>

18 <div class="container my-5">

19 <div class="row justify-content-center">

20 <div class="col-12 d-flex flex-column">

21 <div id="carouselExample" class="carousel slide">

22 <div class="carousel-inner">

23 <div class="carousel-item active">

24

25

26 </div>

27 </div>

28 <button class="carousel-control-prev" type="button" data-bs-target="#carouselExample" data-bs-slide="prev">

29

30 Previous

31 </button>

32 <button class="carousel-control-next" type="button" data-bs-target="#carouselExample" data-bs-slide="next">

33

34 Next

35 </button>

36 </div>

37 <div class="text-center">

38 <h2 th:text="\${article.subtitle}">Sottotitolo:</h2>

39

40 <p class="fs-5" th:if="\${article.category} != null">Categorie:</p>

41 <ul style="list-style-type: none; padding-left: 0;">

42 \${article.category.name}

43

44 <p class="fs-5" th:if="\${article.category == null}">Nessuna categoria:</p>

```

43     <div class="text-muted my-3">
44         <p>Mediato 11 · \${article.publishDate}</p>
45         <a class="text-muted" href="#"> You, 1 second ago · Uncommitted changes</a>
46     </div>
47     </div>
48     </div>
49     <div>
50         <p>${article.body}</p>
51         <div class="text-center">
52             <a href="#" class="text-secondary pointer" onclick="goBack()>Torna indietro</a>
53         </div>
54     </div>
55 </div>
56 <head th:insert="~{index :: footer}"></head>
57 <script th:replace="~{index :: bootstrapScript}"></script>
58 <script>
59     function goBack() {
60         window.history.back();
61     }
62 </script>
63 </body>
64 </html>

```

Notiamo una cosa interessante utilizzata. Un piccolo script javascript che ci permette di tornare alla pagina precedente.

```

<script>
    function goBack() {
        window.history.back();
    }
</script>

```

Prima di procedere con un test dobbiamo rendere anche questa rotta accessibile ad utenti non loggati, aggiungiamo l'uri nel nostro security config.
in "src\main\java\it\aulab\progetto_finale_docente\config\SecurityConfig.java"
aggiungiamo solo "/articles/detail/**" al requestMatchers

```

.requestMatchers(...patterns:"/register", "/", "/articles", "/images/**", "/articles/detail/**").permitAll()

```

Facciamo ora un test e vediamo che tutto funziona correttamente

CLICK PER RICERCA PER CATEGORIA

Ma nella nostra card di dettaglio abbiamo predisposto due link, uno che ci consentirà di visualizzare tutti gli articoli di una specifica categoria, l'altro tutti gli articoli inseriti da uno specifico autore.

Procediamo quindi con l'abilitare il link della ricerca per categoria.

Per poter far funzionare correttamente il tutto, avremo bisogno di un controller per le categorie ed al suo interno creare l'handler che attiverà la logia.

Creiamo quindi il controller delle categorie "CategoryController.java" in "src\main\java\it\aulab\progetto_finale_docente\controllers" ed al suo interno inseriamo l'handler per la ricerca

in "src\main\java\it\aulab\progetto_finale_docente\controllers\CategoryController.java"

```

1  package it.aulab.progetto_finale_demo_doc.controllers;
2
3  import java.util.List;
4
5  import org.modelmapper.ModelMapper;
6  import org.springframework.stereotype.Controller;
7  import org.springframework.ui.Model;
8  import org.springframework.web.bind.annotation.GetMapping;
9  import org.springframework.web.bind.annotation.PathVariable;
10 import org.springframework.web.bind.annotation.RequestMapping;
11 import org.springframework.web.bind.annotation.RequestMethod;
12
13 import it.aulab.progetto_finale_demo_doc.articles.Article;
14 import it.aulab.progetto_finale_demo_doc.articles.ArticleId;
15 import it.aulab.progetto_finale_demo_doc.articles.Category;
16 import it.aulab.progetto_finale_demo_doc.models.CategoryModel;
17 import it.aulab.progetto_finale_demo_doc.services.ArticleService;
18 import it.aulab.progetto_finale_demo_doc.services.CategoryService;
19
20 @Controller
21 @RequestMapping("/categories")
22 public class CategoryController {
23
24     @Autowired
25     private ArticleService articleService;
26
27     @Autowired
28     private CategoryService categoryService;
29
30     @Autowired
31     private ModelMapper modelMapper;
32
33     //Rotta per la ricerca dell'articolo in base alla categoria
34     @GetMapping("/search/{id}")
35     public String categorySearch(@PathVariable("id") Long id, Model viewModel) {
36         Category category = categoryService.read(id);
37
38         viewModel.addAttribute("title", "tutti gli articoli trovati per categoria " + category.getName());
39
40         List

articles = articleService.searchByCategory(modelMapper.map(category, destinationType:Category.class));
41         viewModel.addAttribute("articles", articles);
42
43         return "article/articles";
44     }
45
46 }


```

Il metodo appena creato non farà altro che caricare tutti gli articoli di una data categoria ed inviarli al template articles che ricordiamo essere il template di index degli articoli.

Questo metodo però richiede l'utilizzo del metodo "read" del "CategoryService" e del metodo "searchByCategory" di "ArticleService"

Iniziamo dal metodo read che ricordiamo esiste ma non possiede la giusta logica.

in "src\main\java\it\aulab\progetto_finale_docente\services\CategoryService.java"

```

src > main > java > it > aulab > progetto_finale_demo_doc > services > CategoryService.java > CategoryService > readAll()
You, 4 seconds ago | 2 authors (Alessandro Leo and one other)
1 package it.aulab.progetto_finale_demo_doc.services;
2
3 import java.security.Principal;
4 import java.util.List;
5 import java.util.ArrayList;
6 import org.modelmapper.ModelMapper;
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.stereotype.Service;
9 import org.springframework.web.multipart.MultipartFile;
10
11 import it.aulab.progetto_finale_demo_doc.dtos.CategoryDto;
12 import it.aulab.progetto_finale_demo_doc.models.Category;
13 import it.aulab.progetto_finale_demo_doc.repositories.CategoryRepository;
14
15 You, 1 second ago | 2 authors (Alessandro Leo and one other)
16 @Service
17 public class CategoryService implements CrudService<CategoryDto, Category, Long> {
18
19     @Autowired
20     private CategoryRepository categoryRepository;
21
22     @Autowired
23     private ModelMapper modelMapper;
24
25     @Override
26     public List<CategoryDto> readAll() {
27         List<CategoryDto> dtos = new ArrayList<CategoryDto>();
28         for(Category category: categoryRepository.findAll()){
29             dtos.add(modelMapper.map(category, destinationType<CategoryDto.class>));
30         }
31         return dtos;
32     }
33
34     @Override
35     public CategoryDto read(Long key) {
36         return modelMapper.map(categoryRepository.findById(key), destinationType<CategoryDto.class>);
37     }
38 }

```

E dedichiamoci al metodo "searchByCategory" di article service non ancora implementato, costruiamolo.

in "src\main\java\it\aulab\progetto_finale_docente\services\ArticleService.java"

```

src > main > java > it > aulab > progetto_finale_demo_doc > services > ArticleService.java > ArticleService > searchByCategory(Category)
You, 3 hours ago | 2 authors (Alessandro Leo and one other)
1 package it.aulab.progetto_finale_demo_doc.services;
2
3 import java.security.Principal;
4 import java.util.ArrayList;
5 import java.util.List;
6 import java.util.Optional;
7 import java.util.concurrent.CompletableFuture;
8
9 import org.springframework.security.core.context.SecurityContextHolder;
10 import org.springframework.stereotype.Service;
11 import org.springframework.web.multipart.MultipartFile;
12 import org.springframework.web.server.ResponseStatusException;
13 import org.modelmapper.ModelMapper;
14 import org.springframework.beans.factory.annotation.Autowired;
15 import org.springframework.http.HttpStatus;
16 import org.springframework.security.core.Authentication;
17
18 import it.aulab.progetto_finale_demo_doc.dtos.ArticleDto;
19 import it.aulab.progetto_finale_demo_doc.models.Article;
20 import it.aulab.progetto_finale_demo_doc.models.Category;
21 import it.aulab.progetto_finale_demo_doc.models.User;
22 import it.aulab.progetto_finale_demo_doc.repositories.ArticleRepository;
23 import it.aulab.progetto_finale_demo_doc.repositories.UserRepository;
24
25 You, 3 hours ago | 2 authors (Alessandro Leo and one other)
26 @Service
27 public class ArticleService implements CrudService<ArticleDto, Article, Long> {
28
29     @Override
30     public ArticleDto create(Article article, Principal principal, MultipartFile file) {
31
32         ArticleDto dto = new ArticleDto();
33         dto.setArticle(article);
34         dto.setPrincipal(principal);
35         dto.setFile(file);
36
37         return dto;
38     }
39
40     @Override
41     public ArticleDto update(Long key, Article model, MultipartFile file) {
42         // TODO Auto-generated method stub
43         throw new UnsupportedOperationException("Unimplemented method 'update'");
44     }
45
46     @Override
47     public void delete(Long key) {
48         // TODO Auto-generated method stub
49         throw new UnsupportedOperationException("Unimplemented method 'delete'");
50     }
51
52     @Override
53     public List<ArticleDto> searchByCategory(Category category) {
54
55         List<ArticleDto> dtos = new ArrayList<ArticleDto>();
56         for(Article article: articleRepository.findByCategory(category)){
57             dtos.add(modelMapper.map(article, destinationType<ArticleDto.class>));
58         }
59         return dtos;
60     }
61
62 }

```

```

src > main > java > it > aulab > progetto_finale_demo_doc > services > ArticleService.java > ArticleService > searchByCategory(Category)
60     public List<ArticleDto> searchByCategory(Category category) {
61
62         List<ArticleDto> dtos = new ArrayList<ArticleDto>();
63         for(Article article: articleRepository.findByCategory(category)){
64             dtos.add(modelMapper.map(article, destinationType<ArticleDto.class>));
65         }
66         return dtos;
67     }
68
69 }

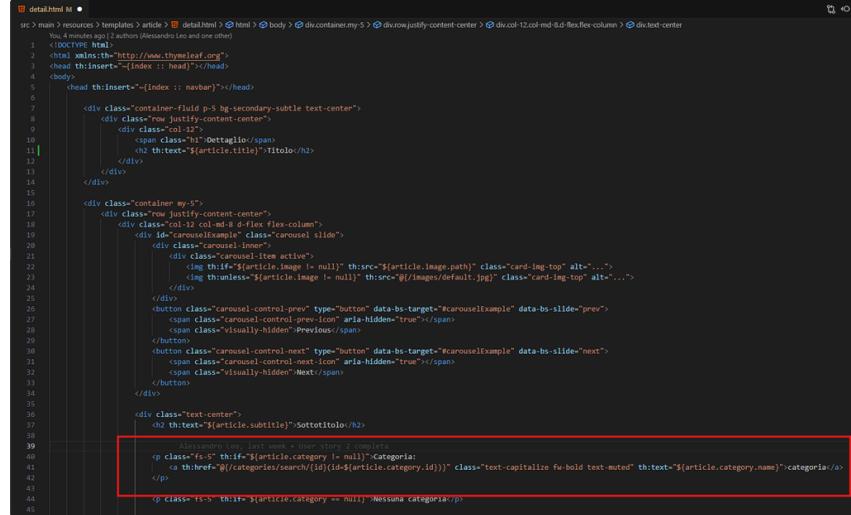
```

Ed importiamo quello che ci serve.

Fatto questo vedremo che non avremo più errori nel metodo precedentemente creato nel "CategoryController"

Ultimo step abilitare il link in tal modo che richiami l'handler e tutto ciò che abbiamo appena creato.

in "src\main\resources\templates\article\detail.html"



```

src > main > resources > templates > article > detail.html > HTML > body > div.container.my-5 > div.row.justify-content-center > div.col-12.col-md-8.d-flex.flex-column > div.text-center
You, 4 minutes ago | 2 authors (Alessandro Leo and one other)
1 <head>
2   html xmlns="http://www.thymeleaf.org">
3     head th:insert="${index :: head}"
4   body
5     head th:insert="${[index :: navbar]}"
6   /div
7 
8   div class="container-fluid p-5 bg-secondary-subtle text-center">
9     div class="row justify-content-center">
10       div class="col-12 col-md-8 d-flex flex-column">
11         div id="carouselExample" class="carousel slide">
12           div class="carousel-inner">
13             div th:if="${article.image != null}" th:src="${article.image.path}" class="card-img-top" alt="..."/>
14             img th:if="${article.image != null}" th:src="@{/images/default.jpg}" class="card-img-top" alt="..."/>
15           /div
16           button class="carousel-control-prev" type="button" data-bs-target="#carouselExample" data-bs-slide="prev">
17             span class="carousel-control-prev-icon" aria-hidden="true"></span>
18             span class="visually-hidden">Previous
19           /button
20           button class="carousel-control-next" type="button" data-bs-target="#carouselExample" data-bs-slide="next">
21             span class="carousel-control-next-icon" aria-hidden="true"></span>
22             span class="visually-hidden">Next
23           /button
24         /div
25       /div
26       div class="text-center">
27         h2 th:text="${article.title}"/h2
28         Alessandro Leo, last week + user story 2 complete
29         p class="fs-5" th:if="${article.category != null}">\${category}
30         th:href="#" th:if="${categorySearch[id eq ${article.category.id}]}" class="text-capitalize fw-bold text-muted" th:text="${article.category.name}">\${category}
31         p class="fs-5" th:if="!${title.id.category == null}"> Nessuna categoria
32       /div
33     /div
34   /div
35 
```

Facciamo un test e vedremo che tutto funziona correttamente

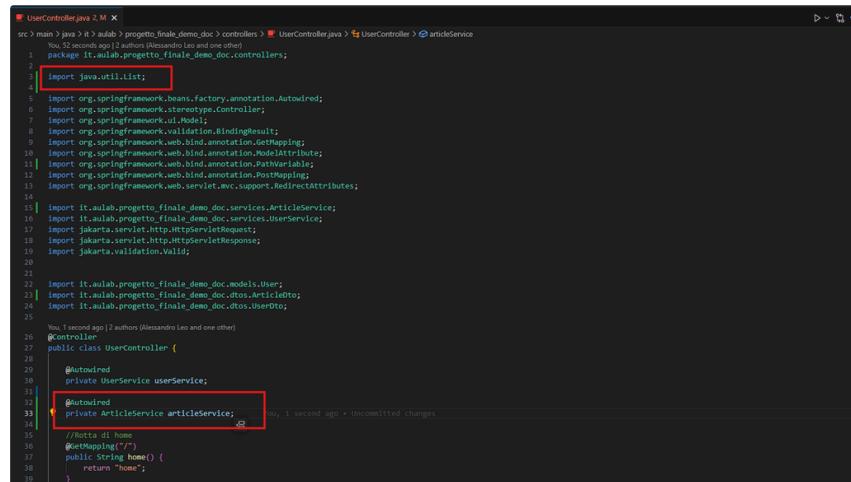
⚠ NB: Facciamo un test con un utente registrato, più avanti abiliteremo le rotte nel config per permettere anche ad un utente non loggato di poter accedere a queste rotte.

CLICK PER RICERCA PER AUTORE

Abilitiamo ora la ricerca per autore.

Andiamo all'interno del "UserController" ed aggiugiamo semplicemente la logica di ricerca.

in "src\main\java\it\aulab\progetto_finale_doc\controller\UserController.java"



```

src > main > java > it > aulab > progetto_finale_demo_doc > controllers > UserController.java > UserController > ArticleService
You, 32 seconds ago | 2 authors (Alessandro Leo and one other)
1 package it.aulab.progetto_finale_demo_doc.controllers;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Controller;
7 import org.springframework.validation.BindingResult;
8 import org.springframework.web.bind.annotation.GetMapping;
9 import org.springframework.web.bind.annotation.ModelAttribute;
10 import org.springframework.web.bind.annotation.PathVariable;
11 import org.springframework.web.bind.annotation.PostMapping;
12 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
13
14 import org.springframework.web.servlet.ModelAndView;
15
16 import it.aulab.progetto_finale_demo_doc.services.ArticleService;
17 import it.aulab.progetto_finale_demo_doc.services.UserService;
18 import jakarta.servlet.http.HttpServletRequest;
19 import jakarta.validation.Valid;
20
21
22 import it.aulab.progetto_finale_demo_doc.models.User;
23 import it.aulab.progetto_finale_demo_doc.dtos.ArticleDto;
24 import it.aulab.progetto_finale_demo_doc.dtos.UserDto;
25
26 You, 1 second ago | 2 authors (Alessandro Leo and one other)
27 @Controller
28 public class UserController {
29
30   @Autowired
31   private UserService userService;
32
33   @Autowired
34   private ArticleService articleService; , 1 second ago + Uncommitted changes
35
36   //Rotta di home
37   @GetMapping("/")
38   public String home() {
39     return "home";
40   }
41 }

```

```

UserController.java 2, M
src > main > java > it > aulab > progetto_finale_doc > controllers > UserController.java > UserController > articleService
27 public class UserController {
28
29     //Rotta per il salvataggio della registrazione
30     @PostMapping("register/save")
31     public String registration(@Valid @ModelAttribute("user") UserDto userDto,
32                             BindingResult result,
33                             Model model,
34                             RedirectAttributes redirectAttributes,
35                             HttpServletRequest request, HttpServletResponse response){
36
37         User existingUser = userService.findUserByEmail(userDto.getEmail());
38
39         if (existingUser != null && existingUser.getEmail() != null && !existingUser.getEmail().isEmpty()){
40             result.rejectValue("email", null,
41                                 "There is already an account registered with the same email");
42         }
43
44         if (result.hasErrors()){
45             model.addAttribute("user", userDto);
46             return "auth/register";
47         }
48
49         userService.saveUser(userDto, redirectAttributes, request, response);
50
51         redirectAttributes.addFlashAttribute(attributeName:"successMessage", attributeValue:"Registrazione avvenuta!");
52
53         return "redirect:/";
54     }
55
56     //Rotta per la ricerca degli articoli in base all'utente
57     @GetMapping("/search/{id}")
58     public String userArticlesSearch(@PathVariable("id") Long id, Model viewModel) {
59
60         User user = userService.find(id);
61         viewModel.addAttribute("title", "Tutti gli articoli trovati per utente " + user.getUsername());
62
63         List<ArticleDto> articles = articleService.searchByAuthor(user);
64         viewModel.addAttribute("articles", articles);
65
66         return "article/articles";
67     }
68
69 }

```

ed importiamo quello che ci serve.

Anche questo metodo ha bisogno di due metodi non ancora implementati. Per quanto riguarda il metodo “*find*” dobbiamo compiere due passaggi, il primo è creare la firma del metodo nel service.

in “src\main\java\it\aulab\progetto_finale_docente\services\UserService.java”

```

UserService.java M
src > main > java > it > aulab > progetto_finale_doc > services > UserService.java > find(Long)
You, 1 second ago | authors (Alessandro Leo and one other)
1 package it.aulab.progetto_finale_demo_doc.services;
2
3 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
4
5 import it.aulab.progetto_finale_demo_doc.dtos.UserDto;
6 import it.aulab.progetto_finale_demo_doc.models.User;
7 import jakarta.servlet.http.HttpServletRequest;
8 import jakarta.servlet.http.HttpServletResponse;
9
10 You, 1 second ago | authors (Alessandro Leo and one other)
11 public interface UserService {
12     void saveUser(UserDto userDto, RedirectAttributes redirectAttributes, HttpServletRequest request, HttpServletResponse response);
13     User findUserByEmail(String email);
14     User find(long id);
15 }

```

Il secondo è creare l’implementazione

in “src\main\java\it\aulab\progetto_finale_docente\services\UserServiceImpl.java”

```

UserServiceimpl.java M
src > main > java > it > aulab > progetto_finale_doc > services > UserServiceimpl.java > authenticateUserAndSetSession(User, UserDto, HttpServletRequest)
33 public class UserServiceimpl implements UserService {
34
35     public void saveUser(UserDto userDto, RedirectAttributes redirectAttributes, HttpServletRequest request, HttpServletResponse response){
36
37         Role role = roleRepository.findByName("ROLE_USER");
38         user.setRoles(List.of(role));
39
40         userRepository.save(user);
41
42         authenticateUserAndSetSession(user, userDto, request);
43     }
44
45     public void authenticateUserAndSetSession(User user, UserDto userDto, HttpServletRequest request) {
46
47         try {
48             CustomUserDetails userDetails = customUserDetailsService.loadUserByUsername(user.getEmail());
49
50             UsernamePasswordAuthenticationToken authoken = new UsernamePasswordAuthenticationToken(userDetails.getUsername(), userDto.getPassword());
51
52             Authentication authentication = authenticationManager.authenticate(authoken);
53
54             SecurityContextHolder.getContext().setAuthentication(authentication);
55
56             HttpSession session = request.getSession(create=true);
57             session.setAttribute(name:"SPRING_SECURITY_CONTEXT", SecurityContextHolder.getContext());
58
59         } catch (AuthenticationException e) {
60             e.printStackTrace();
61         }
62     }
63
64 }
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81     @Override
82     public User find(long id) {
83         return userRepository.findById(id).get();
84     }
85

```

Per ultimo poi la creazione del metodo “*searchByAuthor*” che attualmente non abbiamo ancora implementato, procediamo

in “src\main\java\it\aulab\progetto_finale_docente\services\ArticleService.java”

```

src > main > java > it > aulab > progetto_finale_demo_doc > services > ArticleService.java > ArticleService > searchByCategory(Category)
26 public class ArticleService implements CrudService<ArticleDto, Article, Long>
27 {
28     @Override
29     public ArticleDto update(Long key, Article model, MultipartFile file) {
30         // TODO Auto-generated method stub
31         throw new UnsupportedOperationException(message:"Unimplemented method 'update'");
32     }
33
34     @Override
35     public void delete(Long key) {
36         // TODO Auto-generated method stub
37         throw new UnsupportedOperationException(message:"Unimplemented method 'delete'");
38     }
39
40     public List<ArticleDto> searchByCategory(Category category){
41         List<ArticleDto> dtos = new ArrayList<ArticleDto>();
42         for(Article article: articleRepository.findByCategory(category)){
43             dtos.add(modelMapper.map(article, destinationType:ArticleDto.class));
44         }
45         You, 7 seconds ago + Uncommitted changes
46         return dtos;
47     }
48
49     public List<ArticleDto> searchByAuthor(User user){
50         List<ArticleDto> dtos = new ArrayList<ArticleDto>();
51         for(Article article: articleRepository.findByUser(user)){
52             dtos.add(modelMapper.map(article, destinationType:ArticleDto.class));
53         }
54         return dtos;
55     }
56 }

```

e andiamo ad abilitare il link in tal modo che richiami la nuova logica

in "src\main\resources\templates\article\detail.html"

```

src > main > resources > templates > article > detail.html > HTML > body > div.container.my-5 > div.row.justify-content-center > div.col-12.col-md-8.d-flex.flex-column > div.text-center > div.text-muted.my-3 > p > a.text-muted
1 <html xmlns="http://www.w3.org/1999/xhtml">
2     <head>
3         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
4         <title>Article Detail</title>
5         <link href="css/bootstrap.min.css" rel="stylesheet" type="text/css" />
6         <link href="css/style.css" rel="stylesheet" type="text/css" />
7     </head>
8     <body>
9         <div class="container my-5">
10             <div class="row justify-content-center">
11                 <div class="col-12 col-md-8 d-flex flex-column">
12                     <div class="d-flex align-items-center justify-content-between mb-3">
13                         <div>
14                             <img alt="Profile picture placeholder" class="img-fluid rounded-circle" style="width: 40px; height: 40px; border-radius: 50%; border: 1px solid #ccc;"/>
15                         <div>
16                             <button class="btn btn-primary" type="button" data-bs-target="#carouselExample" data-bs-slide="prev"><span class="fas fa-angle-left"></button>
17                             <button class="btn btn-primary" type="button" data-bs-target="#carouselExample" data-bs-slide="next"><span class="fas fa-angle-right"></button>
18                         <div id="carouselExample" class="carousel slide" data-bs-ride="carousel">
19                             <div class="carousel-inner">
20                                 <div class="carousel-item active">
21                                     
22                                 </div>
23                             </div>
24                             <div class="carousel-indicators" style="bottom: -10px; left: 50%; transform: translateX(-50%);">
25                                 <span data-bs-target="#carouselExample" data-bs-index="1" class="active" style="background-color: #ccc;">
26                                 <span data-bs-target="#carouselExample" data-bs-index="2" class="">
27                                 <span data-bs-target="#carouselExample" data-bs-index="3" class="">
28                             </div>
29                         </div>
30                         <div class="text-center">
31                             <h2>${article.subtitle}</h2>
32                             <h3>${article.title}</h3>
33                             <p class="fs-5" th:if="${article.category != null}">Categoria:<br/>
34                                 <a href="#">${category}</a>
35                             </p>
36                             <p class="fs-5" th:if="${article.category == null}">Nessuna categoria:<br/>
37                         </p>
38                         <div class="text-muted my-3">
39                             <p>Redatto il <span th:text="${article.publishDate}">data</span> da <span th:text="${article.user.username}">User</span> <small>Alessandro Leo last week + User story 2 completa</small>
40                         </div>
41                         <div>
42                             <ul class="list-group list-group-flush">
43                                 <li>${article.content}</li>
44                             </ul>
45                         </div>
46                     </div>
47                 </div>
48             </div>
49         </div>
50     </body>
51 </html>

```

Facciamo un test e vediamo che tutto funziona.

⚠ NB: Test sempre con utente loggato.

ABILITAZIONE ROTTE NEL SECURITY CONFIG

Ora notiamo immediatamente che, se non siamo registrati né loggati non riusciamo a visualizzare gli articoli per categoria o autore.

Molto semplicemente va aggiunta l'eccezione nella nostra configurazione di sicurezza

in "src\main\java\it\aulab\progetto_finale_docente\config\SecurityConfig.java"

modifichiamo in questo modo

```

    .requestMatchers(...patterns:"/register", "/", "/articles", "/images/**", "/articles/detail/**", "/categories/search/{id}", "/search/{id}").permitAll()

```

Facciamo un test e vediamo che tutto funziona anche con utenti non registrati o loggati.

VISUALIZZAZIONE CARD IN HOME

Ultimo step per poter completare questa user story è di far visualizzare nella home un numero di articoli a scelta partendo dal più recente

Andiamo a modificare quindi l'handler che ci indirizza alla home

in "src\main\java\it\aulab\progetto_finale_docente\controllers\UserController.java"

```

src > main > java > it > aulab > progetto_finale_demo_doc > controllers > UserController.java > UserController > home(Model)
1 package it.aulab.progetto_finale_demo_doc.controllers;
2
3 import java.util.Collections;
4 import java.util.Comparator;
5 import java.util.List;
6 import java.util.stream.Collectors;
7
8 import org.springframework.beans.factory.annotation.Autowired;
9

```

```

You, 1 hour ago | 2 authors (Alessandro Leo and one other)
@Controller
public class UserController {

    @Autowired
    private UserService userService;

    @Autowired
    private ArticleService articleService;

    //Rotta di home Alessandro Leo, 3 days ago + US01
    @GetMapping("/")
    public String home(Model viewModel) {
        List<ArticleDto> articles = articleService.readAll();

        //ordino e invio al template gli articoli ordinati in modo decrescente
        Collections.sort(articles, Comparator.comparing(ArticleDto::getPublishDate).reversed());

        List<ArticleDto> lastThreeArticles = articles.stream().limit(maxSize:3).collect(Collectors.toList());

        viewModel.addAttribute("articles", lastThreeArticles);

        return "home";
    }

    //Rotta per la registrazione
    @GetMapping("register")
    public String register(Model model) {
        model.addAttribute("user", new UserDto());
        return "auth/register";
    }
}

```

In questo metodo abbiamo applicato la stessa logica dell'index degli articoli ma con un passaggio in più, quello di recuperare solo i primi tre elementi della collezione.

Andiamo ora nel template "home" ed aggiungiamo il codice per la visualizzazione degli articoli inviati tramite l'handler

in "src\main\resources\templates\home.html"

```

home.html M X
src > main > resources > templates > home.html > HTML > body > div.container
2   <html lang="en"
6     <body>
8       <head> th:insert="~{index :: navbar}"</head>
9
10      <div th:if="${params.notAuthorized}">
11          <div class="alert alert-danger">
12              Not authorized!
13          </div>
14      </div>
15
16      <div th:if="${successMessage}" class="alert alert-success">
17          <p>${successMessage}</p>
18      </div>
19
20      <div class="container-fluid">
21          <div class="row justify-content-center mb-5">
22              <div class="col-10 mb-5">
23                  <h1 class="mb-5"> Welcome to <span class="display-4 fw-bold d-block">Aulab Chronicle</span></h1>
24
25              </div>
26          </div>
27
28          <div class="container" th:if="${articles.size() > 0}">
29              <div class="row justify-content-center">
30                  <div class="col-12 col-md-4 d-flex justify-content-center">
31                      <div class="text-center"> Gli ultimi articoli:</div>
32                  </div>
33
34                  <div class="row justify-content-center mb-5">
35                      <div th:each="article:${articles}" class="col-12 col-md-4 d-flex justify-content-center">
36                          <div class="card w-100 h-100">
37                              
38                              
39
40                              <div class="card-body">
41                                  <h3 class="card-title" th:text="${article.title}"></h3>
42                                  <p class="card-text" th:text="${article.subtitle}"></p>
43                                  <p class="card-text" th:text="${article.user.username}"></p>
44
45                                  <a th:href="@{/articles/detail/{id}}({id= ${article.id}})" class="btn btn-primary">Leggi</a>
46
47                              </div>
48
49                      </div>
50                  </div>
51          </div>
52      </div>
53
54  You, 3 seconds ago + Uncommitted changes

```

Facciamo un test e vediamo che tutto funziona correttamente.