



Xideral

Java Academy

Week 5

CRUD System with Spring Data JPA
and Spring Security

Presented by:

Edgar Itzak Sánchez Rogers

Monterrey Nuevo León México at 15 September 2024

Introduction:

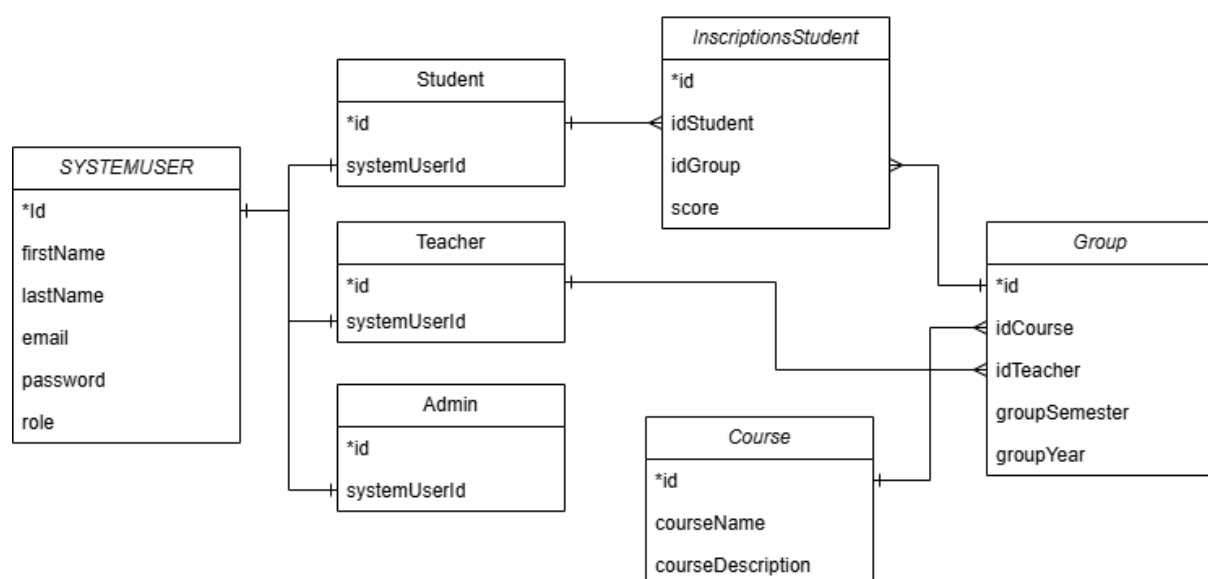
This is a project to demonstrate what I have learned during the academy in Java. This project consists of CRUD (CREATE READ UPDATE DELETE) using Spring boot, spring jpa and spring security. This project is able to manage 3 types of users: student, teacher, and admin. Each of them has different access permissions and roles. For example, admin user can create, delete and modify other users. The credentials of the created or modified users are updated so that when creating a new user, it is possible to log in with the new user. This project uses:

- Spring Boot
- Spring JPA
- Spring Security
- Lombok
- Thymeleaf
- Bootstrap

Entities:

Database Diagram

CRUD System with Spring Data JPA and Spring Security



System User:

```
1 package com.edgaritzak.gradeManagerSystem.entity;
2
3 import jakarta.persistence.*;
4 import lombok.AllArgsConstructor;
5 import lombok.Data;
6 import lombok.NoArgsConstructor;
7 import lombok.ToString;
8
9 @Entity
10 @Data
11 @ToString(exclude = {"student", "teacher"})
12 @AllArgsConstructor
13 @NoArgsConstructor
14 @Table(name="TBL_SYSTEM_USER")
15 public class SystemUser {
16
17     @Id
18     @Column(name="ID")
19     @GeneratedValue(strategy = GenerationType.IDENTITY)
20     private int id;
21     @Column(name="FIRST_NAME")
22     private String firstName;
23     @Column(name="LAST_NAME")
24     private String lastName;
25     @Column(name="EMAIL")
26     private String email;
27     @Column(name="PASSWORD")
28     private String password;
29     @Column(name="ROLE")
30     private String role;
31
32     //Relationships
33     @OneToOne(mappedBy = "systemUser")
34     private Student student;
35
36     @OneToOne(mappedBy = "systemUser")
37     private Teacher teacher;
38
39     @OneToOne(mappedBy = "systemUser")
40     private Admin admin;
41 }
```

Admin:

```
1 package com.edgaritzak.gradeManagerSystem.entity;
2
3 import jakarta.persistence.*;
4
5
6 @Entity
7 @Data
8 @AllArgsConstructor
9 @NoArgsConstructor
10 @Table(name="TBL_ADMIN")
11 public class Admin{
12
13     @Id
14     @Column(name="ID")
15     @GeneratedValue(strategy = GenerationType.IDENTITY)
16     private int id;
17     @OneToOne
18     @JoinColumn(name = "ID_SYSTEM_USER")
19     private SystemUser systemUser;
20
21 }
```

Teacher:

```
1 package com.edgaritzak.gradeManagerSystem.entity;
2
3 import java.util.List;
4
5 @Entity
6 @Data
7 @ToString(exclude = "group")
8 @AllArgsConstructor
9 @NoArgsConstructor
10 @Table(name="TBL_TEACHER")
11 public class Teacher{
12
13     @Id
14     @Column(name="ID")
15     @GeneratedValue(strategy = GenerationType.IDENTITY)
16     private int id;
17
18     @OneToOne
19     @JoinColumn(name = "ID_SYSTEM_USER")
20     private SystemUser systemUser;
21
22     //Relationships
23     @OneToMany(mappedBy = "teacher")
24     private List<Group> group;
25 }
26
27
```

Student:

```
1 package com.edgaritzak.gradeManagerSystem.entity;
2
3 import java.util.List;
4
5 @Entity
6 @Data
7 @ToString(exclude = "inscriptionsStudents")
8 @AllArgsConstructor
9 @NoArgsConstructor
10 @Table(name="TBL_STUDENT")
11 public class Student{
12
13     @Id
14     @Column(name="ID")
15     @GeneratedValue(strategy = GenerationType.IDENTITY)
16     private int id;
17
18     @OneToOne
19     @JoinColumn(name = "ID_SYSTEM_USER")
20     private SystemUser systemUser;
21
22     //Relationships
23     @OneToMany(mappedBy = "student")
24     private List<InscriptionsStudents> inscriptionsStudents;
25 }
26
27
```

Course:

```
1 package com.edgaritzak.gradeManagerSystem.entity;
2
3 import java.util.List;
4
5 @Entity
6 @Data
7 @ToString(exclude = "group")
8 @AllArgsConstructor
9 @NoArgsConstructor
10 @Table(name="TBL_COURSE")
11 public class Course {
12
13     @Id
14     @Column(name="ID")
15     @GeneratedValue(strategy= GenerationType.IDENTITY)
16     private int id;
17     @Column(name="COURSE_NAME")
18     private String name;
19     @Column(name="COURSE_DESCRIPTION")
20     private String description;
21
22     //Relationships
23     @OneToMany(mappedBy = "course")
24     private List<Group> group;
25 }
```

Inscriptions Students:

```
1 package com.edgaritzak.gradeManagerSystem.entity;
2
3 import jakarta.persistence.*;
4
5 @Entity
6 @Data
7 @AllArgsConstructor
8 @NoArgsConstructor
9 @Table(name="TBL_INSCRIPTIONS")
10 public class InscriptionsStudents {
11
12     @Id
13     @Column(name="ID")
14     @GeneratedValue(strategy=GenerationType.IDENTITY)
15     private int id;
16     @ManyToOne
17     @JoinColumn(name="ID_GROUP")
18     private Group group;
19     @ManyToOne
20     @JoinColumn(name="ID_STUDENT")
21     private Student student;
22     @Column(name="SCORE")
23     private int score;
24 }
```

Group:

```
1 package com.edgaritzak.gradeManagerSystem.entity;
2
3 import java.util.List;
4
5 @Entity
6 @Data
7 @ToString(exclude = "inscriptionsStudents")
8 @AllArgsConstructor
9 @NoArgsConstructor
10 @Table(name="TBL_GROUP")
11 public class Group {
12
13     @Id
14     @Column(name="ID")
15     @GeneratedValue(strategy= GenerationType.IDENTITY)
16     private int id;
17
18     @ManyToOne
19     @JoinColumn(name = "ID_COURSE")
20     private Course course;
21
22     @ManyToOne
23     @JoinColumn(name = "ID_TEACHER")
24     private Teacher teacher;
25
26     @Column(name="GROUP_SEMESTER")
27     private int semester;
28
29     @Column(name="GROUP_YEAR")
30     private int year;
31
32     //Relationships
33     @OneToMany(mappedBy="group")
34     private List<InscriptionsStudents> inscriptionsStudents;
35 }
```

DTO

Group With Score

```
1 package com.edgaritzak.gradeManagerSystem.dto;
2
3 public class CourseGroupWithScoreDTO {
4
5     private int idInscription;
6     private int idGroup;
7     private String courseName;
8     private String courseDescription;
9     private String teacherName;
10    private int semester;
11    private int year;
12    private double score;
13
14    public CourseGroupWithScoreDTO(int idInscription, int idGroup, String
15        teacherName, int semester, int year, double score) {
16        super();
17        this.idInscription = idInscription;
18        this.idGroup = idGroup;
19        this.courseName = courseName;
20        this.courseDescription = courseDescription;
21        this.teacherName = teacherName;
22        this.semester = semester;
23        this.year = year;
24        this.score = score;
25    }
26
27    public String getTeacherName() {
28        return teacherName;
29    }
30
31    public void setTeacherName(String teacherName) {
32        this.teacherName = teacherName;
33    }
34 }
```

Group With Course

```
1 package com.edgaritzak.gradeManagerSystem.dto;
2
3 public class GroupWithCourseDTO {
4
5     private int groupId;
6     private String courseName;
7     private String description;
8     private int semester;
9     private int year;
10    private String teacher;
11
12    public GroupWithCourseDTO(int groupId, String courseName, String description, int semester,
13        super();
14        this.groupId = groupId;
15        this.courseName = courseName;
16        this.description = description;
17        this.semester = semester;
18        this.year = year;
19    }
20
21    public GroupWithCourseDTO(int groupId, String courseName, String description, int semester,
22        super();
23        this.groupId = groupId;
24        this.courseName = courseName;
25        this.description = description;
26        this.semester = semester;
27        this.year = year;
28        this.teacher = teacher;
29    }
30
31    public int getGroupId() {
32        return groupId;
33    }
```

Student with score:

```
1 package com.edgaritzak.gradeManagerSystem.dto;
2
3 public class StudentWithScoreDTO {
4
5     private int idInscription;
6     private int idStudent;
7     private String firstName;
8     private String lastName;
9     private int score;
10
11    public StudentWithScoreDTO(int idInscription, int idStudent, String firstName, String lastName,
12        this.idInscription = idInscription;
13        this.idStudent = idStudent;
14        this.firstName = firstName;
15        this.lastName = lastName;
16        this.score = score;
17    }
18
19
20    public int getIdInscription() {
21        return idInscription;
22    }
23
24
25    public void setIdInscription(int idInscription) {
26        this.idInscription = idInscription;
27    }
28
29
30    public int getIdStudent() {
31        return idStudent;
32    }
33    }
```

System User with Role Id:

```

1 package com.edgaritzak.gradeManagerSystem.dto;
2
3 public class SystemUserWithRoleIdDTO {
4
5     private int id;
6     private String firstName;
7     private String lastName;
8     private String email;
9     private String password;
10    private String role;
11    private int roleId;
12
13    public SystemUserWithRoleIdDTO(int id, String firstName, String lastName, String email,
14        String role, int roleId) {
15        this.id = id;
16        this.firstName = firstName;
17        this.lastName = lastName;
18        this.email = email;
19        this.password = password;
20        this.role = role;
21        this.roleId = roleId;
22    }
23
24    public int getId() {
25        return id;
26    }
27
28    public void setId(int id) {
29        this.id = id;
30    }
31
32    public String getFirstName() {
33        return firstName;

```

Repositories:

```

1 package com.edgaritzak.gradeManagerSystem.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5
6 public interface CourseRepository extends JpaRepository<Course,Integer>{
7
8 }

```

```

1 package com.edgaritzak.gradeManagerSystem.repository;
2
3 import java.util.List;
4
11 public interface InscriptionsStudentsRepository extends JpaRepository<InscriptionsStudents, Integer>{
12
13     //GET INSCRIPTIONS WHERE GROUPID
14     @Query("Select I from InscriptionsStudents AS I JOIN I.group AS G WHERE G.id = :groupId")
15     List<InscriptionsStudents> findInscriptionsStudentsByGroupId(@Param("groupId")int groupId);
16
17     //UPDATE INSCRIPTION SCORE
18     @Modifying
19     @Query("Update InscriptionsStudents I SET I.score = :score where I.id = :idInscription")
20     void updateScore(@Param("idInscription")int idInscription,@Param("score") double score);
21
22
23
24     @Query("Select I from InscriptionsStudents AS I WHERE I.student.id = :studentId")
25     List<InscriptionsStudents> findInscriptionsStudentsByStudentId(@Param("studentId")int studentId);
26
27     @Modifying
28     @Query("Delete from InscriptionsStudents AS I WHERE I.student.id = :studentId")
29     void deleteByStudentId(@Param("studentId")int id);
30 }

```


Services

Group service:

```
1 package com.edgaritzak.gradeManagerSystem.service;
2
3 import java.util.List;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21 @Service
22 public class GroupServiceImpl implements GroupService {
23
24     CourseRepository courseRepo;
25     GroupRepository groupRepo;
26     IncriptionsStudentsRepository inscriptionsStudentsRepo;
27     @Autowired
28     public GroupServiceImpl(GroupRepository groupRepo, IncriptionsStudentsRepository inscriptionsStudentsRepo,
29                             CourseRepository courseRepo) {
30         this.groupRepo = groupRepo;
31         this.inscriptionsStudentsRepo = inscriptionsStudentsRepo;
32         this.courseRepo = courseRepo;
33     }
34
35     @Transactional
36     public void deleteAllInscriptionsByStudentId(int id) {
37         inscriptionsStudentsRepo.deleteByStudentId(id);
38     }
39
40     @Override
41     public Group findGroupById(int id) {
42         Group group = groupRepo.findById(id).orElseThrow(() -> new NoSuchElementException("No value found"));
43         return group;
44     }
45
46     @Override
47     public List<Group> findTeachersGroups(int teacherId) {
48         return groupRepo.findTeachersGroups(teacherId);
49     }
50 }
```

Inscriptions Student Service:

```
1 package com.edgaritzak.gradeManagerSystem.service;
2
3 import java.util.List;
4
5
6
7
8
9
10
11
12
13
14
15
16 @Service
17 public class InscriptionsStudentsServiceImpl implements InscriptionsStudentsService {
18     InscriptionsStudentsRepository inscriptionsStudentsRepo;
19
20
21     @Autowired
22     public InscriptionsStudentsServiceImpl(InscriptionsStudentsRepository inscriptionsStudentsRepo) {
23         this.inscriptionsStudentsRepo = inscriptionsStudentsRepo;
24     }
25
26     @Transactional
27     @Override
28     public void updateScore(int idInscription, double score) {
29         inscriptionsStudentsRepo.updateScore(idInscription, score);
30     }
31
32     @Override
33     public InscriptionsStudents findInscriptionsStudentsById(int id) {
34         return inscriptionsStudentsRepo.findById(id).orElseThrow(() -> new NoSuchElementException("No
35     })
36
37     @Override
38     public List<InscriptionsStudents> findInscriptionsStudentsByGroupId(int groupId) {
39         return inscriptionsStudentsRepo.findInscriptionsStudentsByGroupId(groupId);
40     }
41     @Override
42     public List<InscriptionsStudents> findInscriptionsStudentsByStudentId(int studentId) {
43         return inscriptionsStudentsRepo.findInscriptionsStudentsByStudentId(studentId);
44     }
45 }
```

Student Service Impl:

```
1 package com.edgaritzak.gradeManagerSystem.service;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5
6
7
8 @Service
9 public class StudentServiceImpl implements StudentService{
10
11     StudentRepository studentRepo;
12
13     @Autowired
14     public StudentServiceImpl(StudentRepository groupRepo) {
15         this.studentRepo = groupRepo;
16     }
17
18     @Override
19     public Student findBySystemUserId(int systemUserId) {
20         return studentRepo.findBySystemUserId(systemUserId);
21     }
22 }
```

System User Service Impl:

```
1 package com.edgaritzak.gradeManagerSystem.service;
2
3 import java.util.ArrayList;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21 @Service
22 public class SystemUserServiceImpl implements SystemUserService{
23
24     SystemUserRepository systemUserRepo;
25     InscriptionsStudentsRepository inscriptionsStudentsRepo;
26     StudentRepository studentRepository;
27     TeacherRepository teacherRepository;
28
29     @Autowired
30     public SystemUserServiceImpl(SystemUserRepository systemUserRepo,
31         InscriptionsStudentsRepository inscriptionsStudentsRepo, StudentRepository studentRepository,
32         TeacherRepository teacherRepository) {
33         this.systemUserRepo = systemUserRepo;
34         this.inscriptionsStudentsRepo = inscriptionsStudentsRepo;
35         this.studentRepository = studentRepository;
36         this.teacherRepository = teacherRepository;
37     }
38
39     public Student saveStudent(Student student) {
40         return studentRepository.save(student);
41     }
42     public Teacher saveTeacher(Teacher teacher) {
43         return teacherRepository.save(teacher);
44     }
45
46     public SystemUser save(SystemUser systemUser) {
47         return systemUserRepo.save(systemUser);
48     }
49
50     public void deleteById(int id) {
```

Teacher Service Impl:

```
1 package com.edgaritzak.gradeManagerSystem.service;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5
6
7
8 @Service
9 public class TeacherServiceImpl implements TeacherService {
10
11     TeacherRepository teacherRepo;
12
13     @Autowired
14     public TeacherServiceImpl(TeacherRepository teacherRepo) {
15         this.teacherRepo = teacherRepo;
16     }
17
18     @Override
19     public Teacher findBySystemUserId(int systemUserId) {
20         return teacherRepo.findBySystemUserId(systemUserId);
21     }
22 }
```

User Session Service:

```
1 package com.edgaritzak.gradeManagerSystem.service;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5 @Service
6 public class UserSessionServiceImpl implements SessionService {
7
8     private SystemUserRepository systemUserRepo;
9     @Autowired
10    public UserSessionServiceImpl(SystemUserRepository systemUserRepo) {
11        this.systemUserRepo = systemUserRepo;
12    }
13
14    //SYSTEMUSER TRANSFORM DTO (BY EMAIL)
15    @Override
16    public SystemUserWithRoleIdDTO findUserWithRoleDTOByEmail(String email) {
17        SystemUser tempUser = systemUserRepo.findByEmail(email);
18        int roleId = 0;
19        if (tempUser.getRole().equals("STUDENT")) roleId = tempUser.getStudent().getId();
20        if (tempUser.getRole().equals("TEACHER")) roleId = tempUser.getTeacher().getId();
21        if (tempUser.getRole().equals("ADMIN")) roleId = tempUser.getAdmin().getId();
22        SystemUserWithRoleIdDTO user = new SystemUserWithRoleIdDTO(
23            tempUser.getId(),
24            tempUser.getFirstName(),
25            tempUser.getLastName(),
26            tempUser.getEmail(),
27            tempUser.getPassword(),
28            tempUser.getRole(),
29            roleId
30        );
31        return user;
32    }
33 }
```

Spring Security Config

In Memory User Details Manager:

```
22 @Configuration
23 public class SecurityConfig {
24
25     @Autowired
26     private SystemUserServiceImpl systemUserServiceImpl;
27     private final InMemoryUserDetailsManager userDetailsManager;
28
29     public SecurityConfig() {
30         this.userDetailsManager = new InMemoryUserDetailsManager();
31     }
32
33     @Bean
34     public InMemoryUserDetailsManager userDetailsManager() {
35         return userDetailsManager;
36     }
37     //ADD USERS
38     @PostConstruct
39     public void initializeUsers() {
40         updateUsers();
41     }
42
43     public void updateUsers() {
44         List<UserDetails> userList = systemUserServiceImpl.findAll()
45             .stream()
46             .map(x -> User.builder()
47                 .username(x.getEmail())
48                 .password("{noop}" + x.getPassword())
49                 .roles(x.getRole())
50                 .build())
51             .collect(Collectors.toList());
52
53         userList.forEach((user) -> userDetailsManager.createUser(user));
54     }
55     //REMOVE USERS
56     public void removeUsers() {
57         List<SystemUser> userList = systemUserServiceImpl.findAll();
58         userList.forEach((user) -> userDetailsManager.deleteUser(user.getEmail()));
59     }
60 }
```

Security Filter Chain:

```
62 @Bean
63 public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
64     http
65         .authorizeHttpRequests(authorizeRequests ->
66             authorizeRequests
67                 .requestMatchers("/student/").hasRole("STUDENT")
68                 .requestMatchers("/student/**").hasRole("STUDENT")
69                 .requestMatchers("/teacher/").hasRole("TEACHER")
70                 .requestMatchers("/teacher/**").hasRole("TEACHER")
71                 .requestMatchers("/admin/").hasRole("ADMIN")
72                 .requestMatchers("/admin/**").hasRole("ADMIN")
73                 .anyRequest().authenticated());
74
75     http.httpBasic(Customizer.withDefaults())
76     .formLogin(formLogin ->
77         formLogin
78             .loginPage("/login")
79             .defaultSuccessUrl("/auth", true)
80             .permitAll()
81             .logout(logout ->
82                 logout
83                     .logoutUrl("/logout")
84                     .logoutSuccessUrl("/login?logout")
85                     .clearAuthentication(true)
86                     .deleteCookies("JSESSIONID")
87                     .invalidateHttpSession(true)
88                     .permitAll())
89             .sessionManagement(sessionManagement -> sessionManagement.sessionCreationPolicy(
90                 SessionManagementPolicy.STATELESS))
91             .csrf(csrf -> csrf.disable());
92     return http.build();
93 }
94 }
```

Controllers

Session Controller:

```
1 package com.edgaritzak.gradeManagerSystem.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5
6 @Controller
7 public class SessionController {
8
9     @Autowired
10     private UserSessionServiceImpl userSessionServiceImpl;
11
12     @GetMapping("/login")
13     public String siteHome(Model model, @RequestParam(value = "error", required = false) String error,
14         @RequestParam(value = "logout", required = false) String logout) {
15         if (error != null) {
16             model.addAttribute("error", "Invalid username or password.");
17         }
18         if (logout != null) {
19             model.addAttribute("logout", "You have been logged out.");
20         }
21         return "login";
22     }
23
24     @GetMapping("/auth")
25     public String authentication(HttpSession session) {
26         SystemUserWithRoleIdDTO user = userSessionServiceImpl.findUserWithRoleIdByEmail(
27             SecurityContextHolder.getContext().getAuthentication().getName());
28         String userRole = user.getRole();
29
30         //SESSION VARIABLES
31         session.setAttribute("systemUserId", user.getId());
32         session.setAttribute("firstName", user.getFirstName());
33         session.setAttribute("lastName", user.getLastName());
34         session.setAttribute("email", user.getEmail());
35         session.setAttribute("password", user.getPassword());
36         session.setAttribute("role", user.getRole());
37         session.setAttribute("roleId", user.getRoleId());
38
39         if(userRole.equals("STUDENT")) return "redirect:/student";
40         if(userRole.equals("TEACHER")) return "redirect:/teacher";
41         if(userRole.equals("ADMIN")) return "redirect:/admin";
42         return "redirect:/error";
43     }
44
45     @ResponseBody
46     @GetMapping("/error")
47     public String error() {
48         return "Welcome to error page";
49     }
50 }
```

Student Controller:

```
1 package com.edgaritzak.gradeManagerSystem.controller;
2
3 import java.util.HashMap;
4
5
6
7
8
9
10
11
12
13
14
15
16 @Controller
17 @RequestMapping("/student")
18 public class StudentController {
19
20
21     private GroupServiceImpl groupServiceImpl;
22     @Autowired
23     public StudentController(GroupServiceImpl groupServiceImpl) {
24         this.groupServiceImpl = groupServiceImpl;
25     }
26
27
28     @GetMapping("/userDetails")
29     public String userDetails(Model model, HttpSession session) {
30         Map<String, Object> attributes = new HashMap<>();
31
32         attributes.put("systemUserId", session.getAttribute("systemUserId"));
33         attributes.put("firstName", session.getAttribute("firstName"));
34         attributes.put("lastName", session.getAttribute("lastName"));
35         attributes.put("email", session.getAttribute("email"));
36         attributes.put("password", session.getAttribute("password"));
37         attributes.put("role", session.getAttribute("role"));
38         attributes.put("roleId", session.getAttribute("roleId"));
39         model.addAttribute("userAttributes", attributes);
40
41         return "userDetails";
42     }
43
44     @GetMapping("")
45     public String studentGroup(Model model, HttpSession session) {
46         Map<String, Object> attributes = new HashMap<>();
47
48         attributes.put("systemUserId", session.getAttribute("systemUserId"));
49         attributes.put("firstName", session.getAttribute("firstName"));
50         attributes.put("lastName", session.getAttribute("lastName"));
51         attributes.put("email", session.getAttribute("email"));
52         attributes.put("password", session.getAttribute("password"));
53         attributes.put("role", session.getAttribute("role"));
54         attributes.put("roleId", session.getAttribute("roleId"));
55         model.addAttribute("userAttributes", attributes);
56
57         List<CourseGroupWithScoreDTO> groupList =
58             groupServiceImpl.findStudentsCourseGroupWithScoreDTO(
59                 (Integer)session.getAttribute("roleId"));
60         model.addAttribute("groupList", groupList);
61
62         if (groupList.isEmpty()) {
63             model.addAttribute("emptyGroupMessage", "No groups found.");
64         }
65
66         return "studentGroup";
67     }
68 }
69
70 }
```

Teacher controller:


```

1 package com.edgaritzak.gradeManagerSystem.controller;
2
3 import java.util.HashMap;
4
5 @Controller
6 @RequestMapping("/teacher")
7 public class TeacherController {
8
9     private SystemUserService systemUserService;
10    private GroupServiceImpl groupServiceImpl;
11    private InscriptionsStudentsServiceImpl inscriptionsStudentsService;
12
13    @Autowired
14    public TeacherController(SystemUserService systemUserService,
15                             GroupServiceImpl groupServiceImpl, InscriptionsStudentsServiceImpl in
16                             scriptionsStudentsService) {
17        super();
18        this.systemUserService = systemUserService;
19        this.groupServiceImpl = groupServiceImpl;
20        this.inscriptionsStudentsService = inscriptionsStudentsService;
21    }
22
23    @GetMapping("")
24    public String teacherGroup(Model model, HttpSession session) {
25        Map<String, Object> attributes = new HashMap<>();
26
27        attributes.put("systemUserId", session.getAttribute("systemUserId"));
28        attributes.put("firstName", session.getAttribute("firstName"));
29        attributes.put("lastName", session.getAttribute("lastName"));
30        attributes.put("email", session.getAttribute("email"));
31        attributes.put("password", session.getAttribute("password"));
32        attributes.put("role", session.getAttribute("role"));
33        attributes.put("roleId", session.getAttribute("roleId"));
34        model.addAttribute("userAttributes", attributes);
35
36        List<GroupWithCourseDTO> groupList =
37            groupServiceImpl.findTeachersGroupWithCourseDTO(
38                (Integer)session.getAttribute("roleId"));
39        model.addAttribute("groupList", groupList);
40
41        if (groupList.isEmpty()) {
42            model.addAttribute("emptyGroupMessage", "No groups found.");
43        }
44        return "teacherGroup";
45    }
46
47    @GetMapping("/userDetails")
48    public String userDetails(Model model, HttpSession session) {
49        Map<String, Object> attributes = new HashMap<>();
50
51        attributes.put("systemUserId", session.getAttribute("systemUserId"));
52        attributes.put("firstName", session.getAttribute("firstName"));
53        attributes.put("lastName", session.getAttribute("lastName"));
54        attributes.put("email", session.getAttribute("email"));
55        attributes.put("password", session.getAttribute("password"));
56        attributes.put("role", session.getAttribute("role"));
57        attributes.put("roleId", session.getAttribute("roleId"));
58        model.addAttribute("userAttributes", attributes);
59
60        return "userDetails";
61    }
62
63 }

```

```

79= @GetMapping("/groupDetails")
80 public String showGroupStudents(@RequestParam("groupId") int groupId, Model model) {
81     Map<String, Object> courseAttributes = new HashMap<>();
82     GroupWithCourseDTO group = groupServiceImpl.findGroupWithCourseDTOByGroupId(groupId);
83     courseAttributes.put("groupId", group.getGroupId());
84     courseAttributes.put("courseName", group.getCourseName());
85     courseAttributes.put("description", group.getDescription());
86     courseAttributes.put("semester", group.getSemester());
87     courseAttributes.put("year", group.getYear());
88     model.addAttribute("courseAttributes", courseAttributes);
89
90
91     List<StudentWithScoreDTO> studentList = systemUserServiceImpl.findStudentsWithScoreByGroup(
92     model.addAttribute("studentList", studentList);
93     if (studentList.isEmpty()) {
94         model.addAttribute("emptyGroupMessage", "No students found.");
95     }
96
97     return "groupDetails";
98 }
99
100= @GetMapping("/updateGroupDetails")
101 public String updateGroupStudents(@RequestParam("groupId") int groupId, Model model) {
102     Map<String, Object> courseAttributes = new HashMap<>();
103     GroupWithCourseDTO group = groupServiceImpl.findGroupWithCourseDTOByGroupId(groupId);
104     courseAttributes.put("groupId", group.getGroupId());
105     courseAttributes.put("courseName", group.getCourseName());
106     courseAttributes.put("description", group.getDescription());
107     courseAttributes.put("semester", group.getSemester());
108     courseAttributes.put("year", group.getYear());
109     model.addAttribute("courseAttributes", courseAttributes);
110
111
112     List<StudentWithScoreDTO> studentList = systemUserServiceImpl.findStudentsWithScoreByGroup(
113     model.addAttribute("studentList", studentList);
114     if (studentList.isEmpty()) {
115         model.addAttribute("emptyGroupMessage", "No students found.");
116     }
117
118     return "updateGroupDetails";
119 }
120
121= @PostMapping("/update")
122 public String update(@RequestParam Map<String, String> params, @RequestParam("groupId")String g
123     System.out.println("TEST");
124     Map<Integer, Integer> valuesToUpdate = new HashMap<>();
125     for (String key : params.keySet()) {
126         if (key.startsWith("score-")) {
127             Integer idInscription = Integer.valueOf(key.substring("score-".length()));
128             double doubleScore = Double.valueOf(params.get(key));
129             Integer score = (int)doubleScore;
130             //ADD VALUES TO UPDATE (ID) (SCORE)
131             valuesToUpdate.put(idInscription, score);
132         }
133     }
134     //UPDATE
135     inscriptionsStudentsServiceImpl.updateScores(valuesToUpdate);
136     return "redirect:/teacher/groupDetails?groupId="+groupId;
137 }
138 }

```

Admin controller:

```

1 package com.edgaritzak.gradeManagerSystem.controller;
2
3 import java.util.HashMap;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28 @Controller
29 @RequestMapping("/admin")
30 public class AdminController {
31
32     @Autowired
33     private SecurityConfig securityConfig;
34
35     private UserSessionServiceImpl userSessionServiceImpl;
36     private SystemUserServiceImpl systemUserServiceImpl;
37     private GroupServiceImpl groupServiceImpl;
38
39     @Autowired
40     public AdminController(SystemUserServiceImpl systemUserServiceImpl, Student
41         GroupServiceImpl groupServiceImpl, InscriptionsStudentsServiceImpl
42         UserSessionServiceImpl userSessionServiceImpl) {
43         super();
44         this.systemUserServiceImpl = systemUserServiceImpl;
45         this.groupServiceImpl = groupServiceImpl;
46         this.userSessionServiceImpl = userSessionServiceImpl;
47     }
48
49
50     @GetMapping("")
51     public String teacherGroup(Model model, HttpSession session) {
52         Map<String, Object> attributes = new HashMap<>();
53
54         attributes.put("systemUserId", session.getAttribute("systemUserId"));
55         attributes.put("firstName", session.getAttribute("firstName"));
56         attributes.put("lastName", session.getAttribute("lastName"));
57         attributes.put("email", session.getAttribute("email"));
58         attributes.put("password", session.getAttribute("password"));
59         attributes.put("role", session.getAttribute("role"));
60         attributes.put("roleId", session.getAttribute("roleId"));
61         model.addAttribute("userAttributes", attributes);
62
63         return "adminHome";
64     }
65
66     @GetMapping("/userDetails")
67     public String userDetails(Model model, HttpSession session) {
68         Map<String, Object> attributes = new HashMap<>();
69
70         attributes.put("systemUserId", session.getAttribute("systemUserId"));
71         attributes.put("firstName", session.getAttribute("firstName"));
72         attributes.put("lastName", session.getAttribute("lastName"));
73         attributes.put("email", session.getAttribute("email"));
74         attributes.put("password", session.getAttribute("password"));
75         attributes.put("role", session.getAttribute("role"));
76         attributes.put("roleId", session.getAttribute("roleId"));
77         model.addAttribute("userAttributes", attributes);
78         return "userDetails";
79     }
80
81     @GetMapping("/courseList")
82     public String showCourseList(Model model) {
83         List<Course> courseList = groupServiceImpl.findAllCourses();
84         model.addAttribute("courseList", courseList);
85         if (courseList.isEmpty()) {
86             model.addAttribute("emptyGroupMessage", "No courses found.");
87         }
88     }
89 }

```

```

87         }
88         return "adminCourseList";
89     }
90     @GetMapping("/groupList")
91     public String showGroupList(Model model) {
92         List<GroupWithCourseDTO> groupList = groupServiceImpl.findAllGroupWithCourseDTO();
93         model.addAttribute("groupList", groupList);
94         if (groupList.isEmpty()) {
95             model.addAttribute("emptyGroupMessage", "No group found.");
96         }
97         return "adminGroupList";
98     }
99
100
101 //
102 // STUDENT
103 //
104
105     @GetMapping("/studentList")
106     public String showGroupStudents(Model model) {
107         List<SystemUserWithRoleIdDTO> studentList = systemUserServiceImpl.findAllStudentsDTO();
108         model.addAttribute("studentList", studentList);
109         if (studentList.isEmpty()) {
110             model.addAttribute("emptyGroupMessage", "No students found.");
111         }
112         return "adminStudentList";
113     }
114
115     @GetMapping("/createStudent")
116     public String createStudents(Model model) {
117         List<SystemUserWithRoleIdDTO> studentList = systemUserServiceImpl.findAllStudentsDTO();
118         model.addAttribute("studentList", studentList);
119         if (studentList.isEmpty()) {
120             model.addAttribute("emptyGroupMessage", "No students found.");
121         }
122         return "adminNewStudent";
123     }
124
125     @PostMapping("/createStudent/save")
126     public String saveNewStudents(@RequestParam("firstName")String firstName,
127                                   @RequestParam("lastName")String lastName, @RequestParam("email")String email,
128                                   @RequestParam("password")String password) {
129
130
131         securityConfig.removeUsers();
132
133         SystemUser user =systemUserServiceImpl.save(new SystemUser(0,firstName,lastName,email,password));
134         user.setStudent(systemUserServiceImpl.saveStudent(new Student(0,user,null)));
135         systemUserServiceImpl.save(user);
136         securityConfig.updateUsers();
137
138         return "redirect:/admin/studentList";
139     }
140
141
142     @GetMapping("/editStudent")
143     public String updateStudent(@RequestParam("id") int id, Model model) {
144         SystemUserWithRoleIdDTO student = userSessionServiceImpl.findUserWithRoleIdDTOBySystemId(id);
145         model.addAttribute("student", student);
146         return "adminEditStudent";
147     }

```

```

149@ @PostMapping("/editStudent/updateStudent")
150 public String update(@RequestParam("id")String id, @RequestParam("firstName")String firstName,
151 @RequestParam("lastName")String lastName, @RequestParam("email")String email,
152 @RequestParam("password")String password) {
153
154     int idInt = Integer.parseInt(id);
155     SystemUser user =systemUserServiceImpl.findUserById(idInt);
156     user.setFirstName(firstName);
157     user.setLastName(lastName);
158     user.setEmail(email);
159     user.setPassword(password);
160
161
162     //UPDATE
163     securityConfig.removeUsers();
164     systemUserServiceImpl.save(user);
165     securityConfig.updateUsers();
166
167     return "redirect:/admin/studentList";
168 }
169
170@ @PostMapping("/deleteStudent")
171 public String update(@RequestParam("id")String id) {
172
173     int idInt = Integer.parseInt(id);
174
175     //Delete Students Groups
176     groupServiceImpl.deleteAllInscriptionsByStudentId(idInt);
177     securityConfig.removeUsers();
178     //Delete Student
179     systemUserServiceImpl.deleteById(idInt);
180     securityConfig.updateUsers();
181
182     return "redirect:/admin/studentList";
183 }
184
185
186
187 //
188 //TEACHER
189 //
190
191
192
193@ @GetMapping("/teacherList")
194 public String showTeachersList(Model model) {
195     List<SystemUserWithRoleIdDTO> teacherList = systemUserServiceImpl.findAllTeachersDTO();
196     model.addAttribute("teacherList", teacherList);
197     if (teacherList.isEmpty()) {
198         model.addAttribute("emptyGroupMessage", "No teachers found.");
199     }
200     return "adminTeacherList";
201 }
202
203@ @GetMapping("/createTeacher")
204 public String createTeachers(Model model) {
205     List<SystemUserWithRoleIdDTO> teacherList = systemUserServiceImpl.findAllStudentsDTO();
206     model.addAttribute("teacherList", teacherList);
207     if (teacherList.isEmpty()) {
208         model.addAttribute("emptyGroupMessage", "No teachers found.");
209     }
210     return "adminNewTeacher";
211 }
212
213@ @PostMapping("/createTeacher/save")
214 public String saveNewTeacher(@RequestParam("firstName")String firstName,
215 @RequestParam("lastName")String lastName, @RequestParam("email")String email,
216 @RequestParam("password")String password) {
217     securityConfig.removeUsers();
218     //SAVE
219     SystemUser user =systemUserServiceImpl.save(new SystemUser(0,firstName,lastName,email,passw
220 user.setTeacher(systemUserServiceImpl.saveTeacher(new Teacher(0,user,null)));
221 systemUserServiceImpl.save(user);
222 securityConfig.updateUsers();
223     return "redirect:/admin/teacherList";
224 }
225
226@ @GetMapping("/editTeacher")
227 public String updateTeacher(@RequestParam("id") int id, Model model) {
228     SystemUserWithRoleIdDTO teacher = userSessionServiceImpl.findUserWithRoleIdBySystemId(
229 model.addAttribute("teacher", teacher);
230     return "adminEditTeacher";
231 }
232
233@ @PostMapping("/editTeacher/updateTeacher")
234 public String updateTeacher(@RequestParam("id")String id, @RequestParam("firstName")String first
235 @RequestParam("lastName")String lastName, @RequestParam("email")String email,
236 @RequestParam("password")String password) {
237
238     int idInt = Integer.parseInt(id);
239     SystemUser user =systemUserServiceImpl.findUserById(idInt);
240     user.setFirstName(firstName);
241     user.setLastName(lastName);
242     user.setEmail(email);
243     user.setPassword(password);
244     //UPDATE
245     securityConfig.removeUsers();
246     systemUserServiceImpl.save(user);
247     securityConfig.updateUsers();
248     return "redirect:/admin/teacherList";
249 }
250 }

```


Template(Example)

Login:

```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4   <title>Login</title>
5   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="style
6 </head>
7 <body>
8   <h1>Login</h1>
9   <form th:action="@{/login}" method="post">
10     <div>
11       <label for="username">Username:</label>
12       <input type="text" id="username" name="username" required />
13     </div>
14     <div>
15       <label for="password">Password:</label>
16       <input type="password" id="password" name="password" required />
17     </div>
18     <div>
19       <button type="submit" class="btn btn-success">Login</button>
20     </div>
21     <div th:if="${param.error}">
22       <p>Invalid username or password.</p>
23     </div>
24     <div th:if="${param.logout}">
25       <p>You have been logged out.</p>
26     </div>
27   </form>
28 </body>
29 </html>
```

Explanation

Because this application implements Spring Security, trying to access any address will redirect to the /login page. Users are loaded from the database tbl_SystemUser. Depending on which type of user logs in, the page is redirected to the user's homepage.

 localhost:8080/login

Login

Username:

Password:

Login

All the functions of each user are shown below:

Student functions:

- View groups
- View profile details
- Logout

Welcome Bob Smith

[View profile information](#)

Your groups

Id Inscription	Id Group	Course Name	Description	Teacher Name	Semester	Year	Score
2	1	Introduction to Java	A comprehensive introduction to Java programming language, covering basic concepts and advanced topics.	Hannah Montana	1	2024	80.0
15	3	Database Management Systems	An in-depth course on database design, SQL, and the principles of relational databases.	Hannah Montana	1	2024	90.0

[Logout](#)

User Detalis - Bob Smith

[Return to home page](#)

Name	Value
System User Id	2
First Name	Bob
Last Name	Smith
Email	bob.smith@example.com
Password	password123
Role	STUDENT
Role Id	2

[Logout](#)

Teacher functions:

- View groups
- View profile details
- View group details
- Update Grades
- Logout

Welcome Hannah Montana

View profile information

Your groups

Id Group	Course Name	Description	Semester	Year	View Details
1	Introduction to Java	A comprehensive introduction to Java programming language, covering basic concepts and advanced topics.	1	2024	View details
3	Database Management Systems	An in-depth course on database design, SQL, and the principles of relational databases.	1	2024	View details
7	Web Development with Spring Boot	A practical guide to developing web applications using Spring Boot and related technologies.	1	2024	View details

Logout

Group Details - Introduction to Java

groupId: 1

Description: A comprehensive introduction to Java programming language, covering basic concepts and advanced topics.

Semester: 1

Year: 2024

Return to list Update grades

Students

Id Student	First Name	lastName	Score
2	Bob	Smith	80
3	Edgar	Itzak	1
4	Diana	Prince	0
5	Ethan	Hunt	0
6	Fiona	Shrek	0

Logout

Updating Grades:

Cancel changes and return

Edit student's score

Update Scores

Id Student	First Name	lastName	Score
2	Bob	Smith	<input type="text" value="80"/>
3	Edgar	Itzak	<input type="text" value="100"/>
4	Diana	Prince	<input type="text" value="70"/>
5	Ethan	Hunt	<input type="text" value="50"/>
6	Fiona	Shrek	<input type="text" value="85"/>

Logout

Admin functions:

- View all students
- Create a new student
- Update a student
- Delete student
- View all teachers
- Create a new teacher
- Update a teacher

- View all courses
- View all groups
- View profile details
- Logout

Welcome Tony Stark

View profile information

Options

View all Students View all Teachers View all Groups View all Courses

Logout

User Detalis - Tony Stark

Return to home page

Name	Value
System User Id	11
First Name	Tony
Last Name	Stark
Email	tony.stark@example.com
Password	ironman
Role	ADMIN
Role Id	1

Logout

All students

[Return to home page](#)

[Add a new Student](#)

Student list

System Id	First Name	Last Name	Email	Password	Role	Student Id	Update	Delete
2	Bob	Smith	bob.smith@example.com	password123	STUDENT	2	Edit	Delete
3	Edgar	Itzak	edgaritzak@outlook.com	password123	STUDENT	3	Edit	Delete
4	Diana	Prince	diana.prince@example.com	password123	STUDENT	4	Edit	Delete
5	Ethan	Hunt	ethan.hunt@example.com	password123	STUDENT	5	Edit	Delete
6	Fiona	Shrek	fiona.shrek@example.com	password123	STUDENT	6	Edit	Delete
12	Dan	Vic	danvic@gmail.com	MRDAN	STUDENT	8	Edit	Delete
13	Edgar	Itzak	edgaritzak@gmail.com	pw	STUDENT	9	Edit	Delete

Edit Student

[Return to home page](#)

Edit Student

System Id	First Name	Last Name	Email	Password	Role	Student Id
2	Bob	Smith	bob.smith@example.com	password123	STUDENT	2

[Save changes](#)

[Cancel changes](#)

[Logout](#)

New Student

[Return to home page](#)

New Student

First Name	Last Name	Email	Password
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

[Create new Student](#)

[Logout](#)

Same functions with teachers:

All teachers

[Return to home page](#)

[Add a new Teacher](#)

Teacher list

System Id	First Name	Last Name	Email	Password	Role	Teacher Id	Update
8	Hannah	Montana	hannah.montana@example.com	password1234	TEACHER	1	Edit
9	Ian	Fleming	ian.fleming@example.com	password123	TEACHER	2	Edit
10	Judy	Hopps	judy.hopps@example.com	password123	TEACHER	3	Edit
14	Edgar	Itzak	teacheritzak@outlook.com	PASS	TEACHER	4	Edit
17	I AM THE	TEACHER	EMAIL@ME.COM	PASS1	TEACHER	5	Edit

[Logout](#)

All Groups

[Return to home page](#)

Group list

groupid	courseName	description	semester	year	teacher
1	Introduction to Java	A comprehensive introduction to Java programming language, covering basic concepts and advanced topics.	1	2024	Hannah Montana
2	Introduction to Java	A comprehensive introduction to Java programming language, covering basic concepts and advanced topics.	1	2024	Ian Fleming
3	Database Management Systems	An in-depth course on database design, SQL, and the principles of relational databases.	1	2024	Hannah Montana
4	Database Management Systems	An in-depth course on database design, SQL, and the principles of relational databases.	1	2024	Ian Fleming
5	Web Development with Spring Boot	A practical guide to developing web applications using Spring Boot and related technologies.	1	2024	Ian Fleming
6	Web Development with Spring Boot	A practical guide to developing web applications using Spring Boot and related technologies.	1	2024	Judy Hopps
7	Web Development with Spring Boot	A practical guide to developing web applications using Spring Boot and related technologies.	1	2024	Hannah Montana

All Courses

[Return to home page](#)

Courses

Id	Name	Description
1	Introduction to Java	A comprehensive introduction to Java programming language, covering basic concepts and advanced topics.
2	Database Management Systems	An in-depth course on database design, SQL, and the principles of relational databases.
3	Web Development with Spring Boot	A practical guide to developing web applications using Spring Boot and related technologies.
4	Machine Learning Fundamentals	An introductory course on machine learning concepts, algorithms, and applications.

[Logout](#)

Conclusion:

Spring Security is crucial in a CRUD project because it provides a robust authentication and authorization layer, ensuring that only authorized users can access or modify data. This protects the application from unwanted access and security vulnerabilities, ensuring that CRUD operations are performed securely and in accordance with established permissions.

Using controllers to display web interfaces in a CRUD project enables efficient interaction between users and the database. Controllers handle user requests, orchestrate business logic, and return views that facilitate data visualization and manipulation, making the application more intuitive and accessible to users.