2023

Diploma App

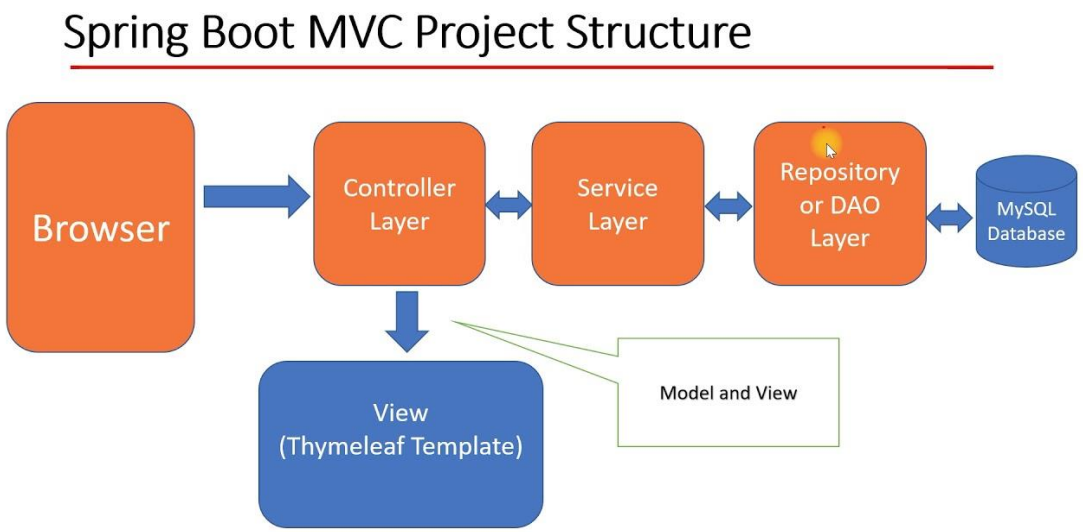
ΑΝΤΩΝΟΓΛΟΥ ΝΑΤΑΛΙΑ, ΑΜ 3387

ΛΕΙΒΑΔΙΤΟΥ ΣΤΥΛΙΑΝΗ-ΠΑΝΤΕΛΙΝΑ, ΑΜ 3268

Περιεχόμενα

Εισαγωγή

Η εφαρμογή, η οποία αναπτύχθηκε με το Java framework Spring Boot, ακολουθεί το πρότυπο MVC (Model-View-Controller) και υλοποιεί την παρακάτω λογική σχεδίαση:



Με βάση το παραπάνω σχεδιάγραμμα έχουν δημιουργηθεί 4 πακέτα (packages):

1. Controllers: Περιέχει τις κλάσεις που υλοποιούν τους Controllers.
2. Entity: Ουσιαστικά περιέχει τα μοντέλα (Models) μας δηλαδή τις οντότητες.
3. Services: Περιέχει τις κλάσεις που υλοποιούν το Business Layer.
4. Dao: Περιέχει τις κλάσεις που υλοποιούν το Data Access Layer.

Επιπρόσθετα, έχει δημιουργηθεί ένα πακέτο config, που περιλαμβάνει μία κλάση που υλοποιεί την αυθεντικοποίηση και το hashing στους κωδικούς.

Τέλος, για το View, χρησιμοποιήθηκε το Thymeleaf με το οποίο δημιουργήθηκαν templates και τα οποία ουσιαστικά αποτελούν το Frontend της εφαρμογής.

Ανάλυση Σχεδίασης και Υλοποίησης Εφαρμογής

1. Σχεδίαση Βάσης Δεδομένων

Στο πρώτο στάδιο του σχεδιασμού, με βάση οι λειτουργικές απαιτήσεις τις εφαρμογής, καταγράφηκαν οι οντότητες και τα χαρακτηριστικά αυτών. Αναλυτικά:

α. Οντότητα Χρήστης (Users):

Χαρακτηριστικά: id, user_name, password, full_name, enabled, avg_grade, remain_courses, years, role, speciality

β. Οντότητα Διπλωματική (Diploma):

Χαρακτηριστικά: diploma_id, title, objectives, user, user_id, available

γ. Οντότητα Αιτήσεις (Applications):

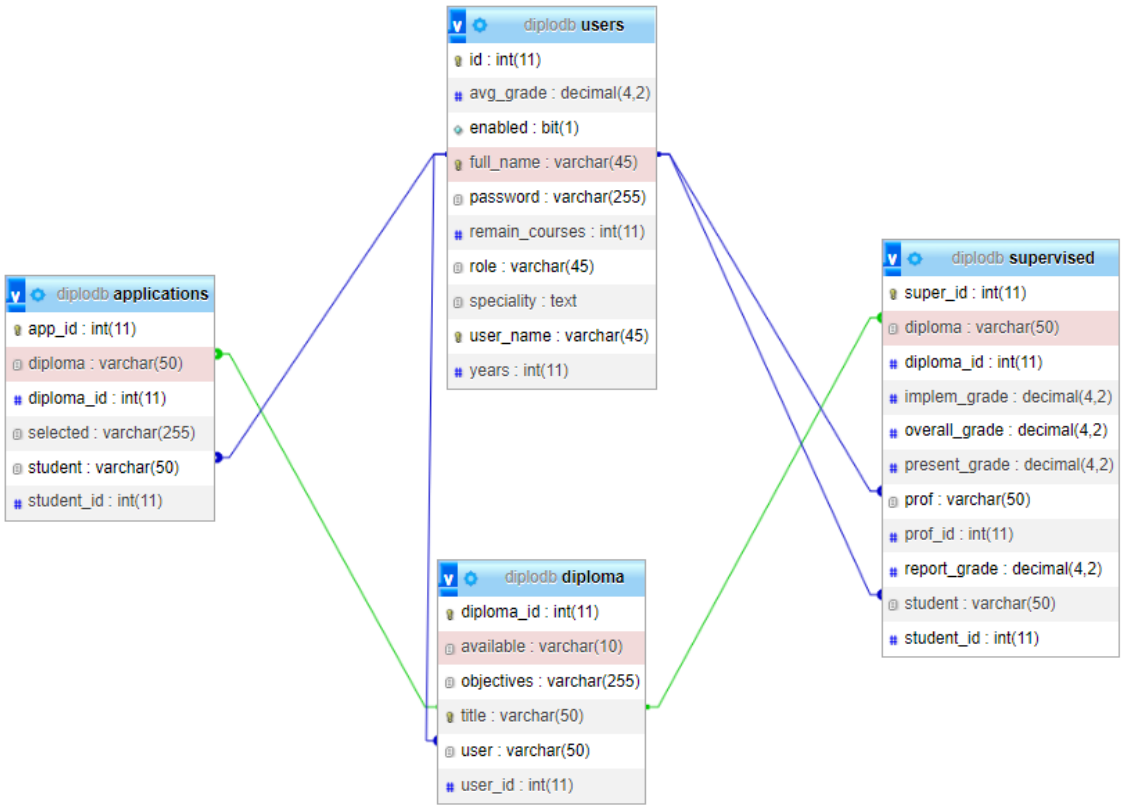
Χαρακτηριστικά: app_id, diploma, diploma_id, student, student_id, selected

δ. Οντότητα Επιβλέψεις (Supervised):

Χαρακτηριστικά: super_id, diploma, diploma_id, implem_grade, overall_grade, present_grade, prof, prof_id, report_grade, student, student _id

Ως βάση δεδομένων χρησιμοποιήθηκε η MySQL και ως Σύστημα Διαχείρισης Βάσεων Δεδομένων το phpMyAdmin.

Σχεδίαση:



Με μπλε και πράσινες γραμμές αναλύονται τα ξένα κλειδιά (foreign keys). Τα ξένα κλειδιά έχουν δημιουργηθεί μέσω του phpMyAdmin και όχι μέσω του hibernate.

2. Πακέτο entity

Στο πακέτο entity υπάρχουν 4 κλάσεις, 1 για κάθε οντότητα.

α. Κλάση User

Στην κλάση User υλοποιούμε την οντότητα του χρήστη. Πρέπει να τονίσουμε ότι δεν δημιουργήσαμε κατασκευαστή (constructor) γιατί δεν απαιτείται. Ο κώδικας έχει όπως παρακάτω:

```
package com.diplomatics.entity;

import javax.persistence.*;
import java.math.BigDecimal;

@Entity
@Table(name = "users") →Με το όρο Table αντιστοιχούμε την κλάση με τον πίνακα που θέλουμε στη ΒΔ
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY) → Από εδώ και κάτω γίνεται ο καθορισμός
    private Integer id;                                    των χαρακτηριστικών

    @Column(precision = 4, scale = 2 )
    private BigDecimal avg_grade;

    @Column(nullable = false, unique = true, length = 50)
    private String userName;

    @Column(length = 255, nullable = false)
    private String password;

    @Column(length = 50, nullable = false)
    private String full_name;

    @Column(length = 10, nullable = false)
    private String role;

    @Column
    private Integer years;

    @Column
    private Integer remain_courses;

    @Column
    private String specialist;

    private boolean enabled;


    public Integer getId() { →Από εδώ και κάτω δημιουργούμε τους Getters και Setters
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public BigDecimal getAvg_grade() { return avg_grade; }

    public void setAvg_grade(BigDecimal avg_grade) {
        this.avg_grade = avg_grade;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
```

```
    public String getFull_name() {
        return full_name;
    }

    public void setFull_name(String full_name) {
        this.full_name = full_name;
    }

    public String getRole() {
        return role;
    }

    public void setRole(String role) {
        this.role = role;
    }

    public Integer getYears() {
        return years;
    }

    public void setYears(Integer years) {
        this.years = years;
    }

    public Integer getRemain_courses() {
        return remain_courses;
    }

    public void setRemain_courses(Integer remain_courses) {
        this.remain_courses = remain_courses;
    }

    public String getSpeciality() {
        return speciality;
    }

    public void setSpeciality(String speciality) {
        this.speciality = speciality;
    }

    @Override
    public String toString() {
        return full_name;
    }

    public boolean isEnabled() {
        return enabled;
    }

    public void setEnabled(boolean enabled) {
        this.enabled = enabled;
    }
    public Boolean getEnabled() { return enabled; }
}
```

β. Κλάση Diploma

Η ανάλυση είναι όμοια με την κλάση User, αφού και αυτή χρησιμοποιείται για την υλοποίηση μιας οντότητας, αυτή της Διπλωματικής (diploma).

γ. Κλάση Applications

Η ανάλυση είναι όμοια με την κλάση User, αφού και αυτή χρησιμοποιείται για την υλοποίηση μιας οντότητας, αυτή των Αιτήσεις (Applications).

δ. Κλάση Supervised

Η ανάλυση είναι όμοια με την κλάση User, αφού και αυτή χρησιμοποιείται για την υλοποίηση μιας οντότητας, αυτή των Αιτήσεις (Applications). Η μόνη διαφορά έχει να κάνει στο ότι σε αυτή την κλάση απαιτείται κατασκευαστής

```
public Supervised(Integer diploma_id, String diploma, Integer student_id, String student, Integer
prof_id, String prof) {
    this.diploma_id = diploma_id;
    this.diploma = diploma;
    this.student_id = student_id;
    this.student = student;
    this.prof_id = prof_id;
    this.prof = prof;
}
```

3. Πακέτο controllers

Στο πακέτο entity υπάρχουν 4 κλάσεις και κάθε μια από αυτές αφορούν μία οντότητα.

α. Κλάση UserController

```
package com.diplomatics.controllers;

import com.diplomatics.entity.User;
import com.diplomatics.services.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.web.authentication.logout.SecurityContextLogoutHandler;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.Objects;

@Controller →Με αυτό το annotation καθορίζουμε ότι αυτή η κλάση είναι controller
public class UserController {

    @Autowired  →Με το annotation @Autowired «ενώνουμε» κλάσεις που είναι απαραίτητες
    private BCryptPasswordEncoder passwordEncoder=new BCryptPasswordEncoder();

    @Autowired
    private UserService service=new UserService();

    @Autowired
    private UserService userService;

    Η παρακάτω μέθοδος εμφανίζει το προφίλ του χρήστη
    @GetMapping("/myprofile") →Με το annotation @GetMapping ουσιαστικά αντιστοιχούμε
    public String showUserList(Model model) {     το url με την μέθοδο
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        User listUsers=userService.findExistUser(auth.getName());
        model.addAttribute("listUsers", listUsers);

        return "myprofile"; →Σε αυτό το σημείο λέμε πιο template θέλουμε να εμφανίσει
    }                        στον χρήστη.

Δηλαδή όταν ένας χρήστης πληκτρολογήσει στο url (μετά την αυθεντικοποίησή) του π.χ.
http://localhost:8080/myprofile θα γίνει map το /myprofile και θα τρέξει η μέθοδος και στο τέλος θα του
εμφανίσει το template με το όνομα myprofile που βρίσκεται στα resources/templates

    Η παρακάτω μέθοδος εμφανίζει την φόρμα εγγραφής για νέο χρήστη
    @GetMapping("/register")
    public String showNewForm(Model model) {
        model.addAttribute("user", new User());
        model.addAttribute("pageTitle", "Εγγραφή Χρήστη");
        return "user_form";
    }

    Η παρακάτω μέθοδος αποσυνδέει με ασφάλεια τον χρήστη
    @GetMapping("/logout")
    public String logoutPage(HttpServletRequest request, HttpServletResponse response) {
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
```

```
        if (auth != null){
            new SecurityContextLogoutHandler().logout(request, response, auth);
        }
        return "redirect:/";
    }

    Η παρακάτω μέθοδος αποθηκεύει τον νέο χρήστη και την επεξεργασία του προφίλ ενός υπάρχοντος χρήστη
    @PostMapping("/users/save")
    public String saveUser(User user) {

        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        String name = auth.getName();

        Αν ο χρήστης έχει username "anonymousUser" πρόκειται για νέο χρήστη
        if (name!="anonymousUser"){
            User activeUser=userService.findExistUser(auth.getName());

            if(user.getPassword().isEmpty()){
                user.setPassword(activeUser.getPassword());
            }else{
                user.setPassword(passwordEncoder.encode(user.getPassword()));
            }

            if(!Objects.equals(activeUser.getUserName(), user.getUserName())){
                service.save(user);
                return "redirect:/login";
            }

            if(user.getEnabled()){
                service.save(user);
                return "redirect:/myprofile";
            }
        }

        Εδω γίνεται hashing του κωδικού για ασφάλεια
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        user.setEnabled(true);
        service.save(user);
        return "redirect:/login";
    }

    Η παρακάτω μέθοδος εμφανίζει την φόρμα για την επεξεργασία του προφίλ ενός υπάρχοντος χρήστη
    @GetMapping("/users/edit/{id}")
    public String showEditForm(@PathVariable("id") Integer id, Model model, RedirectAttributes ra) {

        User user = service.get(id);
        model.addAttribute("user", user);
        model.addAttribute("pageTitle", "Επεξεργασία Προφίλ");

        return "user_form_2";
    }

    Η παρακάτω μέθοδος χρησιμοποιείται κατά την εγγραφή και επεξεργασία του προφίλ για τον έλεγχο του
username
@PostMapping("/users/exist/{uname}")
public @ResponseBody String checkUser(@PathVariable("uname") String uname) {

    User user=userService.findExistUser(uname);

    if(user==null){
        return "Οχι";
    }else{
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        String name = auth.getName();
        if (name!="anonymousUser"){
            User activeUser=userService.findExistUser(auth.getName());
            if(activeUser.getUserName()==user.getUserName()){
                return "Οχι";
            }
        }
        return "Ναι";
    }
}
```

```
    Η παρακάτω μέθοδος χρησιμοποιείται κατά την εγγραφή και επεξεργασία του προφίλ για τον έλεγχο του
ονοματεπώνυμου του χρήστη
@PostMapping("/users/existFullname/{fname}")
public @ResponseBody String checkUserFullname(@PathVariable("fname") String fname) {

    User user=userService.findExistUserFullname(fname);
    if(user==null){
        return "Οχι";
    }else{

        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        String name = auth.getName();

        if (name!="anonymousUser"){
            User activeUser=userService.findExistUser(auth.getName());
            if(activeUser.getUserName()==user.getUserName()){
                return "Οχι";
            }
        }
        return "Ναι";
    }

}
```

β. Κλάση DiplomaController

```
package com.diplomatics.controllers;

import com.diplomatics.entity.Applications;
import com.diplomatics.entity.User;
import com.diplomatics.services.ApplicationsService;
import com.diplomatics.services.DiplomaService;
import com.diplomatics.entity.Diploma;
import com.diplomatics.services.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import java.util.List;

@Controller
public class DiplomaController {

    @Autowired
    private DiplomaService diplomaService=new DiplomaService();

    @Autowired
    private ApplicationsService applicationsService=new ApplicationsService();

    @Autowired
    private UserService userService=new UserService();

    Η παρακάτω μέθοδος εμφανίζει την φόρμα για την Καταχώρηση Νέας Διπλωματικής
    @GetMapping("/add-diploma")
    public String addDiploma(Model model){
        model.addAttribute("diploma", new Diploma());
        model.addAttribute("pageTitle", "Καταχώρηση Νέας Διπλωματικής");
        return "diploma_form";
    }

    Η παρακάτω μέθοδος εμφανίζει όλες τις διπλωματικές που έχει καταχωρήσει ένας Καθηγητής ενώ στην
περίπτωση του Μαθητή όλες τις διπλωματικές που είναι διαθέσιμες για
να κάνει apply. Στην περίπτωση που έχει κάνει apply αλλά δεν επιλέχτηκε εμφανίζεται κατάλληλο μήνυμα.
    @GetMapping("/diploma")
    public String showDiplomaList(Model model) {
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        User user=userService.findExistUser(auth.getName());
        String role=user.getRole();
        List<Diploma> listDiplomas;

        String flag="";
```

```
        if(role.equals("STUD")) {
            Applications app=applicationsService.findUsersApp(user.getId());
            if (app!=null) {
                if (app.getSelected().equals("Επιλεχθήκατε")) {
                    flag="no";
                } else {
                    flag="yes";
                }
            }else{
                flag="yes";
            }
            listDiplomas = diplomaService.findDiplomasAvailable();
        }else{
            listDiplomas = diplomaService.findDiplomas(user.getId());
        }
        model.addAttribute("flag", flag);
        model.addAttribute("listDiplomas", listDiplomas);
        model.addAttribute("role", role);

        return "diploma";
    }

    Η παρακάτω μέθοδος αποθηκεύει την διπλωματική εργασία (είτε νέα είτε παλιά που επεξεργάστηκε ο
καθηγητής)
    @PostMapping("/diploma/save")
    public String saveDiploma(Diploma diploma, RedirectAttributes ra){

        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        User user=userService.findExistUser(auth.getName());
        if(user!=null){
            diploma.setUser(user.getFull_name());
            diploma.setUser id(user.getId());
            diploma.setAvailable("Ναι");
            ra.addFlashAttribute("message", "Η Διπλωματική καταχωρήθηκε επιτυχώς");
        }
        diplomaService.save(diploma);
        return "redirect:/diploma";
    }

    Η παρακάτω μέθοδος είναι προσβάσιμη μόνο από τον μαθητή και εμφανίζει όλες τις πληροφορίες για μια
συγκεκριμένη  διπλωματική εργασία
    @GetMapping("/diploma/info/{id}")
    public String diplomaInfo(@PathVariable("id")Integer id, Model model)  {

        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        User user=userService.findExistUser(auth.getName());
        String role=user.getRole();

        Diploma diploma=diplomaService.getDiploma(id);
        model.addAttribute("diploma", diploma);
        model.addAttribute("role", role);
        model.addAttribute("info", "Λεπτομέρειες Διπλωματικής Εργασίας " + diploma.getTitle() );
        return "diploma_info";

    }

    Η παρακάτω μέθοδος είναι προσβάσιμη μόνο από τον καθηγητή και εμφανίζει την φόρμα για την
επεξεργασία μιας διπλωματικής εργασίας
    @GetMapping("/diploma/edit/{id}")
    public String diplomaEdit(@PathVariable("id") Integer id, Model model){
        Diploma diploma=diplomaService.getDiploma(id);
        model.addAttribute("diploma", diploma);
        model.addAttribute("pageTitle", "Επεξεργασία Διπλωματικής");
        return "diploma_form";
    }

    Η παρακάτω μέθοδος είναι προσβάσιμη μόνο από τον καθηγητή και χρησιμοποιείται για την διαγραφή μιας
διπλωματικής εργασίας αν αυτή δεν έχει ανατεθεί ακόμα
    @GetMapping("/diploma/delete/{id}")
    public String diplomaDelete(@PathVariable("id") Integer id, RedirectAttributes ra){
        diplomaService.delete(id);
        List<Applications> apps=applicationsService.findByDiploma(id);
        if(apps!=null) {
            if (!apps.isEmpty()) {
                for (int i = 0; i < apps.size(); i++) {
                    applicationsService.delete(apps.get(i));
                }
```

```
            }
            ra.addFlashAttribute("message", "Η Διπλωματική διαγράφηκε επιτυχώς");
        }

        return "redirect:/diploma";
    }

    Η παρακάτω μέθοδος χρησιμοποείται κατά την καταχώρηση ή επεξεργασία μιας διπλωματικής για να
ελέγχει την ύπαρξη άλλης διπλωματικής με τον ίδιο τίτλο. Αν υπάρχει εμφανίζει κατάλληλο μήνυμα
    @PostMapping("/diploma/existDiploma")
    @ResponseBody
    public String checkDiploma(@RequestBody Diploma diploma) {

        String title = diploma.getTitle();

        Diploma diplomaExist=diplomaService.findDiploma(title);

        if(diplomaExist==null){
            return "{\"message\":\"" + "Οχι" + "\"}";
        }else{
            if(diploma.getDiploma_id()==diplomaExist.getDiploma_id()){
                return "{\"message\":\"" + "Οχι" + "\"}";
            }
            return "{\"message\":\"" + "Ναι" + "\"}";
        }
    }
}
```

γ. Κλάση ApplicationsController

```
package com.diplomatics.controllers;

import com.diplomatics.entity.Applications;
import com.diplomatics.entity.Diploma;
import com.diplomatics.entity.Supervised;
import com.diplomatics.entity.User;

import com.diplomatics.services.ApplicationsService;

import com.diplomatics.services.DiplomaService;
import com.diplomatics.services.SupervisedService;
import com.diplomatics.services.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import java.math.BigDecimal;
import java.util.List;
import java.util.Random;

@Controller
public class ApplicationsController {

    @Autowired
    private UserService userService=new UserService();

    @Autowired
    private DiplomaService diplomaService = new DiplomaService();

    @Autowired
    private ApplicationsService applicationsService = new ApplicationsService();

    @Autowired
    private SupervisedService supervisedService = new SupervisedService();

        Η παρακάτω μέθοδος εμφανίζει όλες τις Αιτήσεις που έχει κάνει ο Μαθητής ενώ την περίπτωση του
Καθηγητή όλες τις διπλωματικές του που έχουν αιτήσεις.
    @GetMapping ("/applications")
```

```
    public String showApplications(Model model){

        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        User user=userService.findExistUser(auth.getName());
        String role=user.getRole();

        List<Applications> listApplications;

        if(role.equals("STUD")) {
            listApplications = applicationsService.getApplication(user.getId());
        }else{
            listApplications = applicationsService.getApplicationStud(user.getFull_name());
        }
        model.addAttribute("listApplications", listApplications);
        model.addAttribute("role", role);

        return "applications";
    }

    Η παρακάτω μέθοδος χρησιμοποιείται για να δει αναλυτικά ο Καθηγητής τις αιτήσεις των Μαθητών για
τις Διπλωματικές του.
    @GetMapping("/applications/info/{id}")
    public String showAppInfo(@PathVariable("id")Integer id, Model model){
        Diploma diploma=diplomaService.getDiploma(id);
        List<User> listUsers;
        if(diploma.getAvailable().equals("Οχι")){
            listUsers = userService.getStudentsReady(id);
        }else {
            listUsers = userService.getAppStudents(id);
        }
        model.addAttribute("listUsers", listUsers);
        model.addAttribute("diploma", diploma);
        model.addAttribute("id", id);
        return "applications_info";
    }

    Η παρακάτω μέθοδος χρησιμοποιείται για να κάνει αίτηση ο Μαθητής σε μια Διπλωματική. Σε περίπτωση
που έχει ξανά υποβάλλει αίτηση για την συγκεκριμένη Διπλωματική εμφανίζεται κατάλληλο μήνυμα ενημέρωσης
    @GetMapping("/applications/apply/{id}")
    public String applyApp(@PathVariable("id")Integer id, RedirectAttributes ra) {

        Diploma diploma = diplomaService.getDiploma(id);
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        User user=userService.findExistUser(auth.getName());
        Integer uId=user.getId();
        Applications appExists=applicationsService.getApplicationAgain(id,uId);
        if (appExists==null) {
            Applications app = new Applications();
            app.setDiploma(diploma.getTitle());
            app.setDiploma_id(diploma.getDiploma_id());
            app.setStudent(user.getFull_name());
            app.setStudent_id((user.getId()));
            app.setSelected("Σε Αναμονή");
            applicationsService.save(app);
            ra.addFlashAttribute("message", "Η αίτησή σας πραγματοποιήθηκε.");
            return "redirect:/diploma";
        }else{
            ra.addFlashAttribute("message", "Έχετε υποβάλει ήδη αίτηση για την υπόψη διπλωματική
εργασία.");
            return "redirect:/diploma";
        }
    }

    Η παρακάτω μέθοδος χρησιμοποιείται για να επιλεχτεί τυχαία ένας Μαθητής από αυτούς που υπέβαλλαν
αίτηση για μια συγκεκριμένη Διπλωματική
   @GetMapping("/applications/random/{id}")
   public String randomApp(@PathVariable("id")Integer id){

        Diploma diploma = diplomaService.getDiploma(id);
        List<User> listUsers;
        listUsers = userService.getAppStudents(id);

        Random rand = new Random();
        User randomUser = listUsers.get(rand.nextInt(listUsers.size()));
        Applications app=applicationsService.applyFinalUser(randomUser.getId(), id);
        Supervised supervised = new Supervised(app.getDiploma_id(),
                app.getDiploma(), randomUser.getId(), randomUser.getFull_name(), diploma.getUser_id(),
```

```
diploma.getUser());

        supervisedService.save(supervised);

        diploma.setAvailable("Οχι");
        diplomaService.save(diploma);

        List<Applications> users = applicationsService.applyUsers(randomUser.getId(), id);
        if (!users.isEmpty()) {
           for (int i = 0; i < users.size(); i++) {
               users.get(i).setSelected("Δεν επιλεχθήκατε");
               applicationsService.save(users.get(i));
           }
        }
        // για τον επιλεγέντα μαθητή κάνω όλα τα applications "Δεν επιλεχθήκατε"
        List<Applications> applications = applicationsService.dontApplyUser(randomUser.getId());
        if (!applications.isEmpty()) {
           for (int i = 0; i < applications.size(); i++) {
               applications.get(i).setSelected("Δεν επιλεχθήκατε");
               applicationsService.save(applications.get(i));
           }
        }

       // για τον επιλεγέντα μαθητή κάνω όλα το επιλεχθεν application "Επιλεχθήκατε"
        app.setSelected("Επιλεχθήκατε");
        applicationsService.save(app);
        return "redirect:/supervised";

   }

    Η παρακάτω μέθοδος χρησιμοποιείται για να επιλεχτεί ο Μαθητής με τον μεγαλύτερο βαθμό από αυτούς
που υπέβαλλαν αίτηση για μια συγκεκριμένη Διπλωματική. Σε περίπτωση ίδιου βαθμού επιλέγεται ο πρώτος
Μαθητής της λίστας από την απάντηση της Βάσης Δεδομένων
    @GetMapping("/applications/bestgrade/{id}")
    public String bestGradeApp(@PathVariable("id")Integer id){

        Diploma diploma = diplomaService.getDiploma(id);
        List<User> listUsers;
        listUsers = userService.getAppStudents(id);
        int flag=0;
        double grade= 0;
        for(int i=0; i<listUsers.size(); i++){
            BigDecimal stu_grade=listUsers.get(i).getAvg_grade();
            if (stu_grade.doubleValue()>grade){
                grade=stu_grade.doubleValue();
                flag=i;
            }
        }

        Applications app=applicationsService.applyFinalUser(listUsers.get(flag).getId(), id);

        Supervised supervised = new Supervised(app.getDiploma_id(),
                app.getDiploma(), listUsers.get(flag).getId(), listUsers.get(flag).getFull_name(),
diploma.getUser_id(), diploma.getUser());

        supervisedService.save(supervised);

        diploma.setAvailable("Οχι");
        diplomaService.save(diploma);

        List<Applications> users = applicationsService.applyUsers(listUsers.get(flag).getId(), id);
        if (!users.isEmpty()) {
            for (int i = 0; i < users.size(); i++) {
                users.get(i).setSelected("Δεν επιλεχθήκατε");
                applicationsService.save(users.get(i));
            }
        }
        // για τον επιλεγέντα μαθητή κάνω όλα τα applications "Δεν επιλεχθήκατε"
        List<Applications> applications =
applicationsService.dontApplyUser(listUsers.get(flag).getId());
        if (!applications.isEmpty()) {
            for (int i = 0; i < applications.size(); i++) {
                applications.get(i).setSelected("Δεν επιλεχθήκατε");
                applicationsService.save(applications.get(i));
            }
        }
        // για τον επιλεγέντα μαθητή κάνω όλα το επιλεχθεν application "Επιλεχθήκατε"
```

```
        app.setSelected("Επιλεχθήκατε");
        applicationsService.save(app);
        int super_id=supervised.getSuper_id();
        return "redirect:/supervised";
    }

    Η παρακάτω μέθοδος χρησιμοποιείται για να επιλεχτεί ο Μαθητής με τα λιγότερα  μαθήματα από αυτούς
που υπέβαλλαν αίτηση για μια συγκεκριμένη Διπλωματική. Σε περίπτωση ίδιων μαθημάτων επιλέγεται ο πρώτος
Μαθητής της λίστας από την απάντηση της Βάσης Δεδομένων
    @GetMapping("/applications/fewcourses/{id}")
    public String fewCoursesApp(@PathVariable("id")Integer id){

        Diploma diploma = diplomaService.getDiploma(id);
        List<User> listUsers;
        listUsers = userService.getAppStudents(id);
        int flag=0;
        int courses= 100;
        for(int i=0; i<listUsers.size(); i++){
            int coursesUser=listUsers.get(i).getRemain_courses();
            if (coursesUser<courses){
                courses=coursesUser;
                flag=i;
            }
        }

        Applications app=applicationsService.applyFinalUser(listUsers.get(flag).getId(), id);

        Supervised supervised = new Supervised(app.getDiploma_id(),
                app.getDiploma(), listUsers.get(flag).getId(), listUsers.get(flag).getFull_name(),
diploma.getUser_id(), diploma.getUser());

        supervisedService.save(supervised);

        diploma.setAvailable("Οχι");
        diplomaService.save(diploma);

        List<Applications> users = applicationsService.applyUsers(listUsers.get(flag).getId(), id);
        if (!users.isEmpty()) {
            for (int i = 0; i < users.size(); i++) {
                users.get(i).setSelected("Δεν επιλεχθήκατε");
                applicationsService.save(users.get(i));
            }
        }
        // για τον επιλεγέντα μαθητή κάνω όλα τα applications "Δεν επιλεχθήκατε"
      List<Applications> applications = applicationsService.dontApplyUser(listUsers.get(flag).getId());
        if (!applications.isEmpty()) {
            for (int i = 0; i < applications.size(); i++) {
                applications.get(i).setSelected("Δεν επιλεχθήκατε");
                applicationsService.save(applications.get(i));
            }
        }

        // για τον επιλεγέντα μαθητή κάνω όλα το επιλεχθεν application "Επιλεχθήκατε"
        app.setSelected("Επιλεχτήκατε");
        applicationsService.save(app);

        return "redirect:/supervised";
    }

    Η παρακάτω μέθοδος χρησιμοποιείται για να επιλεχτεί ο Μαθητής σύμφωνα με τον βαθμό και τα μαθήματα
που καθορίζει ο Καθηγητής. Σε περίπτωση που υπάρχουν παραπάνω από ένας μαθητής, επιλέγεται ο πρώτος
Μαθητής της λίστας από την απάντηση της Βάσης Δεδομένων
    @PostMapping("/applications/set")
    public String setApp(@RequestParam int id, @RequestParam BigDecimal grade, @RequestParam Integer
courses ){

        Diploma diploma = diplomaService.getDiploma(id);
        List<User> listUsers;
        listUsers = userService.getAppStudentsSets(id, grade, courses);

        if (listUsers.isEmpty()){
            return "redirect:/applications/info/"+id;
        }
        Random rand = new Random();
        User randomUser = listUsers.get(rand.nextInt(listUsers.size()));
        Applications app=applicationsService.applyFinalUser(randomUser.getId(), id);
        Supervised supervised = new Supervised(app.getDiploma_id(),
```

```
                app.getDiploma(), randomUser.getId(), randomUser.getFull_name(), diploma.getUser_id(),
diploma.getUser());

        supervisedService.save(supervised);

        diploma.setAvailable("Οχι");
        diplomaService.save(diploma);

        List<Applications> users = applicationsService.applyUsers(randomUser.getId(), id);
        if (!users.isEmpty()) {
            for (int i = 0; i < users.size(); i++) {
                users.get(i).setSelected("Δεν επιλεχθήκατε");
                applicationsService.save(users.get(i));
            }
        }
        // για τον επιλεγέντα μαθητή κάνω όλα τα applications "Δεν επιλεχθήκατε"
        List<Applications> applications = applicationsService.dontApplyUser(randomUser.getId());
        if (!applications.isEmpty()) {
            for (int i = 0; i < applications.size(); i++) {
                applications.get(i).setSelected("Δεν επιλεχθήκατε");
                applicationsService.save(applications.get(i));
            }
        }
        // για τον επιλεγέντα μαθητή κάνω όλα το επιλεχθεν application "Επιλεχθήκατε"
        app.setSelected("Επιλεχθήκατε");
        applicationsService.save(app);
        return "redirect:/supervised";
    }
}
```

δ. Κλάση SupervisedController

```
package com.diplomatics.controllers;

import com.diplomatics.entity.Supervised;
import com.diplomatics.entity.User;
import com.diplomatics.dao.UserDao;
import com.diplomatics.services.SupervisedService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import java.util.List;

@Controller
public class SupervisedController {

    @Autowired
    private SupervisedService supervisedService=new SupervisedService();

    @Autowired
    private UserDao userRepository;

    Η παρακάτω μέθοδος χρησιμοποιείται για την εμφάνιση των αναθετημένων διπλωματικών. Στην περίπτωση
του Μαθητή, αυτός βλέπει την διπλωματική του και την βαθμολογία που έχει σε αυτήν. Στην περίπτωση του
Καθηγητή, αυτός βλέπει όλες τις Διπλωματικές που επιβλέπει.
    @GetMapping("/supervised")
    public String showSupervised(Model model){
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        List<User> u=userRepository.findById(auth.getName());
        User user=u.get(0);
        String role=user.getRole();
        List<Supervised> listSupervised;

        if(role.equals("STUD")) {
            listSupervised=supervisedService.findSupervised(user.getId());
        }else{
            listSupervised=supervisedService.findSupervisedStu(user.getId());
        }

        model.addAttribute("listSupervised", listSupervised);
        model.addAttribute("role", role);
```

```
        return "supervised";
    }

    Η παρακάτω μέθοδος χρησιμοποιείται για την εμφάνιση της φόρμας μέσω της οποίος ο Καθηγητής
καταχωρεί τους βαθμούς για την συγκεκριμένη Διπλωματική.   @GetMapping("/supervised/edit/{id}")
    public String showEditForm(@PathVariable("id") Integer id, Model model) {

        Supervised supervised= supervisedService.findSupervisedById(id);
        model.addAttribute("supervised", supervised);
        model.addAttribute("pageTitle", "Βαθμολόγηση Διπλωματικής : " + supervised.getDiploma());

        return "supervised_form";
    }

    Η παρακάτω μέθοδος χρησιμοποιείται για την αποθήκευση μιας νέας Διπλωματικής ή για την αποθήκευση
της βαθμολογίας που καταχωρεί ο καθηγητής για μια Διπλωματική.
    @PostMapping("/supervised/save")
    public String saveForm(Supervised supervised, RedirectAttributes ra) {
        supervisedService.save(supervised);
        if(ra!=null) {
            ra.addAttribute("message", "Επιτυχής βαθμολόγηση");
        }
        return "redirect:/supervised";
    }

}
```

4. Πακέτο services

α. Κλάση UserService

```
package com.diplomatics.services;

import com.diplomatics.dao.UserDao;
import com.diplomatics.entity.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.math.BigDecimal;
import java.util.List;
import java.util.Optional;

@Service
public class UserService {

    Με αυτό το «@Autowired private UserDao repo» κάνω την «σύνδεση» με το Dao του User. Λόγω του ότι το
UserDao είναι interface πρέπει να κάνω implement και override της μεθόδους του UserDao
    @Autowired private UserDao repo=new UserDao() {
        @Override
        public Optional<User> findByUserName(String username) { return Optional.empty(); }

        @Override
        public List<User> findById(String username) { return null; }

        @Override
        public List<User> findAppStudents(int app_id) { return null;}

        @Override
        public List<User> getStudentsReady(int app_id) {return null; }

        @Override
        public List<User> getAppStudentsSets(int app_id, BigDecimal grade, int courses){return null;}

        @Override
        public User findExistUser(String user_name) { return null;}

        @Override
        public User findExistUserFullname(String full_name) { return null;}

        @Override
        public <S extends User> S save(S s) { return null; }

        @Override
        public <S extends User> Iterable<S> saveAll(Iterable<S> iterable) { return null; }
```

```
        @Override
        public Optional<User> findById(Integer integer) {return Optional.empty();}

        @Override
        public boolean existsById(Integer integer) {return false; }

        @Override
        public Iterable<User> findAll() { return null;}

        @Override
        public Iterable<User> findAllById(Iterable<Integer> iterable) { return null;}

        @Override
        public long count() { return 0;}

        @Override
        public void deleteById(Integer integer) { }

        @Override
        public void delete(User user) { }

        @Override
        public void deleteAllById(Iterable<? extends Integer> iterable) { }

        @Override
        public void deleteAll(Iterable<? extends User> iterable) {}

        @Override
        public void deleteAll() {}
    };

    Από εδώ και κάτω είναι τα services που χρησιμοποιώ για να καλέσω τις μεθόδους του UserDao. Αυτό το
κάνω γιατί από τις αρχές σχεδίασης και ανάπτυξης καθώς και από την προτεινόμενη χρήση της Spring Boot
τα ερωτήματα στην βάση γίνονται από το Repository/Dao της εφαρμογής. Το Service Layer καλεί αυτές τις
μεθόδους και προωθεί τα αποτελέσματα στο Controller layer.
    public void save(User user) { repo.save(user);  }

    public User get(Integer id) { Optional<User> result = repo.findById(id);  return result.get();  }

    public  List<User> getAppStudents(Integer id) { return repo.findAppStudents(id); }

    public  List<User> getAppStudentsSets(Integer id, BigDecimal grade, Integer courses) { return
repo.getAppStudentsSets(id, grade, courses); }

    public  List<User> getStudentsReady(Integer id) { return repo.getStudentsReady(id); }

    public User findExistUser(String username) { return repo.findExistUser(username);}

    public User findExistUserFullname(String full_name){return repo.findExistUserFullname(full_name);}
}
```

β. Κλάση MyUserDetailsService

Αυτή η κλάση υλοποιεί (implements) την κλάση UserDetailsService της Spring Boot και χρησιμοποιείται για την αυθεντικοποίηση του χρήστη.

```
package com.diplomatics.services;


import com.diplomatics.dao.UserDao;
import com.diplomatics.entity.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class MyUserDetailsService implements UserDetailsService {

    @Autowired
```

```
    UserDao userRepository;

    @Override
    public UserDetails loadUserByUsername(String userName) throws UsernameNotFoundException {
        Optional<User> user = userRepository.findByUserName(userName);

        user.orElseThrow(() -> new UsernameNotFoundException("Δεν βρέθηκε: " + userName));

        return user.map(MyUserDetails::new).get();
    }


}
```

γ. Κλάση MyUserDetails

Αυτή η κλάση υλοποιεί (implements) την κλάση UserDetails της Spring Boot και χρησιμοποιείται, όπως και η MyUserDetailsService παραπάνω, για την αυθεντικοποίηση του χρήστη

```
package com.diplomatics.services;

import com.diplomatics.entity.User;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import java.util.Collection;
import java.util.Collections;

public class MyUserDetails implements UserDetails {

    private String userName;
    private String password;

    private String full_name;

    private boolean active;

    private User user;
    public MyUserDetails(User user) {
        this.userName = user.getUserName();
        this.password = user.getPassword();
        this.active = user.isEnabled();
        this.user=user;
        this.full_name=user.getFull_name();
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {

        return Collections.singleton(new SimpleGrantedAuthority(user.getRole()));
    }

    public String getFull_Name() {
        return full_name;
    }
    @Override
    public String getPassword() {
        return password;
    }

    @Override
    public String getUsername() { return userName; }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }
```

```
    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return active;
    }

}
```

δ. Κλάση DiplomaService

```
package com.diplomatics.services;

import com.diplomatics.dao.DiplomaDao;
import com.diplomatics.entity.Diploma;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class DiplomaService {

    Με αυτό το «@Autowired private DiplomaDao repo» κάνω την «σύνδεση» με το Dao του Diploma. Λόγω του
ότι το DiplomaDao είναι interface πρέπει να κάνω implement και override της μεθόδους του DiplomaDao
    @Autowired
    private DiplomaDao repo=new DiplomaDao() {
        @Override
        public List<Diploma> findDiplomas(int user_id) { return null; }

        @Override
        public List<Diploma> findDiplomasAvailable() { return null; }

        @Override
        public Diploma findDiploma(String title) {return null;}

        @Override
        public <S extends Diploma> S save(S s) { return null;}

        @Override
        public <S extends Diploma> Iterable<S> saveAll(Iterable<S> iterable) { return null; }

        @Override
        public Optional<Diploma> findById(Integer integer) {return Optional.empty(); }

        @Override
        public boolean existsById(Integer integer) {return false; }

        @Override
        public Iterable<Diploma> findAll() { return null; }

        @Override
        public Iterable<Diploma> findAllById(Iterable<Integer> iterable) { return null; }

        @Override
        public long count() { return 0;}

        @Override
        public void deleteById(Integer integer) {

        @Override
        public void delete(Diploma diploma) {}

        @Override
        public void deleteAllById(Iterable<? extends Integer> iterable) {}

        @Override
        public void deleteAll(Iterable<? extends Diploma> iterable) {}

        @Override
```

```
        public void deleteAll() {}
    };


    Ομοίως με την κλάση UserService, από εδώ και κάτω είναι τα services που χρησιμοποιώ για να καλέσω
τις μεθόδους του DiplomaDao.

    public List<Diploma> listAll() { return (List<Diploma>) repo.findAll(); }

    public void save(Diploma diploma) { repo.save(diploma); }

    public List<Diploma> findDiplomasAvailable() { return repo.findDiplomasAvailable(); }

    public Diploma getDiploma(Integer id) { Optional<Diploma> result = repo.findById(id);
        return result.get(); }

    public void delete(Integer id) { repo.deleteById(id); }

    public Diploma findDiploma(String title){ return repo.findDiploma(title);}

    public List<Diploma> findDiplomas(Integer user_id){ return repo.findDiplomas(user_id);}

}
```

ε. Κλάση ApplicationsService

```
package com.diplomatics.services;

import com.diplomatics.entity.Applications;
import com.diplomatics.dao.ApplicationsDao;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class ApplicationsService {

    Ομοίως με τις προηγούμενες κλάσεις
    @Autowired
    private ApplicationsDao repo = new ApplicationsDao() {
        @Override
        public Applications findApplicationAgain(int diploma_id, int student_id) { return null;}

        @Override
        public List<Applications> findApplicationStud(String user) {return null;}

        @Override
        public List<Applications> findApplication(int student_id) {return null;}

        @Override
        public List<Applications> applyUsers(int student_id, int diploma_id) {return null;}

        @Override
        public Applications applyFinalUser(int student_id, int diploma_id) {return null;}

        @Override
        public List<Applications> dontApplyUser(int student_id) { return null;}

        @Override
        public Applications findUsersApp(int student_id) {return null; }

        @Override
        public List<Applications> findByDiploma(int diploma_id) { return null;}

        @Override
        public <S extends Applications> S save(S s) { return null; }

        @Override
        public <S extends Applications> Iterable<S> saveAll(Iterable<S> iterable) { return null; }

        @Override
        public Optional<Applications> findById(Integer integer) { return Optional.empty();}
```

```
        @Override
        public boolean existsById(Integer integer) { return false;}

        @Override
        public Iterable<Applications> findAll() {return null; }

        @Override
        public Iterable<Applications> findAllById(Iterable<Integer> iterable) { return null; }

        @Override
        public long count() { return 0; }

        @Override
        public void deleteById(Integer integer) {}

        @Override
        public void delete(Applications applications) {}

        @Override
        public void deleteAllById(Iterable<? extends Integer> iterable) { }

        @Override
        public void deleteAll(Iterable<? extends Applications> iterable) { }

        @Override
        public void deleteAll() {}
    };


    Ομοίως με την κλάση UserService, από εδώ και κάτω είναι τα services που χρησιμοποιώ για να καλέσω
τις μεθόδους του ApplicationsDao.

    public void save(Applications application) {
        repo.save(application);
    }

    public void delete(Applications application) { repo.delete(application); }

    public Applications getApplicationAgain(Integer diploma_id, Integer student_id) {
        return repo.findApplicationAgain(diploma_id, student_id);
    }

    public List<Applications> getApplicationStud(String user){
        return repo.findApplicationStud(user);
    }

    public List<Applications> getApplication(Integer student_id) {
        return repo.findApplication(student_id);
    }

    public List<Applications> applyUsers(Integer student_id, Integer diploma_id) { return
repo.applyUsers(student_id, diploma_id); }

    public Applications applyFinalUser(Integer student_id, Integer diploma_id) { return
repo.applyFinalUser(student_id, diploma_id); }

    public List<Applications> dontApplyUser(Integer student_id) { return
repo.dontApplyUser(student_id); }

    public Applications findUsersApp(Integer student_id) { return repo.findUsersApp(student_id); }

    public List<Applications> findByDiploma(Integer diploma_id) { return
repo.findByDiploma(diploma_id); }
}
```

στ. Κλάση SupervisedService

```
package com.diplomatics.services;

import com.diplomatics.entity.Supervised;
import com.diplomatics.dao.SupervisedDao;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
```

```
import java.util.Optional;

@Service
public class SupervisedService {

    Ομοίως με τις προηγούμενες κλάσεις
    @Autowired
    private SupervisedDao supervisedRepository=new SupervisedDao() {
        @Override
        public List<Supervised> findSupervised(int student_id) {return null;}

        @Override
        public List<Supervised> findSupervisedStu(int prof_id) {return null; }

        @Override
        public Supervised findSupervisedById(int super_id) {return null;}

        @Override
        public <S extends Supervised> S save(S s) { return null; }

        @Override
        public <S extends Supervised> Iterable<S> saveAll(Iterable<S> iterable) {return null;}

        @Override
        public Optional<Supervised> findById(Integer integer) {return Optional.empty(); }

        @Override
        public boolean existsById(Integer integer) { return false; }

        @Override
        public Iterable<Supervised> findAll() { return null;}

        @Override
        public Iterable<Supervised> findAllById(Iterable<Integer> iterable) {return null; }

        @Override
        public long count() {return 0;}

        @Override
        public void deleteById(Integer integer) {}

        @Override
        public void delete(Supervised supervised) {}

        @Override
        public void deleteAllById(Iterable<? extends Integer> iterable) { }

        @Override
        public void deleteAll(Iterable<? extends Supervised> iterable) {}

        @Override
        public void deleteAll() { }
    };


    Ομοίως με την κλάση UserService, από εδώ και κάτω είναι τα services που χρησιμοποιώ για να καλέσω
τις μεθόδους του SupervisedDao

    public List<Supervised> findSupervised(Integer student_id){
        return supervisedRepository.findSupervised(student_id);
    }
    public List<Supervised> findSupervisedStu(Integer prof_id){
        return supervisedRepository.findSupervisedStu(prof_id);
    }

    public void save(Supervised supervised) {
        supervisedRepository.save(supervised);
    }

   public Supervised findSupervisedById(Integer super_id){return
supervisedRepository.findSupervisedById(super_id);}
}
```

5. Πακέτο dao

Σε αυτό το πακέτο εμπεριέχονται όλες οι κλάσεις του Repository/Dao Layer. Κάθε μία από αυτές περιλαμβάνει τα απαραίτητα ερωτήματα (queries) που πραγματοποιούνται στην βάση δεδομένων.

α. Κλάση UserDao

```
package com.diplomatics.dao;

import com.diplomatics.entity.User;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import java.math.BigDecimal;
import java.util.List;
import java.util.Optional;

@Repository
public interface UserDao extends CrudRepository<User, Integer> {



    Εύρεση του χρήστη με το username
    Optional<User> findByUserName(String username);



    Εύρεση των χρηστών που έχουν κάνει apply σε μια συγκεκριμένη διπλωματική και βρίσκονται σε αναμονή
    @Query(value = "SELECT * FROM users INNER JOIN applications ON users.id = applications.student_id
where applications.diploma_id =?1 AND applications.selected='Σε Αναμονή'", nativeQuery = true)
    List<User> findAppStudents(int app_id);



    Εύρεση των χρηστών που έχουν κάνει apply σε μια συγκεκριμένη διπλωματική
    @Query(value = "SELECT * FROM users INNER JOIN applications ON users.id = applications.student_id
where applications.diploma_id =?1", nativeQuery = true)
    List<User> getStudentsReady(int app_id);



    Εύρεση των χρηστών που έχουν κάνει apply σε μια συγκεκριμένη διπλωματική, βρίσκονται σε αναμονή,
έχουν μέσο όρο μεγαλύτερο από αυτόν που θα καθορίσει ο καθηγητής και ομοίως λιγότερα μαθήματα
    @Query(value = "SELECT * FROM users INNER JOIN applications ON users.id = applications.student_id
where applications.diploma_id =?1 AND applications.selected='Σε Αναμονή' AND avg_grade > ?2 AND
remain_courses< ?3", nativeQuery = true)
    List<User> getAppStudentsSets(int app_id, BigDecimal grade, int courses);



    Εύρεση του χρήστη με το username
    @Query(value = "SELECT * FROM users WHERE user_name = ?1", nativeQuery = true)
    User findExistUser(String user_name);



    Εύρεση του χρήστη με το ονοματεπώνυμο
    @Query(value = "SELECT * FROM users WHERE full_name = ?1", nativeQuery = true)
    User findExistUserFullname(String full_name);

}
```

β. Κλάση DiplomaDao

```
package com.diplomatics.dao;

import com.diplomatics.entity.Diploma;

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
```

```
public interface DiplomaDao extends CrudRepository<Diploma, Integer> {

    Εύρεση των διπλωματικών του χρήστη με το used_id
    @Query(value = "SELECT * FROM diploma WHERE user_id = ?1 ", nativeQuery = true)
    List<Diploma> findDiplomas(int user_id);

    Εύρεση των διπλωματικών που είναι διαθέσιμες
    @Query(value = "SELECT * FROM diploma WHERE available = 'Ναι' ", nativeQuery = true)
    List<Diploma> findDiplomasAvailable();

    Εύρεση μιας διπλωματικής με έναν συγκεκριμένο τίτλο
    @Query(value = "SELECT * FROM diploma WHERE title = ?1 ", nativeQuery = true)
    Diploma findDiploma(String title);


}
```

γ. Κλάση ApplicationsDao

```
package com.diplomatics.dao;

import com.diplomatics.entity.Applications;

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
@Repository
public interface ApplicationsDao extends CrudRepository<Applications, Integer> {

    Εύρεση αν ο χρήστης έχει ξανά υποβάλει αίτηση γι
    @Query(value = "SELECT * FROM applications WHERE diploma_id = ?1 and student_id = ?2", nativeQuery
= true)
    Applications findApplicationAgain(int diploma_id, int student_id);

    Εύρεση όλων των διπλωματικών ενός χρήστη, για τις οποίες έχουν υποβληθεί αιτήσεις
    @Query(value = "SELECT * FROM diploma INNER JOIN applications ON diploma.diploma_id =
applications.diploma_id AND diploma.user =?1 GROUP BY diploma", nativeQuery = true)
    List<Applications> findApplicationStud(String user);

    Εύρεση όλων αιτήσεων που έχει κάνει ένας χρήστης
    @Query(value = "SELECT * FROM applications WHERE student_id = ?1", nativeQuery = true)
    List<Applications> findApplication(int student_id);

    Εύρεση όλων αιτήσεων που δεν έχει κάνει ένας συγκεκριμένος χρήστης για μια συγκεκριμένη διπλωματική
    @Query(value = "SELECT * FROM applications WHERE student_id != ?1 AND diploma_id=?2", nativeQuery =
true)
    List<Applications> applyUsers(int student_id, int diploma_id);

    Εύρεση της αίτησης ενός χρήστη για μια διπλωματική
    @Query(value = "SELECT * FROM applications WHERE student_id = ?1 AND diploma_id= ?2", nativeQuery =
true)
    Applications applyFinalUser(int student_id, int diploma_id);


    @Query(value = "SELECT * FROM applications WHERE student_id = ?1 ", nativeQuery = true)
    List<Applications> dontApplyUser(int student_id);

    Εύρεση του χρήστη ο οποίος επιλέχτηκε για μια διπλωματική
    @Query(value = "SELECT * FROM applications WHERE student_id = ?1 AND selected='Επιλεχθήκατε' limit
1", nativeQuery = true)
    Applications findUsersApp(int student_id);

    Εύρεση των αιτήσεων για μια συγκεκριμένη διπλωματική
    @Query(value = "SELECT * FROM applications WHERE diploma_id = ?1 ", nativeQuery = true)
    List<Applications> findByDiploma(int diploma_id);
}
```

δ. Κλάση SupervisedDao

```
package com.diplomatics.dao;

import com.diplomatics.entity.Supervised;
```

```
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface SupervisedDao extends CrudRepository<Supervised, Integer> {

    Εύρεση την επίβλεψη διπλωματικής για έναν μαθητή
    @Query(value = "SELECT * FROM supervised WHERE student_id = ?1", nativeQuery = true)
    List<Supervised> findSupervised(int student_id);

    Εύρεση τις επιβλέψεις διπλωματικών για έναν καθηγητή
    @Query(value = "SELECT * FROM supervised WHERE prof_id = ?1", nativeQuery = true)
    List<Supervised> findSupervisedStu(int prof_id);

    Εύρεση μιας συγκεκριμένης επιβλέψης διπλωματικής
    @Query(value = "SELECT * FROM supervised WHERE super_id = ?1", nativeQuery = true)
    Supervised findSupervisedById(int super_id);
}
```

6. Πακέτο config

Εδώ υπάρχει η κλάση SecurityConfiguration μέσω της οποίας υλοποιείται η αυθεντικοποίηση του χρήστη.

```
package com.diplomatics.config;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.http.HttpMethod;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;


@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

        private UserDetailsService service;

        @Autowired
        public void WebSecurityConfig(UserDetailsService service) {
            this.service = service;
        }

        @Autowired
        UserDetailsService userDetailsService;

        @Override
        protected void configure(AuthenticationManagerBuilder auth) throws Exception {
            auth.userDetailsService(userDetailsService);
        }

        Εδώ ουσιαστικά καθορίζουμε την πρόσβαση των χρηστών στην εφαρμογή μας
        @Override
        protected void configure(HttpSecurity http) throws Exception {
            http.authorizeRequests()
                    .antMatchers("/register").permitAll()
                    .antMatchers("/diploma").authenticated()
                    .antMatchers("/myprofile").authenticated()
                    .antMatchers("/diploma/info/**").authenticated()
                    .antMatchers("/diploma/**").hasAuthority("PROF")
                    .antMatchers("/applications/**").authenticated()
                    .antMatchers(HttpMethod.GET, "/users/**").authenticated()
                    .antMatchers("/users/exist/*").permitAll()
                    .and().formLogin()
                    .and()
                    .logout()
```

```
                    .logoutUrl("/logout")
                    .logoutSuccessUrl("/")
                    .and()
                    .csrf().disable().cors();
        }

    Χρησιμοποιούμε το BCryptPasswordEncoder για το hashing των κωδικών
    @Bean
    public BCryptPasswordEncoder bCryptPasswordEncoder() {
        return new BCryptPasswordEncoder();
    }

}
```

7. Λοιπές Κλάσεις

Πέραν των ανωτέρω κλάσεων, έχουν δημιουργηθεί δύο ακόμη κλάσεις:

α. Η MainController : Ελέγχει αν ο χρήστης είναι αυθεντικοποιημένος και τον αναδρομολογεί αυτόματα είτε στο login page είτε στο homepage.

```
package com.diplomatics;

import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class MainController {

    @GetMapping("")
    public String showHomePage() {

        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        String name = auth.getName();

        if (name=="anonymousUser"){
            return "index";
        }else{
            return "home";
        }
    }
}
```

β. Η MyWebApplication : Εδώ βρίσκεται η main

```
package com.diplomatics;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MyWebAppApplication {

    public static void main(String[] args) {
        SpringApplication.run(MyWebAppApplication.class, args);
    }
}
```

8. Templates

Για την το frontend της εφαρμογής έχουν δημιουργηθεί τα παρακάτω templates. Για λόγους απλότητας θα αναλυθούν μόνο τα templates που περιέχουν κώδικά Javascript.

α. applications.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
```

```
<head>
  <meta charset="UTF-8">
  <title>Αιτήσεις Διπλωματικών</title>
  <link rel="stylesheet" th:href="@{/css/styleApplication.css}" />
</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <div class="collapse navbar-collapse">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item">
          <a class="nav-link" href="/">Διαχείριση Διπλωματικών</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/myprofile">Προφίλ</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/diploma">Διπλωματικές Εργασίες</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/applications">Αιτήσεις</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/supervised">Ενεργή Διπλωματική Εργασία</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/logout">Αποσύνδεση</a>
        </li>
      </ul>
    </div>
  </nav>


  <div class="container-fluid text-center">
  <div><h2>Αιτήσεις Διπλωματικών</h2></div><br>

  <div>
    <table class="table table-bordered">
      <thead class="thead-dark">
      <tr>
        <th>ID</th>
        <th>Τίτλος</th>
        <th>Κατάσταση</th>
      </tr>
      </thead>
      <tbody>
          <th:block th:each="applications : ${listApplications}">
            <tr>
             <td>[[${applications.app_id}]]</td>
               <td>
                 <a th:href="@{'/diploma/info/'+
${applications.diploma_id}}">[[${applications.diploma}]]</a>
               </td>
               <td th:if="${role == 'PROF'} ">
                  <a th:href="@{'/applications/info/' + ${applications.diploma_id}}">Λεπτομέρειες</a>
               </td>
               <td th:if="${role== 'STUD'} ">[[${applications.selected}]]</td>
            </tr>
          </th:block>
      </tbody>
    </table>
  </div>
</body>
</html>
```

β. applications_info.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Αιτήσεις Διπλωματικών</title>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.4/jquery.min.js" integrity="sha512-
pumBsjNRGGqkPzKHndZMaAG+bir374sORyzM3uulLV14lN5LyykqNk8eEeUlUkB3U0M4FApyaHraT65ihJhDpQ=="
crossorigin="anonymous" referrerpolicy="no-referrer"></script>
  <link rel="stylesheet" th:href="@{/css/styleApplicationInfo.css}" />
</head>
```

```
<body>

<nav class="navbar navbar-expand-lg navbar-light bg-light">

  <div class="collapse navbar-collapse">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item">
        <a class="nav-link" href="/">Διαχείριση Διπλωματικών</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="/myprofile">Προφίλ</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="/diploma">Διπλωματικές Εργασίες</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="/applications">Αιτήσεις</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="/supervised">Ενεργή Διπλωματική Εργασία</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="/logout">Αποσύνδεση</a>
      </li>
    </ul>
  </div>
</nav>
  <div class="container-fluid text-center">
    <div><h2>Αιτήσεις Διπλωματικών</h2></div>
    <div><h3>Κατάλογος Μαθητών</h3></div><br>
    <div th:if="${message}" class="alert alert-success text-center">
      [[${message}]]
    </div>
    <div>
      <table class="table table-bordered">
        <thead class="thead-dark">
        <tr>
          <th>Ονοματεπώνυμο Μαθητή</th>
          <th>Μέσος Όρος</th>
          <th>Υπολοιπόμενα Μαθήματα</th>
        </tr>
        </thead>
        <tbody>
        <th:block th:each="user : ${listUsers}">
          <tr>
            <td>[[${user.full_name}]]</td>
            <td>[[${user.avg_grade}]]</td>
            <td>[[${user.remain_courses}]]</td>
          </tr>
        </th:block>
        </tbody>
      </table>
    </div>
    <div class="text-center" th:if="${diploma.available == 'Ναι' && !listUsers.isEmpty() }">
      <div class="form-group row">
        <a th:href="@{'/applications/random/' + ${id}}">
          <button type="button">Τυχαία Επιλογή</button>
        </a>
        <a th:href="@{'/applications/bestgrade/' + ${id}}">
          <button type="button">Καλύτερος Βαθμός</button>
        </a>
        <a th:href="@{'/applications/fewcourses/' + ${id}}">
          <button type="button">Λιγότερα Μαθήματα</button>
        </a>
        <button onclick="showForm()">Καθορισμός Τιμών</button> → Όταν ο καθηγητής επιλέξει να εισάγει
      </div>                                                    τον ελάχιστο βαθμό και τα μέγιστα
    </div>                                                      μαθήματα για να κάνει την αναθέσει του
  </div>                                                        εμφανίζεται μια νέα φόρμα
  <div id="form-container" style="display: none; padding-top:50px">
    <form th:action="@{/applications/set}" method="post" style="max-width: 500px; margin: 0 auto;">
      <input type="hidden"  id="id" name="id" th:value="${id}">
      <div class="border border-secondary rounded p-3">
        <div class="form-group row">
          <label class="col-sm-4 col-form-label">Όριο Βαθμού:</label>
          <div class="col-sm-8">
            <input type="text" id="grade" name="grade" class="form-control" maxlength="5"
```

```
onkeyup="validateInput()" required/> → Σε κάθε πλήκτρο που πατάει ο χρήστης καλείται η μέθοδος
          </div>                        validateInput
        </div>
        <div class="form-group row">
          <label class="col-sm-4 col-form-label">Οριο Μαθημάτων</label>
          <div class="col-sm-8">
            <input type="text" id="courses" name="courses" class="form-control" maxlength="3"
onkeyup="validateInput()" required/> → Ομοίως η μέθοδος validateInput
          </div>
        </div>

        <div class="form-group row">
          <button type="submit">Υποβολή</button>
        </div>
      </div>
    </form>

    <script>
      function showForm() { → Εμφανίζεται μια νέα φόρμα
        document.getElementById("form-container").style.display = "block";
      }

      function validateInput(){ →Ελέγχει αν ο βαθμός είναι εντός των ορίων και αν περιέχει μόνο
αριθμούς καθώς και για τα μαθήματα
        var grade = document.getElementById("grade");
        grade.value = grade.value.replace(/[^0-9\.]/g, '');

        if (grade.value > 10 || grade.value < 0) {
          alert("Ο μέγιστος βαθμός είναι 10 και ο ελάχιστος 0")
          $('#grade').val("");
        }

        var courses = document.getElementById("courses");
        courses.value = courses.value.replace(/[^0-9]/g, '');

        if (courses < 0) {
          alert("Τα μαθήματα δεν μπορεί να είναι λιγότερα από 0")
          $('#courses').val("");
        }
      }

    </script>
  </div>
</body>
</html>
```

γ. diploma.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Διπλωματικές Εργασίες</title>
  <link rel="stylesheet" th:href="@{/css/styleDiploma.css}" />
</head>
<body>

  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <div class="collapse navbar-collapse">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item">
          <a class="nav-link" href="/">Διαχείριση Διπλωματικών</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/myprofile">Προφίλ</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/diploma">Διπλωματικές Εργασίες</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/applications">Αιτήσεις</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/supervised">Ενεργή Διπλωματική Εργασία</a>
        </li>
        <li class="nav-item">
```

```
          <a class="nav-link" href="/logout">Αποσύνδεση</a>
        </li>
      </ul>
    </div>
  </nav>

  <div class="container-fluid text-center">
    <div><h2>Διπλωματικές Εργασίες</h2></div><br>
    <div class="m-2">
      <a class="h3" th:if="${role} == 'PROF'" th:href="@{/add-diploma}">Καταχώρηση Νέας
Διπλωματικής</a>
    </div>
    <div th:if="${message}" class="alert alert-success text-center">
      [[${message}]]
    </div>
    <td>
      <table class="table table-bordered">
        <thead class="thead-dark">
        <tr>
          <th>ID</th>
          <th>Τίτλος</th>
          <th>Αντικείμενο</th>
          <th th:if="${role} == 'STUD'">Καθηγητής</th>
          <th>Διαθεσιμότητα</th>
          <th ></th>
        </tr>
        </thead>
        <tbody>
        <th:block th:each="diploma : ${listDiplomas}">
          <tr>
            <td>[[${diploma.diploma_id}]]</td>
            <td>[[${diploma.title}]]</td>
            <td>[[${diploma.objectives}]]</td>
            <td th:if="${role} == 'STUD'">[[${diploma.user}]]</td>
            <td th:if="${role == 'PROF'} ">
              <div th:if="${role == 'PROF' && diploma.available == 'Ναι'} ">
                <a class="h4 mr-3"  th:href="@{'/diploma/edit/' +
${diploma.diploma_id}}">Επεξεργασία</a>
                <a class="h4" th:href="@{'/diploma/delete/' + ${diploma.diploma_id}}">Διαγραφή</a>
              </div>
              <div th:unless="${role == 'PROF' && diploma.available == 'Ναι'}">
                Έχει ανατεθεί σε μαθητή
              </div>
            </td>
            <td th:if="${role == 'STUD'} ">
                <div th:if="${flag=='yes'}">
                  <div th:if="${role == 'STUD' && diploma.available == 'Ναι'} ">
                    <a class="h4 mr-3"  th:href="@{'/diploma/info/' +
${diploma.diploma_id}}">Λεπτομέρειες</a>
                    <a class="h4 mr-3"  th:href="@{'/applications/apply/' +
${diploma.diploma_id}}">Apply</a>
                  </div>
                  <div th:unless="${role == 'STUD' && diploma.available == 'Ναι'}">
                    Μη διαθέσιμη. Ανατέθηκε σε άλλο μαθητή
                  </div>
                </div>
                <div th:unless="${flag=='yes'}">
                  Δεν μπορεί να σας ανατεθεί επιπλέον Διπλωματική
                </div>
            </td>
          </tr>
        </th:block>
        </tbody>
      </table>
    </div>
  </div>
</body>
</html>
```

δ. diploma_form.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
```

```
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.4/jquery.min.js" integrity="sha512-
pumBsjNRGGqkPzKHndZMaAG+bir374sORyzM3uulLV14lN5LyykqNk8eEeUlUkB3U0M4FApyaHraT65ihJhDpQ=="
crossorigin="anonymous" referrerpolicy="no-referrer"></script>
  <title>[[${pageTitle}]]</title>
  <link rel="stylesheet" th:href="@{/css/styleDiploma.css}" />
</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <div class="collapse navbar-collapse">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item">
          <a class="nav-link" href="/">Διαχείριση Διπλωματικών</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/myprofile">Προφίλ</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/diploma">Διπλωματικές Εργασίες</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/applications">Αιτήσεις</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/supervised">Ενεργή Διπλωματική Εργασία</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/logout">Αποσύνδεση</a>
        </li>
      </ul>
    </div>
  </nav>
  <div class="container-fluid">
    <div class="text-center"><h2>[[${pageTitle}]]</h2></div>
    <form th:action="@{/diploma/save}" method="post" th:object="${diploma}" style="max-width: 500px;
margin: 0 auto;">
      <input type="hidden" id="diploma_id" th:field="*{diploma_id}">
      <div class="border border-secondary rounded p-3">
        <div class="form-group row">
          <label class="col-sm-4 col-form-label">Τίτλος:</label>
          <div class="col-sm-8">
            <input type="text" id="title" th:field="*{title}" class="form-control" required
minlength="1" maxlength="50" onkeyup="validateTitle()"/> → Σε κάθε πάτημα πλήκτρου, κατά την
          </div>                                          εισαγωγή του τίτλου ελέγχεται αν αυτός
        </div>                                            έχει ήδη καταχωρηθεί μέσω της μεθόδου
        <div class="form-group row">                      validateTitle
          <label class="col-sm-4 col-form-label">Σκοπός-Στόχοι Διπλωματικής:</label>
          <div class="col-sm-8">
            <textarea  cols="40" rows="5" id="objectives" th:field="*{objectives}" class="form-control"
required minlength="1" maxlength="255" />
          </div>
        </div>
        <input type="hidden"  id="available" th:field="*{available}">
        <input type="hidden" id="user" th:field="*{user}">
        <div class="text-center">
          <button type="submit" class="btn btn-primary m-2">Αποθήκευση</button>
          <button type="button" class="btn btn-secondary m-2"
onclick="window.location.href='/diploma'">Ακύρωση</button>
        </div>
      </div>
    </form>
  </div>
  <script>
    function validateTitle() { → Η μέθοδος αυτή φτιάχνει ένα Json αντικείμενο και το προωθεί στον
      setTimeout(() => {          DiplomaController ου ελέγχει αν ο τίτλος υπάρχει ήδη. Αν υπάρχει
                                  εμφανίζει κατάλληλο popup και καθαρίζει το πεδίο.
       var formData = {
            diploma_id : $("#diploma_id").val(),
            title : $("#title").val(),
            objectives : $("#objectives").val(),
            available : $("#available").val(),
            user : $("#user").val()
        };
        $.ajax({
        type: "POST",
        contentType : "application/json",
        url: "/diploma/existDiploma",
```

```
        data : JSON.stringify(formData),
       dataType : 'json',
        success: function(data){
        console.log(data);
          if(data.message=="Ναι"){
            alert("Ο τίτλος υπάρχει ήδη!")
            $('#title').val("");
          }
        }
      });
    }, 1000);
   };
  </script>
</body>
</html>
```

ε. diploma_info.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Λεπτομέρειες</title>
    <link rel="stylesheet" th:href="@{/css/styleDiplomaInfo.css}" />
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="collapse navbar-collapse">
            <ul class="navbar-nav mr-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/">Διαχείριση Διπλωματικών</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/myprofile">Προφίλ</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/diploma">Διπλωματικές Εργασίες</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/applications">Αιτήσεις</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/supervised">Ενεργή Διπλωματική Εργασία</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/logout">Αποσύνδεση</a>
                </li>
            </ul>
        </div>
    </nav>
    <div class="container-fluid text-center">
        <a class="info">[[${info}]]</a>
        <a class="professor">Υπεύθυνος Καθηγητής</a>
        <a class="professorUser">[[${diploma.user}]]</a>
        <a class="purpose">Σκοπός</a>
        <a class="diplomaObj">[[${diploma.objectives}]]</a>
        <div th:if="${diploma.available == 'Ναι' && role=='STUD'} "></div>
        <a class="apply" th:href="@{'/applications/apply/' + ${diploma.diploma_id}}" >Apply</a>
    </div>
</body>
</html>
```

στ. home.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Κεντρική Σελίδα</title>
  <link rel="stylesheet" th:href="@{/css/styleHome.css}" />
</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <div class="collapse navbar-collapse">
```

```
      <ul class="navbar-nav mr-auto">
        <li class="nav-item">
          <a class="nav-link" href="/">Διαχείριση Διπλωματικών</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/myprofile">Προφίλ</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/diploma">Διπλωματικές Εργασίες</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/applications">Αιτήσεις</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/supervised">Ενεργή Διπλωματική Εργασία</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/logout">Αποσύνδεση</a>
        </li>
      </ul>
    </div>
  </nav>
  <div class="container-fluid text-center">
    <div><h2>Επισκόπηση Εφαρμογής</h2> <br>
      <p> Ο σκοπός της εφαρμογής αυτής είναι να επιτρέπει στους μαθητές να επιλέγουν ανάμεσα σε
διαθέσιμες διπλωματικές εργασίες από διάφορους καθηγητές και να υποβάλλουν
        αίτηση για ανάληψη διπλωματικής που τους ενδιαφέρει. Η εφαρμογή επιπλέον, επιτρέπει στους
καθηγητές να αναθέτουν διπλωματικές στους μαθητές και να παρακολουθούν την
        πορεία εκπόνησής τους.
      </p>
    </div>
  </div>
</body>
</html>
```

ζ. index.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Κεντρική Σελίδα</title>
    <link rel="stylesheet" th:href="@{/css/styleIndex.css}" />
</head>
<body>
    <div class="container-fluid text-center">
        <h1>Καλώς ήρθατε στο Σύστημα Διαχείρισης Διπλωματικών Εργασίων</h1><br>
        <a class="h3" th:href="@{/login}">Είσοδος</a> <br><br>
        <a class="h3" th:href="@{/register}">Εγγραφή</a>
    </div>
</body>
</html>
```

η. myprofile.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Το Προφίλ μου</title>
  <link rel="stylesheet" th:href="@{/css/styleMyProfile.css}" />
</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <div class="collapse navbar-collapse">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item">
          <a class="nav-link" href="/">Διαχείριση Διπλωματικών</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/myprofile">Προφίλ</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/diploma">Διπλωματικές Εργασίες</a>
```

```
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/applications">Αιτήσεις</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/supervised">Ενεργή Διπλωματική Εργασία</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/logout">Αποσύνδεση</a>
        </li>
      </ul>
    </div>
  </nav>
  <div class="container-fluid text-center">
    <div><h2>Το Προφίλ μου</h2></div>
    <div th:if="${message}" class="alert alert-success text-center">
      [[${message}]]
    </div>
    <div>
      <table class="table">
        <thead class="thead-dark">
        <tr>
          <th>Username</th>
        </tr>
        </thead>
        <tbody>
        <th:block th:each="user : ${listUsers}">
          <tr>
            <td><a th:href="@{'/users/edit/' + ${user.id}}">[[${user.userName}]] </a></td>
          </tr>
        </th:block>
        </tbody>
      </table>
    </div>
  </div>
</body>
</html>
```

θ. supervised.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Αναθετημένες Διπλωματικές Εργασίες</title>
  <link rel="stylesheet" th:href="@{/css/styleSupervised.css}" />
</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <div class="collapse navbar-collapse">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item">
          <a class="nav-link" href="/">Διαχείριση Διπλωματικών</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/myprofile">Προφίλ</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/diploma">Διπλωματικές Εργασίες</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/applications">Αιτήσεις</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/supervised">Ενεργή Διπλωματική Εργασία</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/logout">Αποσύνδεση</a>
        </li>
      </ul>
    </div>
  </nav>
  <div class="container-fluid text-center">
  <div><h2>Αναθετημένες Διπλωματικές Εργασίες</h2></div><br>
  <div th:if="${message}" class="alert alert-success text-center">
```

```
    [[${message}]]
  </div>
  <div>
    <table class="table table-bordered">
      <thead class="thead-dark">
      <tr>
        <th>Τίτλος</th>
        <th th:if="${role == 'PROF'} ">Ονοματεπώνυμο Μαθητή</th>
        <th>Implement Grade</th>
        <th>Report Grade</th>
        <th>Present Grade</th>
        <th>Overall Grade</th>
        <th th:if="${role == 'PROF'} " >Βαθμολόγηση</th>
      </tr>
      </thead>
      <tbody>
      <th:block th:each="supervised : ${listSupervised}">
        <tr>
          <td>[[${supervised.diploma}]]</td>
          <td th:if="${role == 'PROF'} ">[[${supervised.student}]]</td>
          <td>[[${supervised.implem_grade}]]</td>
          <td>[[${supervised.report_grade}]]</td>
          <td>[[${supervised.present_grade}]]</td>
          <td>[[${supervised.overall_grade}]]</td>
          <td th:if="${role == 'PROF'} ">
            <a th:href="@{'/supervised/edit/' + ${supervised.super_id}}">Επεξεργασία</a>
          </td>
        </tr>
      </th:block>
      </tbody>
    </table>
  </div>
</div>
</body>
</html>
```

ι. Supervised-form.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Βαθμολόγηση Διπλωματικής</title>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.4/jquery.min.js" integrity="sha512-
pumBsjNRGGqkPzKHndZMaAG+bir374sORyzM3uulLV14lN5LyykqNk8eEeUlUkB3U0M4FApyaHraT65ihJhDpQ=="
crossorigin="anonymous" referrerpolicy="no-referrer"></script>
  <link rel="stylesheet" th:href="@{/css/styleSupervisedForm.css}" />
</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <div class="collapse navbar-collapse">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item">
          <a class="nav-link" href="/">Διαχείριση Διπλωματικών</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/myprofile">Προφίλ</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/diploma">Διπλωματικές Εργασίες</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/applications">Αιτήσεις</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/supervised">Ενεργή Διπλωματική Εργασία</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/logout">Αποσύνδεση</a>
        </li>
      </ul>
    </div>
  </nav>
  <div class="container-fluid">
    <div class="text-center"><h2>[[${pageTitle}]]</h2></div>
```

```
    <form th:action="@{/supervised/save}" method="post" th:object="${supervised}" style="max-width:
500px; margin: 0 auto;">
      <input type="hidden" th:field="*{super_id}">
      <input type="hidden" th:field="*{diploma}">
      <input type="hidden" th:field="*{diploma_id}">
      <input type="hidden" th:field="*{student}">
      <input type="hidden" th:field="*{student_id}">
      <input type="hidden" th:field="*{prof}">
      <input type="hidden" th:field="*{prof_id}">
      <div class="border border-secondary rounded p-3">
        <div class="form-group row">
          <label class="col-sm-4 col-form-label">Βαθμός Υλοποίησης:</label>
          <div class="col-sm-8">
            <input type="text" id="implem_grade" name="implem_grade" th:field="*{implem_grade}"
class="form-control" required minlength="1" maxlength="5" onkeyup="validateInput()"/> → Σε κάθε πάτημα
          </div>                                                                    καλείται η μέθοδος
        </div>                                                                      validateInput
        <div class="form-group row">
          <label class="col-sm-4 col-form-label">Βαθμός Συγγραφής:</label>
          <div class="col-sm-8">
            <input type="text" id="report_grade" name="report_grade" th:field="*{report_grade}"
class="form-control" required minlength="1" maxlength="5" onkeyup="validateInput()"/> →Ομοίως
          </div>
        </div>
        <div class="form-group row">
          <label class="col-sm-4 col-form-label">Βαθμός Παρουσίασης:</label>
          <div class="col-sm-8">
            <input type="text" id="present_grade" name="present_grade" th:field="*{present_grade}"
class="form-control" required minlength="1" maxlength="5" onkeyup="validateInput()"/> →Ομοίως
          </div>
        </div>
        <div class="form-group row">
          <label class="col-sm-4 col-form-label">Τελικός Βαθμός:</label>
          <div class="col-sm-8">
            <input type="text" id="overall_grade" name="overall_grade" th:field="*{overall_grade}"
class="form-control"  minlength="1" maxlength="5" readonly /> →Ο συνολικός βαθμός παράγεται αυτόματα
          </div>                                                    και ενημερώνεται το πεδίο
        </div>
        <div class="text-center">
          <button type="submit" class="btn btn-primary m-2">Αποθήκευση</button>
          <button type="button" class="btn btn-secondary m-2"
onclick="window.location.href='/supervised'">Ακύρωση</button>
        </div>
      </div>
    </form>
  </div>
  <script>

    function validateInput() {     → Η μέθοδος αυτή ελέγχει αν εισάγονται μόνο αριθμοί και η . (τελεία)
      var input1 = document.getElementById("implem_grade");  και αν βρίσκεται ο αριθμός εντός των ορίων
      var input2 = document.getElementById("report_grade");  Αν ο καθηγητής εισάγει και τους 3 βαθμούς
      var input3 = document.getElementById("present_grade"); γίνεται αυτόματα ο υπολογισμός του
      input1.value = input1.value.replace(/[^0-9\.]/g, '');  συνολικού βαθμού και ενημερώνεται το πεδίο
      input2.value = input2.value.replace(/[^0-9\.]/g, '');
      input3.value = input3.value.replace(/[^0-9\.]/g, '');

      if (input1.value > 10 || input1.value < 0) {
        alert("Ο μέγιστος βαθμός είναι 10 και ο ελάχιστος 0")
        $('#implem_grade').val("");
      }
      if (input2.value > 10 || input2.value < 0) {
        alert("Ο μέγιστος βαθμός είναι 10 και ο ελάχιστος 0")
        $('#report_grade').val("");
      }
      if (input3.value > 10 || input3.value < 0) {
        alert("Ο μέγιστος βαθμός είναι 10 και ο ελάχιστος 0")
        $('#present_grade').val("");
      }

      if ((input1.value !="") && (input2.value !="") && (input3.value !="")){
        $('#overall_grade').val(0.7 * input1.value +0.15 * input2.value + 0.15 * input3.value);
      }
    }
  </script>
</body>
```

κ. User_form.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.4/jquery.min.js" integrity="sha512-
pumBsjNRGGqkPzKHndZMaAG+bir374sORyzM3uulLV14lN5LyykqNk8eEeUlUkB3U0M4FApyaHraT65ihJhDpQ=="
crossorigin="anonymous" referrerpolicy="no-referrer"></script>
    <title>[[${pageTitle}]]</title>
  <link rel="stylesheet" th:href="@{/css/styleUserForm.css}" />
</head>
<body>

  <div class="container-fluid">
  <div class="text-center"><h2>[[${pageTitle}]]</h2></div>

  <form th:action="@{/users/save}" method="post" th:object="${user}"
      style="max-width: 500px; margin: 0 auto;">
    <input type="hidden" th:field="*{id}">
    <div class="border border-secondary rounded p-3">
      <div class="form-group row">
        <label class="col-sm-4 col-form-label">Username:</label>
        <div class="col-sm-8">
            <input type="text" id="userName" th:field="*{userName}" class="form-control" maxlength="45"
onkeyup="validateUsername()" required/> → Καλείται η μέθοδος validateUsername

        </div>
      </div>
      <div class="form-group row">
        <label class="col-sm-4 col-form-label">Password:</label>
        <div class="col-sm-8">
          <input type="password" th:field="*{password}" class="form-control" minlength="5"
maxlength="45"  required/>
        </div>
      </div>
      <div class="form-group row">
        <label class="col-sm-4 col-form-label">Full Name:</label>
        <div class="col-sm-8">
          <input type="text" id="fullname" th:field="*{full_name}" class="form-control" maxlength="45"
onkeyup="validateFullname()" required/> → Καλείται η μέθοδος validateFullname
        </div>
      </div>
      <div class="form-group row">
        <label class="col-sm-4 col-form-label">Role:</label>
        <div class="col-sm-8">
          <select th:field="*{role}" class="form-select" aria-label="Default select example">
            <option value="STUD">Μαθητής</option>
            <option value="PROF">καθηγητής</option>
          </select>
        </div>
      </div>
      <div id="student">
        <div class="form-group row">
          <label class="col-sm-4 col-form-label">Average Grade:</label>
          <div class="col-sm-8">
            <input id="avg_grade" type="text" th:field="*{avg_grade}" class="form-control"
maxlength="15" onkeyup="validateInput()"/> → Καλείται η μέθοδος validateInput
          </div>
        </div>
        <div class="form-group row">
          <label class="col-sm-4 col-form-label">Remain Courses:</label>
          <div class="col-sm-8">
            <input id="remain_courses" type="text" th:field="*{remain_courses}" class="form-control"
maxlength="45" onkeyup="validateInput()"/> → Καλείται η μέθοδος validateInput
          </div>
        </div>
        <div class="form-group row">
          <label class="col-sm-4 col-form-label">Years Left:</label>
          <div class="col-sm-8">
            <input id="years" type="text" th:field="*{years}" class="form-control" maxlength="15"
onkeyup="validateInput()"/> → Καλείται η μέθοδος validateInput
          </div>
        </div>
      </div>
```

```
      <div id="prof" style="display: none">
        <div class="form-group row">
          <label class="col-sm-4 col-form-label">Speciality:</label>
          <div class="col-sm-8">
            <input id="speciality" type="text" th:field="*{speciality}" class="form-control"
maxlength="15" />
          </div>
        </div>
      </div>
      <div class="form-group row">
        <input type="hidden" th:field="*{enabled}">
      </div>
      <div class="text-center">
        <button type="submit" class="btn btn-primary m-2">Αποθήκευση</button>
        <button type="button" class="btn btn-secondary m-2"
onclick="window.location.href='/'">Ακύρωση</button>
      </div>
    </div>

  </form>
  </div>
  <script>
    $(function() {        → Όταν χρήστης επιλέξει να εγγραφεί  επιλέγει ποιος θα είναι ο ρόλος του
      $("#role").change(function () {    (Καθηγητής -Μαθητής). Ανάλογα την επιλογή του στο dropdown
        var role = this.value;           εμφανίζονται και αποκρύπτονται αντίστοιχα τα πεδία
        if (role == "STUD") {
          $("#student").show();
          $("#prof").hide();
          $('#speciality').val('');
          $("speciality").prop('required',false);
          $("avg_grade").prop('required',true);
          $("remain_courses").prop('required',true);
          $("years").prop('required',true);
        }else{
          $("#student").hide();
          $("#prof").show();
          $('#remain_courses').val('');
          $('#avg_grade').val('');
          $('#years').val('');
          $("speciality").prop('required',true);
          $("avg grade").prop('required',false);
          $("remain_courses").prop('required',false);
          $("years").prop('required',false);
        };
      });

    });

    function validateInput() { → Εδώ γίνεται έλεγχος αν οι βαθμοί που εισάγει ο χρήστης είναι εντός
        ορίων και αν έχουν εισαχθεί μόνο αριθμοί και .
      var avg_grade = document.getElementById("avg_grade");
      var remain_courses = document.getElementById("remain_courses");
      var years = document.getElementById("years");
      avg_grade.value = avg_grade.value.replace(/[^0-9\.]/g, '');
      remain courses.value = remain courses.value.replace(/[^0-9]/g, '');
      years.value = years.value.replace(/[^0-9]/g, '');

      if (avg_grade.value > 10 || avg_grade.value < 0) {
        alert("Ο μέγιστος βαθμός είναι 10 και ο ελάχιστος 0")
        $('#avg_grade').val("");
      }
    }

    function validateUsername() { → Ελέγχουμε αν το username έχει χρησιμοποιηθεί ξανά
      setTimeout(() => {
        var uname = $('#userName').val();
        if(uname=="anonymousUser"){
          alert("Αυτό το username απαγορεύεται!")
          $('#userName').val("");
        }else{
          $.ajax({
            type: "POST",
            url: "/users/exist/"+uname,
            success: function(msg){
              console.log(msg);
              if(msg=="Ναι"){
```

```
                alert("Το username υπάρχει ήδη!")
                $('#userName').val("");
              }
            }
          });
        }
      }, 2000);
    };

    function validateFullname() { → Ελέγχουμε αν το ονοματεπώνυμο έχει χρησιμοποιηθεί ξανά

      setTimeout(() => {
        var fname = $('#fullname').val();
        if(fname=="anonymousUser"){
          alert("Αυτό το ονοματεπώνυμο απαγορεύεται!")
          $('#fullname').val("");
        }else {
          $.ajax({
            type: "POST",
            url: "/users/existFullname/" + fname,
            success: function (msg) {
              console.log(msg);
              if (msg == "Ναι") {
                alert("Το ονοματεπώνυμο υπάρχει ήδη!")
                $('#fullname').val("");
              }
            }
          });
        }
      }, 2000);
    };

  </script>
</body>
</html>
```

λ. User_form_2.html

Όμοια με την παραπάνω φόρμα. Η διαφορά τους είναι στο ότι η υπόψη φόρμα χρησιμοποιείται μόνο από υφιστάμενους χρήστες ενώ η παραπάνω φόρμα μόνο από τους χρήστες που πρόκειται να εγγραφούν.

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Επεξεργασία Προφίλ</title>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.4/jquery.min.js" integrity="sha512-
pumBsjNRGGqkPzKHndZMaAG+bir374sORyzM3uulLV14lN5LyykqNk8eEeUlUkB3U0M4FApyaHraT65ihJhDpQ=="
crossorigin="anonymous" referrerpolicy="no-referrer"></script>
  <link rel="stylesheet" th:href="@{/css/styleUserForm_2.css}" />
</head>
<body>

<nav class="navbar navbar-expand-lg navbar-light bg-light">

  <div class="collapse navbar-collapse">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item">
        <a class="nav-link" href="/">Διαχείριση Διπλωματικών</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="/myprofile">Προφίλ</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="/diploma">Διπλωματικές Εργασίες</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="/applications">Αιτήσεις</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="/supervised">Ενεργή Διπλωματική Εργασία</a>
      </li>
```

```
      <li class="nav-item">
        <a class="nav-link" href="/logout">Αποσύνδεση</a>
      </li>
    </ul>
  </div>
</nav>

  <div class="container-fluid">
    <div class="text-center"><h2>[[${pageTitle}]]</h2></div>

    <form th:action="@{/users/save}" method="post" th:object="${user}"
          style="max-width: 500px; margin: 0 auto;">
      <input type="hidden" th:field="*{id}">
      <div class="border border-secondary rounded p-3">
        <div class="form-group row">
          <label class="col-sm-4 col-form-label">Username:</label>
          <div class="col-sm-8">
            <input type="text"  id="userName" th:field="*{userName}" class="form-control"
maxlength="45" onkeyup="validateUsername()" required/>
          </div>
        </div>
        <div class="form-group row">
          <label class="col-sm-4 col-form-label">Password:</label>
          <div class="col-sm-8">
            <input type="text" id="pass" th:field="*{password}" class="form-control" minlength="5"
maxlength="45" />
          </div>
        </div>
        <div class="form-group row">
          <label class="col-sm-4 col-form-label">Full Name:</label>
          <div class="col-sm-8">
            <input type="text" id="fullname" th:field="*{full name}" class="form-control"
maxlength="45" onkeyup="validateFullname()" required/>
          </div>
        </div>
        <div th:if="${user.role} == 'STUD'">
          <div class="form-group row">
            <label class="col-sm-4 col-form-label">Average Grade:</label>
            <div class="col-sm-8">
              <input id="avg_grade" type="text" th:field="*{avg_grade}" class="form-control"
maxlength="15" />
            </div>
          </div>
          <div class="form-group row">
            <label class="col-sm-4 col-form-label">Remain Courses:</label>
            <div class="col-sm-8">
              <input id="remain_courses" type="text" th:field="*{remain_courses}" class="form-control"
maxlength="45" />
            </div>
          </div>
          <div class="form-group row">
            <label class="col-sm-4 col-form-label">Years Left:</label>
            <div class="col-sm-8">
              <input id="years" type="text" th:field="*{years}" class="form-control" maxlength="15" />
            </div>
          </div>
        </div>
        <div th:if="${user.role} == 'PROF'" >
          <div class="form-group row">
            <label class="col-sm-4 col-form-label">Speciality:</label>
            <div class="col-sm-8">
              <input id="speciality" type="text" th:field="*{speciality}" class="form-control"
maxlength="15" />
            </div>
          </div>
        </div>
        <div class="form-group row">
          <input type="hidden" th:field="*{role}" th:value="${user.role}">
          <input type="hidden" th:field="*{enabled}">
        </div>
        <div class="text-center">
          <button type="submit" class="btn btn-primary m-2">Αποθήκευση</button>
          <button type="button" class="btn btn-secondary m-2"
onclick="window.location.href='/myprofile'">Ακύρωση</button>
        </div>
      </div>
```

```
    </form>
  </div>
<script>
  $("#avg_grade").keyup(function () {
    var grade = this.value;

    if (grade > 10) {
      alert("Ο μέγιστος βαθμός είναι 10")
      $('#avg_grade').val(0);
    }
  });
  $(document).ready(function() {
    $("#pass").val("");
  });

  function validateUsername() {
    setTimeout(() => {
      var uname = $('#userName').val();
      if(uname=="anonymousUser"){
        alert("Αυτό το username απαγορεύεται!")
        $('#userName').val("");
      }else{
        $.ajax({
          type: "POST",
          url: "/users/exist/"+uname,
          success: function(msg){
            console.log(msg);
            if(msg=="Ναι"){

              alert("Το username υπάρχει ήδη!")
              $('#userName').val("");
            }
          }
        });
      }

    }, 2000);
  };

  function validateFullname() {
    setTimeout(() => {
      var fname = $('#fullname').val();
      if(fname=="anonymousUser"){
        alert("Αυτό το ονοματεπώνυμο απαγορεύεται!")
        $('#fullname').val("");
      }else {
        $.ajax({
          type: "POST",
          url: "/users/existFullname/" + fname,
          success: function (msg) {
            console.log(msg);
            if (msg == "Ναι") {

              alert("Το ονοματεπώνυμο υπάρχει ήδη!")
              $('#fullname').val("");
            }
          }
        });
      }
    }, 2000);
  };

</script>
</body>
</html>
```

9. application.properties

Τα απαιτούμενα στοιχεία για την σύνδεση στην βάση καθορίζονται στο application.properties. Επίσης, για την υποβοήθηση του troubleshooting κατά την ανάπτυξη της εφαρμογής χρησιμοποιήθηκαν, όπως φαίνεται, διάφορα loggers.

```
spring.datasource.url=jdbc:mysql://localhost:3306/diplodb
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.show_sql=true
spring.jpa.properties.hibernate.format_sql=true
log4j.logger.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
```

10. Testing

Για τον έλεγχο της λειτουργικότητας εκτός από τους ελέγχους μέσω του GUI χρησιμοποιήθηκε και το Mockito.

α. UserControllerTest

Σε αυτή την κλάση ελέγχουμε την αποθήκευση του χρήστη

```
package com.diplomatics.controllers;

import com.diplomatics.dao.UserDao;
import com.diplomatics.entity.User;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.Mock;
import org.mockito.junit.jupiter.MockitoExtension;
import org.mockito.junit.jupiter.MockitoSettings;
import org.mockito.quality.Strictness;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;


import static org.mockito.Mockito.when;

@ExtendWith(MockitoExtension.class)
@MockitoSettings(strictness = Strictness.LENIENT)
class UserControllerTest {

    @Mock
    private Authentication auth ;

    @Mock
    private BCryptPasswordEncoder passwordEncoder=new BCryptPasswordEncoder();

    @BeforeEach
    public void initSecurityContext() {
        when(auth.getCredentials()).thenReturn("");
        when(auth.getName()).thenReturn("anonymousUser");
        SecurityContextHolder.getContext().setAuthentication(auth);
        when(passwordEncoder.encode("test")).thenReturn("test");
    }

    @MockBean
    UserDao userDao;

    UserController userController=new UserController();

    @Test
    public void addUser(){
        User user = new User();
        user.setUserName("test");
        user.setSpeciality("pc prof");
        user.setPassword("test");
        user.setRole("PROF");
        user.setFull_name("Nick Test");
        userController.saveUser(user);
    }
}
```

β. DiplomaControllerTest

Σε αυτή την κλάση ελέγχουμε την αποθήκευση καθώς και την διαγραφή μιας διπλωματικής

```
package com.diplomatics.controllers;


import com.diplomatics.entity.Diploma;
import com.diplomatics.entity.User;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.Mock;
import org.mockito.junit.jupiter.MockitoExtension;
import org.mockito.junit.jupiter.MockitoSettings;
import org.mockito.quality.Strictness;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import static org.mockito.Mockito.when;

@ExtendWith(MockitoExtension.class)
@MockitoSettings(strictness = Strictness.LENIENT)
class DiplomaControllerTest {

    @Mock
    private Authentication auth ;

    public User addUser(){
        User user = new User();
        user.setId(5);
        user.setUserName("test");
        user.setSpeciality("pc prof");
        user.setPassword("test");
        user.setRole("PROF");
        user.setFull_name("Nick Test");
        return user;

    }

    public Diploma diplomaCreate(){
        User user=addUser();
        Diploma diploma=new Diploma();
        diploma.setUser(user.getFull_name());
        diploma.setUser_id(user.getId());
        diploma.setObjectives("Testing");
        diploma.setTitle("Testing");
        return diploma;
    }

    @BeforeEach
    public void initSecurityContext() {
        when(auth.getCredentials()).thenReturn("");
        when(auth.getName()).thenReturn("test_user");
        SecurityContextHolder.getContext().setAuthentication(auth);
    }

    DiplomaController diplomaController = new DiplomaController();


    @Test
    public void deleteDiploma(){
        diplomaController.diplomaDelete(8,null);
    }

    @Test
    public void addDiploma(){
        Diploma diploma=diplomaCreate();
        diplomaController.saveDiploma(diploma,null);
    }

    @Test
    public void checkDiploma(){
        Diploma diploma=diplomaCreate();
        diplomaController.checkDiploma(diploma);
    }
}
```

γ. ApplicationsControllerTest

Ομοίως

```
package com.diplomatics.controllers;

import com.diplomatics.entity.Applications;
import com.diplomatics.entity.Diploma;
import com.diplomatics.entity.User;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.Mock;
import org.mockito.junit.jupiter.MockitoExtension;
import org.mockito.junit.jupiter.MockitoSettings;
import org.mockito.quality.Strictness;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.ui.Model;

import static org.mockito.Mockito.when;

@ExtendWith(MockitoExtension.class)
@MockitoSettings(strictness = Strictness.LENIENT)
class ApplicationsControllerTest {

    ApplicationsController applicationsController=new ApplicationsController();

    public User addUser(){
        User user = new User();
        user.setId(5);
        user.setUserName("test");
        user.setSpeciality("pc prof");
        user.setPassword("test");
        user.setRole("PROF");
        user.setFull_name("Nick Test");
        return user;

    }
    @Mock
    private Authentication auth ;

    @BeforeEach
    public void initSecurityContext() {
        when(auth.getCredentials()).thenReturn("");
        when(auth.getName()).thenReturn("test_user");
        SecurityContextHolder.getContext().setAuthentication(auth);
    }

    @Test
    public void addApplication(){
        User user=addUser();
        Applications applications=new Applications();
        applications.setDiploma("Test title diploma");
        applications.setDiploma_id(6);
        applications.setSelected("Οχι");
        applications.setStudent(user.getFull_name());
        applicationsController.showApplications((Model) applications);

    }
}
```

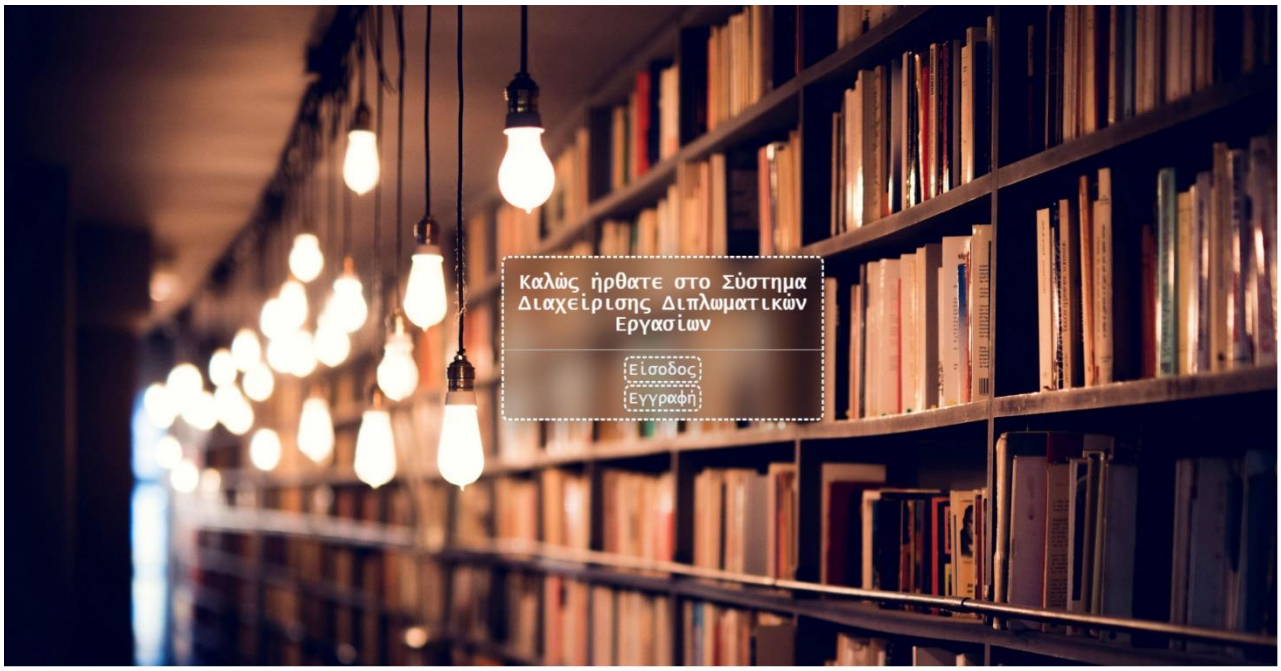
δ. SupervisedControllerTest

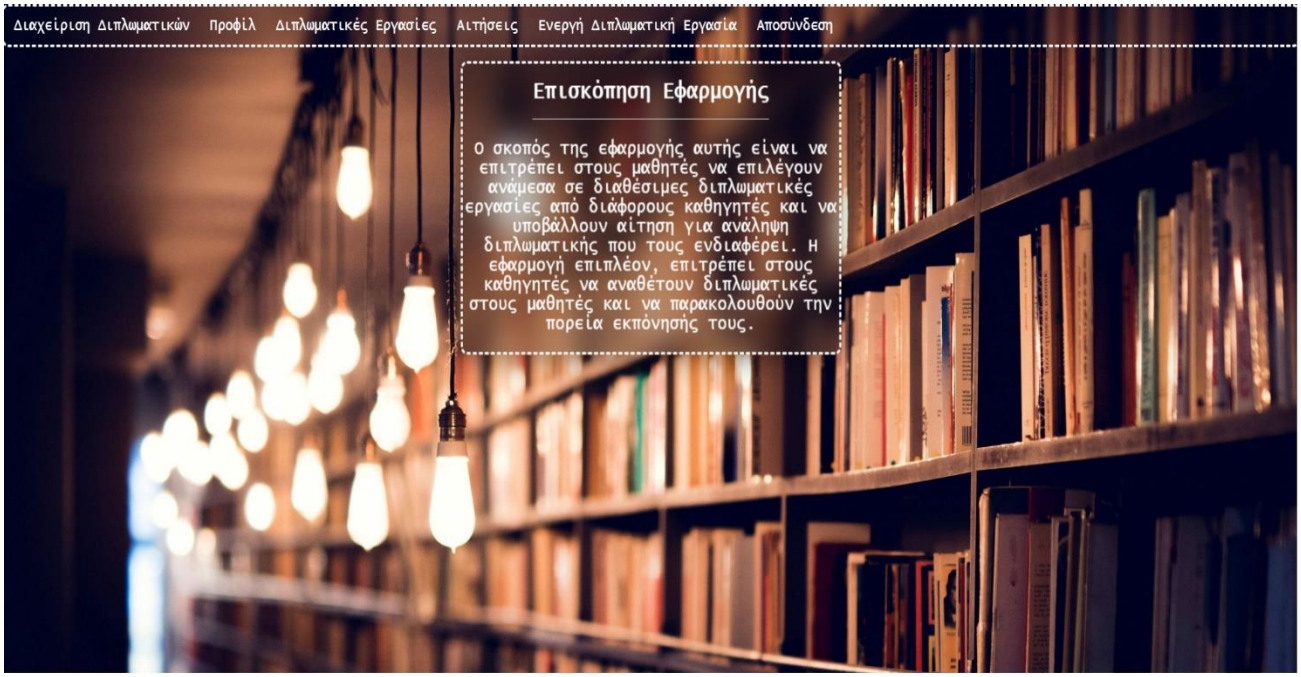
Ομοίως

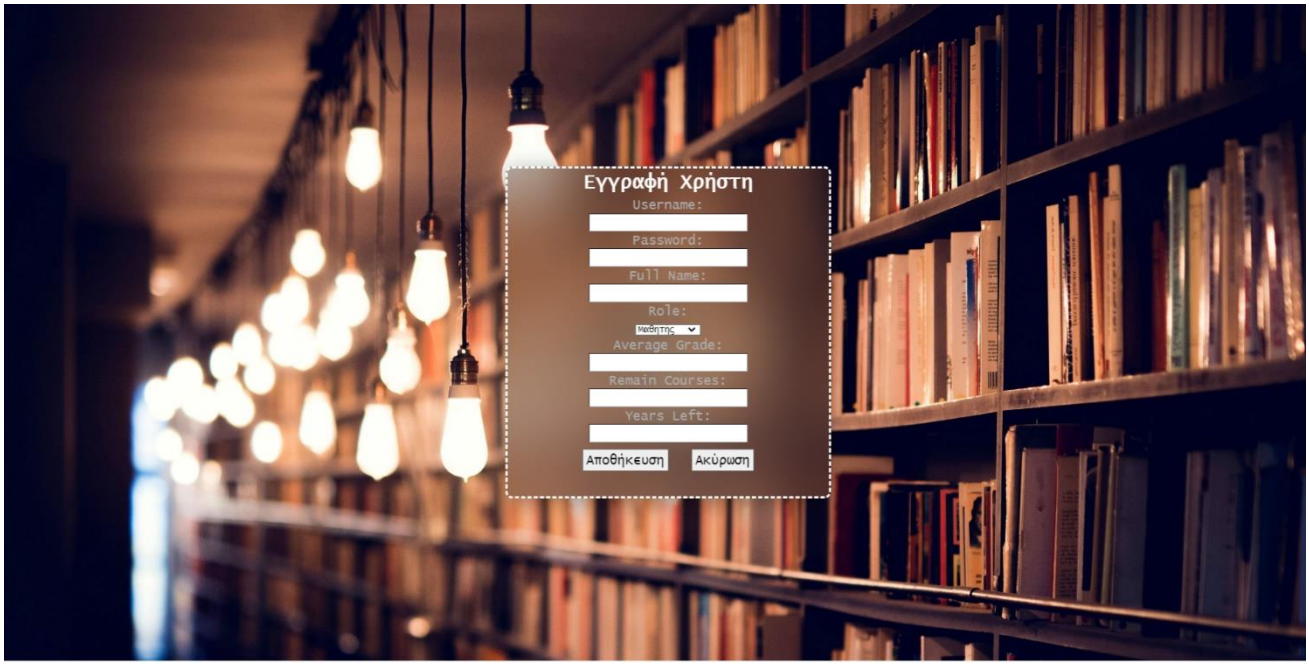
```
package com.diplomatics.controllers;

import com.diplomatics.entity.Supervised;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.junit.jupiter.MockitoExtension;
import org.mockito.junit.jupiter.MockitoSettings;
import org.mockito.quality.Strictness;
```

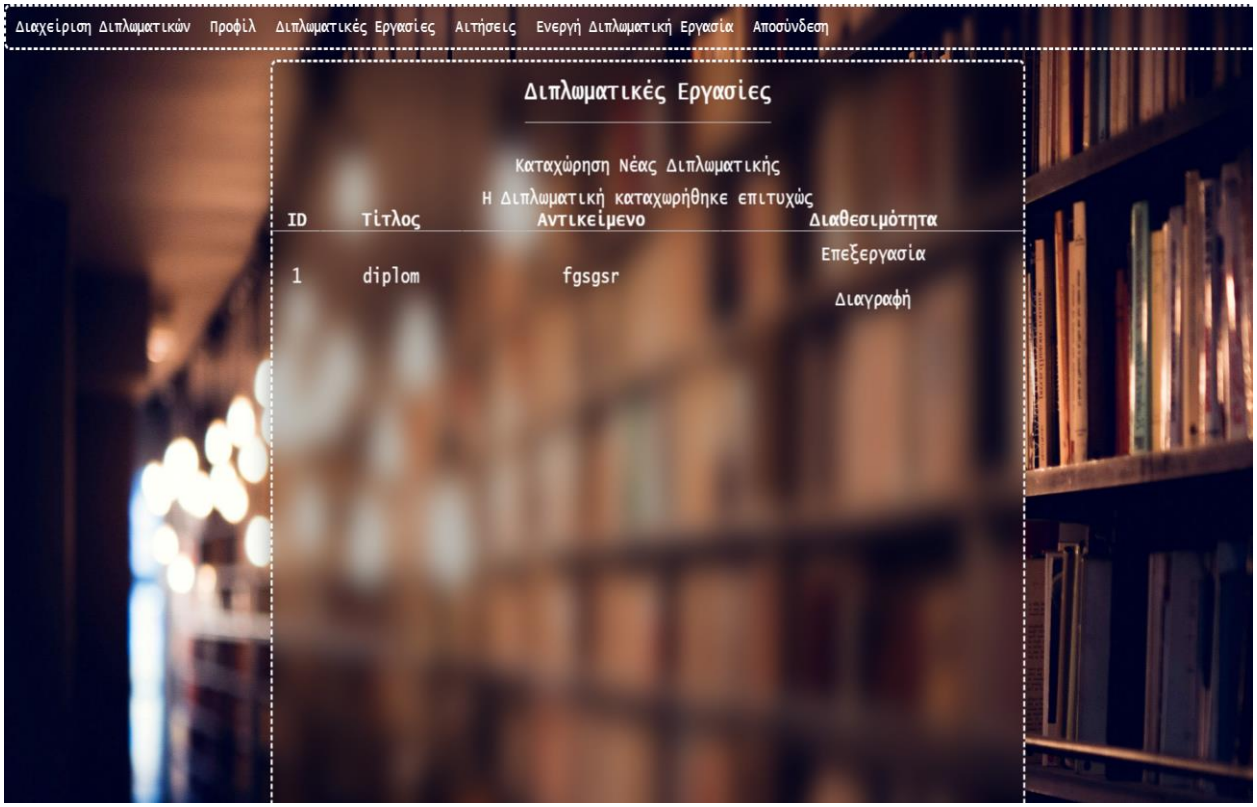
```java
import java.math.BigDecimal;


@ExtendWith(MockitoExtension.class)
@MockitoSettings(strictness = Strictness.LENIENT)
class SupervisedControllerTest {

    SupervisedController supervisedController=new SupervisedController();

    @Test
    public void saveForm(){
        Supervised supervised=new Supervised();
        supervised.setDiploma("Testing");
        supervised.setDiploma_id(4);
        supervised.setStudent("Student_Name");
        supervised.setStudent_id(6);
        supervised.setProf("Prof_Name");
        supervised.setProf_id(10);
        supervised.setImplem_grade(BigDecimal.valueOf(8.7));
        supervised.setPresent_grade(BigDecimal.valueOf(6.9));
        supervised.setReport_grade(BigDecimal.valueOf(8.0));
        BigDecimal val= BigDecimal.valueOf(0.7 * 8.7 +0.15 * 8.0 + 0.15 * 6.9);
        supervised.setOverall_grade(val);
        supervisedController.saveForm(supervised,null);
    }
}
```

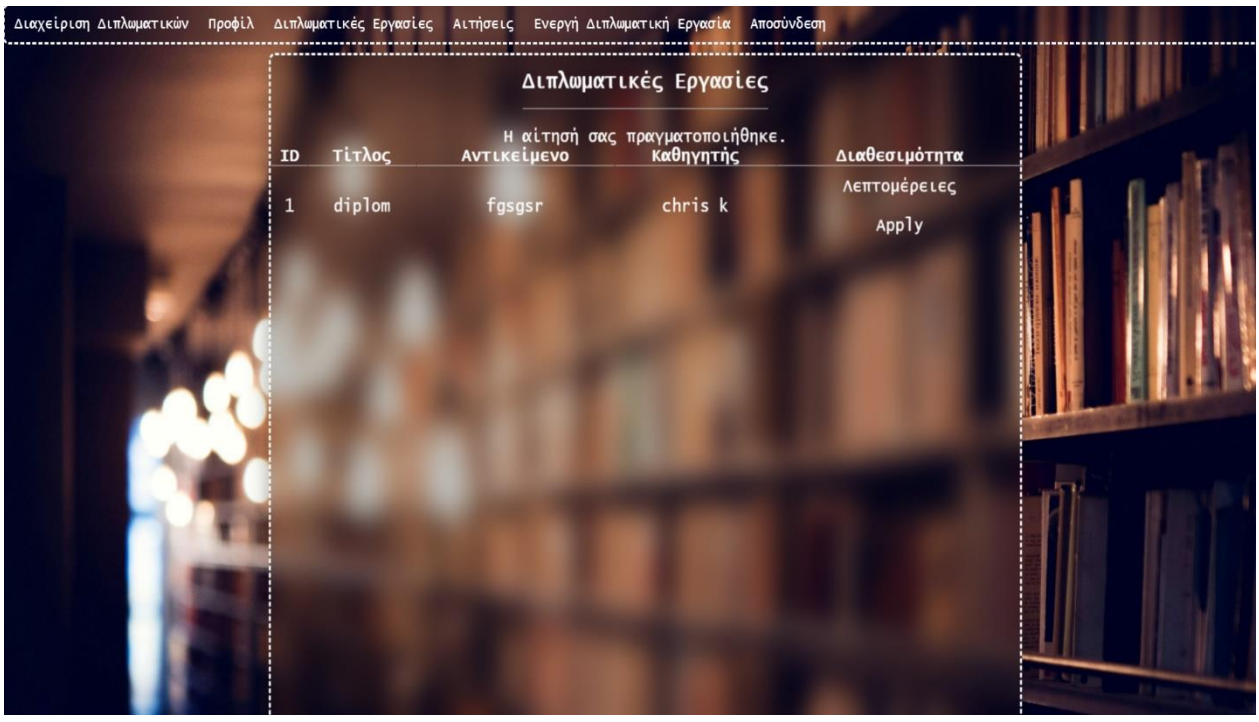
11. Screenshots Εφαρμογής




Διαχείριση Διπλωματικών
Προφίλ
Διπλωματικές Εργασίες
Αιτήσεις
Ενεργή Διπλωματική Εργασία
Αποσύνδεση
Επισκόπηση Εφαρμογής
Ο σκοπός της εφαρμογής αυτής είναι να επιτρέπει στους μαθητές να επιλέγουν ανάμεσα σε διαθέσιμες διπλωματικές εργασίες από διάφορους καθηγητές και να υποβάλλουν αίτηση για ανάληψη διπλωματικής που τους ενδιαφέρει. Η εφαρμογή επιπλέον, επιτρέπει στους καθηγητές να αναθέτουν διπλωματικές στους μαθητές και να παρακολουθούν την πορεία εκπόνησής τους.


Εγγραφή Χρήστη
Username:
Password:
Full Name:
Role:
Μαθητής
Average Grade:
Remain Courses:
Years Left:
Αποθήκευση
Ακύρωση


Διαχείριση Διπλωματικών
Προφίλ
Διπλωματικές Εργασίες
Αιτήσεις
Ενεργή Διπλωματική Εργασία
Αποσύνδεση
Διπλωματικές Εργασίες
Καταχώρηση Νέας Διπλωματικής
Η Διπλωματική καταχωρήθηκε επιτυχώς
ID
Τίτλος
Αντικείμενο
Διαθεσιμότητα
1
diplom
fgsgsr
Επεξεργασία
Διαγραφή


Διαχείριση Διπλωματικών
Προφίλ
Διπλωματικές Εργασίες
Αιτήσεις
Ενεργή Διπλωματική Εργασία
Αποσύνδεση
Διπλωματικές Εργασίες
Η αίτησή σας πραγματοποιήθηκε.
ID
Τίτλος
Αντικείμενο
Καθηγητής
Διαθεσιμότητα
1
diplom
fgsgsr
chris k
Λεπτομέρειες
Apply