Edugate

Dokumentacja

Karolina Tomaszewska

Piotr Szylar

Spis treści

[1. Cel. 1](#_Toc10536257)

[2. Odbiorca. 1](#_Toc10536258)

[3. Interesariusze. 1](#_Toc10536259)

[4. Instalacja. 1](#_Toc10536260)

[5. Opis użytych technologii- backend. 1](#_Toc10536261)

[6. Dokumentacja metod. 2](#_Toc10536262)

[6.1. CommentController. 2](#_Toc10536263)

[6.2. CourseController. 3](#_Toc10536264)

[6.3. CourseRealizationController. 4](#_Toc10536265)

[6.4. FileController 5](#_Toc10536266)

[6.5. GlobalController. 5](#_Toc10536267)

[6.6. HomeController. 5](#_Toc10536268)

[6.7. IndexController. 6](#_Toc10536269)

[6.8. LoginController. 6](#_Toc10536270)

[6.9. SchoolController. 7](#_Toc10536271)

[6.10. SignUpController. 7](#_Toc10536272)

[6.11. UserController. 7](#_Toc10536273)

[6.12. UserProfileController. 9](#_Toc10536274)

[7. Frontend. 9](#_Toc10536275)

[7.1. AddCourse. 9](#_Toc10536276)

[7.2. AddDepartament. 9](#_Toc10536277)

[7.3. AddLecturer. 9](#_Toc10536278)

[7.4. AddSchool. 9](#_Toc10536279)

[7.5. AddUserProfile. 9](#_Toc10536280)

[7.6. CourseView. 9](#_Toc10536281)

[7.7. EditCourseView. 9](#_Toc10536282)

[7.8. Header. 9](#_Toc10536283)

[7.9. headerStudent. 9](#_Toc10536284)

[7.10. Home. 9](#_Toc10536285)

[7.11. Index. 10](#_Toc10536286)

[7.12. Login. 10](#_Toc10536287)

[7.13. Profile. 10](#_Toc10536288)

[7.14. Registration. 10](#_Toc10536289)

[7.15. SelectionBar. 10](#_Toc10536290)

[7.16. ShowAllCourses. 10](#_Toc10536291)

[7.17. ShowCourses. 10](#_Toc10536292)

[7.18. ShowCoursesStudents. 10](#_Toc10536293)

[7.19. ShowMyCourses. 10](#_Toc10536294)

[7.20. ShowStudentsSignedUpForACourse. 10](#_Toc10536295)

[7.21. SignUp ForACourse. 10](#_Toc10536296)

[8. Baza danych. 10](#_Toc10536297)

[9. Funkcjonalności aplikacji. 11](#_Toc10536298)

# 1. Cel.

Aplikacja mająca służyć do wymiany opinii na temat kursów oraz prowadzących, oraz do dzielenia się wiedzą/materiałami.

# 2. Odbiorca.

Docelowym odbiorcą aplikacji są jednostki edukacyjne (uczelnie, szkoły wyższe).

# 3. Interesariusze.

a) Administracja uczelni wyższych (dziekanat)- dzięki aplikacji, praca dziekanatu zostanie ułatwiona, ponieważ będzie mógł on w czasie rzeczywistym tworzyć poszczególne kursy, przypisywać do nich studentów, a także prowadzących, oraz edytować ich opisy.

b) Prowadzący, nauczyciele- dzięki aplikacji, ich praca będzie ułatwiona, ponieważ będą mogli dodawać studentów do swojego kursu, nadawać niektórym studentom uprawnienia do np. dodawania plików na tak zwaną „tablicę” (stronę kursu), a także edytować najważniejsze dane o kursie (opis, syllabus).

c) Uczniowie- dzięki aplikacji, nabywanie wiedzy przez studentów stanie się łatwiejsze. Jest to platforma, dzięki której mogą wymieniać się komentarzami na temat kursu, jaki realizują, mogą go oceniać w skali 1-5, a także łatwiej pobierać materiały, jakie zostaną przez wykładowcę udostępnione.

d) Administrator- osoba, która nadaje uprawnienia dziekanatu w aplikacji.

# 4. Instalacja.

Pobierając aplikację z GitHuba:

1. Git clone repository.

2. Create a database called "edugate".

3. Start MySql server, start Tomcat server.

# 5. Opis użytych technologii- backend.

Aplikacja od strony backendu używa technologii Spring MVC. Projekt Edugate zawiera 8 paczek plików: config, controller, exception, model, payload, repository, security oraz util.

a) Config

Config zawiera klasy odpowiedzialne za zapewnienie właściwej konfiguracji systemu w celu właściwej współpracy z bazą danych i frontenedem oraz ustawienia autoryzacji logowania użytkownika.

b) Controller

Controller zawiera klasy odpowiedzialne za obsługę zapytań z frontendu. Stanowi API do łączenia się z bazą danych; przy użyciu metod CRUD komunikuje się z bazą. Każde żądanie z frontendu charakteryzuje konkretny ciąg URL, który informuje o akcji jaka ma zostać wykonana. Jest on przechwytywany przy pomocy adnotacji @RequestMapping lub adnotacji dla konkretnych metod HTTP (@GetMappping, @PostMapping, itd.) i kierowany do odpowiedniej klasy-kontrolera, gdzie wykonywana jest właściwa metoda.

c) Model

Model zawiera klasy będące zmapowanymi odpowiednikami encji w bazie danych. Umożliwiają konwersję danych z bazy (rekordów) na konkretne, odpowiadające im, obiekty w javie.

d) Repository

Repository zawiera interfejsy używane przez Spring do obsługi operacji CRUD w bazie danych. Definiowane za pomocą adnotacji @Repository oraz rozszerzające interfejs JpaRepository domyślnie pozwalają na realizacje podstawowych i szablonowych operacji CRUD, można jednak definiować nagłówki własnych metod w celu wykonywania skonkretyzowanych operacji czytania, zapisu, usuwania i edycji w bazie.

e) Security

Security zawiera klasy obsługujące poprawną autoryzację użytkownika.

f) Controllers

Controllers zawiera wszystkie klasy, jakie obsługują operacje CRUDowe w aplikacji.

# 6. Dokumentacja metod.

Poniższe spis zawiera wszystkie metody wraz z objaśnieniem, do czego służą, z folderu „Controllers”, jakie używane są przez aplikację.

## 6.1. CommentController.

public ModelAndView commentslistLink() {

return mv;

}

Metoda nie przyjmiuje żadnych parametrów.

Zwraca ModelAndView, a w nim listę wszystkich komentarzy.

## 6.2. CourseController.

public ModelAndView courseslistLink() {

return mv;

}

Metoda nie przyjmiuje żadnych parametrów.

Zwraca ModelAndView, a w nim listę wszystkich kursów.

public ModelAndView addSchool(String name, int semester, int level, String note, String about) {

return mv;

}

Metoda przyjmuje w parametrach: nazwę uczelni, semestr, poziom studiów (I/II stopień), notatkę od prowadzącego oraz opis kursów.

Zwraca ModelAndView, a w nim nowy kurs.

public ModelAndView showCourseView(int IDCourseRealization) {

List<UserProfile> listOfUserProfiles = userProfileRepository.findAll();

List<Comment> listOfComments = commentRepository.findAll();

List<File> listOfFiles = fileRepository.findAll();

mv.addObject("commentList", listOfComments);

mv.addObject("userProfileList", listOfUserProfiles);

mv.addObject("fileList", listOfFiles);

mv.addObject("currentCourseRealization", currentCourseRealization);

return mv;

}

Metoda przyjmuje jako parametr ID realizacji kursu.

Zwraca ModelAndView, a w nim: lista komentarzy, profili użytkowników, plików oraz obecną realizację kursu.

## 6.3. CourseRealizationController.

public ModelAndView homeLink(HttpServletRequest request) {

System.out.println("signupforacourseLink");

ModelAndView mv = new ModelAndView();

mv.setViewName("signupforacourse");

List<CourseRealization> listOfCourseRealizations = courseRealizationRepository.findAll();

Map<CourseRealization, String> courseRealizations = new HashMap<>();

for(CourseRealization courseRealization : listOfCourseRealizations) {

courseRealizations.put(courseRealization, courseRepository.findById((int)courseRealization.getIdcourse()).get().getName());

}

mv.addObject("courseRealizations", courseRealizations);

mv.addObject("confirmation", request.getParameter("confirmation"));

return mv;

}

UZUPEŁNIĆ

public ModelAndView signUpForACourse(int IDUser, int IDCourseRealization) {

userCourseRepository.save(new UserCourse(IDUser, IDCourseRealization));

ModelAndView mv = new ModelAndView("redirect:/signupforacourseLink");

mv.addObject("confirmation", "Dodano do grupy!");

return mv;

}

W parametrach metoda przyjmuje: ID użytkownika obecnie zalogowanego, oraz ID realizacji kursu, na jaki użytkownik chce się zapisać.

Zwraca ModelAndView, a w nim potwierdzenie dodania do grupy.

6.4. FileController.

public ModelAndView filesListLink() {

return mv;

}

Metoda nie przyjmuje żadnych parametrów. Zwraca ModelAneView, a w nim listę wszystkich plików.

## 6.5. GlobalController.

public Users getLoginUsers() {

return loginUsers;

}

Metoda nie przyjmuje żadnych parametrów. Zwraca aktualnie zalogowanego użytkownika.

## 6.6. HomeController.

public String homeLink() {

return "home";

}

Metoda nie przyjmuje żadnych parametrów. Metoda zwraca JSP “Home”.

public String profileLink() {

return "profile";

}

Metoda nie przyjmuje żadnych parametrów. Metoda zwraca JSP “Profil”.

public String addDepartmentLink() {

return "adddepartment";

}

Metoda nie przyjmuje żadnych parametrów. Metoda zwraca JSP “AddDepartament”.

public String addSchoolLink() {

return "addschool";

}

Metoda nie przyjmuje żadnych parametrów. Metoda zwraca JSP “AddSchool”.

public String addCourseLink() {

return "addcourse";

}

Metoda nie przyjmuje żadnych parametrów. Metoda zwraca JSP “AddCourse”.

## 6.7. IndexController.

public String registrationLink() {

return "signup";

}

Metoda nie przyjmuje żadnych parametrów. Metoda zwraca JSP “SignUp”.

public String loginLink() {

return "login";

}

Metoda nie przyjmuje żadnych parametrów. Metoda zwraca JSP “Login”.

## 6.8. LoginController.

public String registrationLink() {

return "home";

}

Metoda nie przyjmuje żadnych parametrów. Po zalogowaniu, przekierowuje na „Home”.

## 6.9. SchoolController.

public ModelAndView addSchool(String name, String about, String description) {

return mv;

}

Metoda w parametrach przyjmuje: nazwę uczelni, opis uczelni (o uczelni), oraz krótki opis. Zwraca ModelAndView, a w nim nowo dodaną do bazy danych szkołę.

## 6.10. SignUpController.

public String registrationLink() {

return "index";

}

Metoda nie przyjmuje żadnych parametrów.

## 6.11. UserController.

public String registration(Model model) {

model.addAttribute("userForm", new Users());

return "registration";

}

UZUPEŁNIĆ

public String registration(@ModelAttribute("userForm") Users userForm, BindingResult bindingResult) {

userValidator.validate(userForm, bindingResult);

if (bindingResult.hasErrors()) {

return "registration";

}

userService.save(userForm);

securityService.autoLogin(userForm.getLogin(), userForm.getPasswordConfirm());

return "redirect:/home";

}

UZUPEŁNIĆ

public String login(Model model, String error, String logout) {

if (error != null)

model.addAttribute("error", "Your username and password is invalid.");

if (logout != null)

model.addAttribute("message", "You have been logged out successfully.");

return "login";

}

UZUPEŁNIĆ

## 6.12. UserProfileController.

public ModelAndView userProfileListLink() {

return mv;

}

Metoda nie przyjmuje żadnych parametrów. Zwraca listę wszystkich profili użytkowników.

# 7. Frontend.

Na frontendzie zastosowna została technologia JSP. Poniżej znajduje się lista wszystkich widoków, jakich używa aplikacja, wraz z wyjaśnieniem, do czego służą.

## 7.1. AddCourse.

Widok odpowiadający za dodanie realizacji kursu do bazy danych.

## 7.2. AddDepartament.

Widok odpowiadający za dodanie wydziału do bazy danych.

## 7.3. AddLecturer.

Widok odpowiadający za dodanie wykładowcy do bazy danych.

## 7.4. AddSchool.

Widok odpowiadający za dodanie uczelni do bazy danych.

## 7.5. AddUserProfile.

Widok odpowiadający za dodanie proflu użytkownika do bazy danych.

## 7.6. CourseView.

Widok odpowiadający za wyświetlenie strony danej realizacji kursu. Na tej stronie znajduje się miejsce na dyskusję na temat kursu(komentarze), są też udowstępniane pliki. Strona umożliwia ocenienie kursu, a także przeczytanie najważniejszych informacji, takich jak: opis i prowadzący.

## 7.7. EditCourseView.

Widok odpowiadający za edycję danych znajdujących się w nagłówku strony CourseView- informacji dotyczących wykładowcy, prowadzącego daną realizację kursu, oraz opisu.

## 7.8. Header.

Menu (pasek nawigazyjny) znajdujące się w nagłówku.

## 7.9. headerStudent.

Menu (pasek nawigazyjny) znajdujące się w nagłówku, używany w widoku dla studenta.

## 7.10. Home.

Strona główna.

## 7.11. Index.

## 7.12. Login.

Widok umożliwiający zalogowanie się do aplikacji.

## 7.13. Profile.

Widok pokazujący profil aktualnie zalogowanego się użytkownika.

## 7.14. Registration.

Widok umożliwiający zarejestrowanie się w serwisie.

## 7.15. SelectionBar.

Widok, w którym znajduje się takz wane “drzewko”, boczny pasek nawigacyjny.

## 7.16. ShowAllCourses.

## 7.17. ShowCourses.

## 7.18. ShowCoursesStudents.

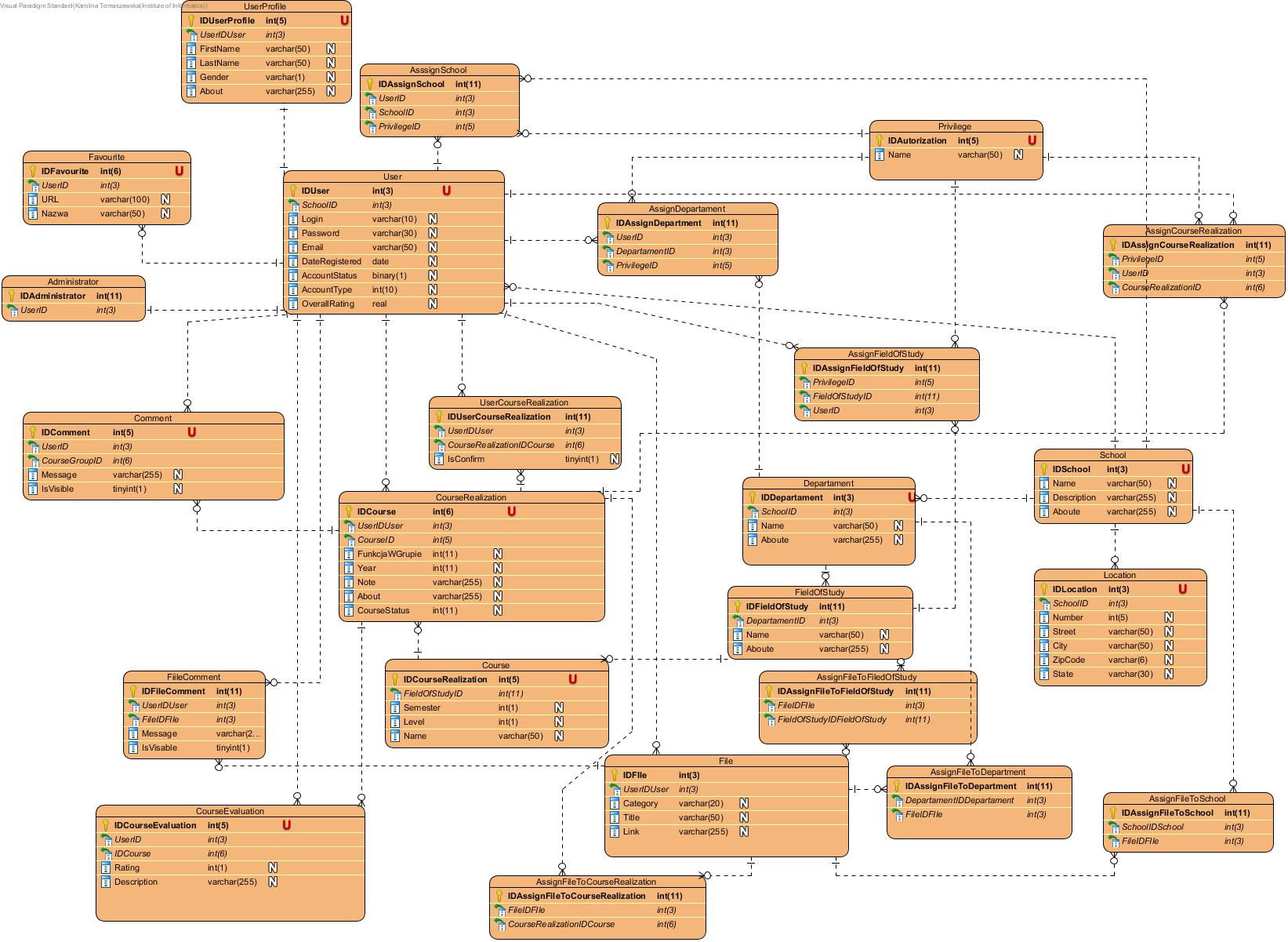
## 7.19. ShowMyCourses.

## 7.20. ShowStudentsSignedUpForACourse.

## 7.21. SignUp ForACourse.

## 8. Baza danych.

Zastosowana została relacyjna baza danych, zrealizowana za pomocą phpMyAdmin. Poniżej, przedstawiony został diagram, na którego podstawie zaimplementowana została baza.



Najważniejsze funkcjonalności takiego modelu przechowywania danych, to elastyczność w nadawaniu przywilejów niektórym użytkownikom (ograniczanie dostępu do niektórych części serwisu), a także możliwość oceny poszczególnych realizacji kursu, a także dodania do nich plików.

# 9. Funkcjonalności aplikacji.

Tu trzeba uzupełnić funkcjonalności aplikacji, wraz ze screenshotami.