СОФИЙСКИ УНИВЕРСИТЕТ
„СВ. КЛИМЕНТ ОХРИДСКИ“

Проект
по
Проектиране и интегриране на софтуерни системи

учебна година 2023/2024

“Elective Courses Platform”

Изготвен от:
Николай Рангелов, 62589
Лиляна Белчева, 62599

Дата: 17.01

Съдържание

История на подобренията

Име	Дата	Причина за промяна	Версия

1. Въведение

Представеният софтуерен продукт е онлайн платформа, която има за цел да улесни процесите, свързани с успешното провеждане на кампанията за Записване на избираеми дисциплини във Факултета по математика и информатика (ФМИ) към Софийския университет "Св.Климент Охридски".

Разработката на програмния продукт е организирана съобразно нуждите на целевата група, състояща се от преподаватели и студенти. Разполага с функционалности, които превръщат задължителното достигане на необходимия брой кредити в приятно изживяване.

Продуктът поощрява развитието на образованието, обучението и ученето както във факултета, така и в университета. Подобрява качеството на преподавателската дейност и допринася за престижа на университета.

2. Дизайн

Elective Courses Platform е продукт, който се очаква да работи съвместно с електронната система СУСИ. Продуктът е зависим от данните, подадени от електронната система, като основно се базира на данните на различните студенти и преподаватели.

В частност се нуждае и от ръководството на Факултета по математика и информатика, като трябва да има достъп до правилника за избираеми дисциплини.

Дизайнът на системата включва разделяне на отговорностите между преподавателите и студентите, като им предоставя специфични функционалности за по-ефективно управление на процеса на записване. Системата е проектирана да бъде лесна за използване и да подпомага потребителите в извършването на задачи свързани с избора на избираеми дисциплини.

Функционалности, зависещи единствено от платформата:

- *Опция за log in с данните от суси - имейл и парола;*

Функционалности на преподаватели:

Всеки преподавател във ФМИ, който води избираеми дисциплини, трябва да има създаден акаунт в системата. За описването на избираемите дисциплини е нужно да се придържа към определен стандарт, създаден от разработчиците на платформата. Преподавателите имат достъп до определен брой функционалности, различни от тези на студента.

- *Функционалност за добавяне на избираема ;*
- *Функционалност за преглед списъка с всички избираеми от съответния преподавател;*
- *Функционалност за преглед на профила;*

Функционалности на студенти:

Целият продукт е ориентиран към студента и неговото улесняване, затова той има и най-много функционалности. Всеки един студент във ФМИ трябва да има акаунт в системата, за да може да завърши успешно. Очаква се да ползват платформата около 4 пъти годишно - в началото на кампаниите и в края им - за записване.

- *Функционалност за записване на избираеми;*
- *Функционалност за отписване на избираеми;*

- *Функционалност за достъп до виртуална таблица, покриваща задължителните кредити, нужни за съответната специалност на студента;*
- *Функционалност за достъп до списъка с всички избираеми;*
- *Функционалност за достъп до записаните от него избираеми;*
- *Функционалност за преглед на профила;*

3. Разработка

В разработката се използва Java Spring Framework с инструмента за управление на зависимости Maven. За база данни се използва mySQL. Приложението се deploy-ва на Tomcat server.

А) Architecture: client/server архитектура - сървър, на който пазим информация за курсовете, потребителите и т.н.

Б) Data structures: използваме text messages под формата на JSON/form форматиран текст

В) Protocol: комуникацията се извършва чрез HTTP протокол.

Г) Interface: използваме web form-based input/output.

Структурата на проекта представлява:

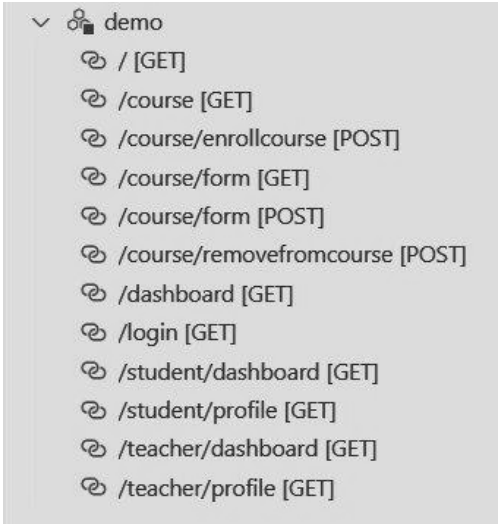
- *SecurityConfig.java -> дефинира настройките за сигурност на приложението, като в случая имаме разграничаване на ролите Преподавател/Студент*
- *controller/ -> контролерите на приложението, отговорни за обработка на HTTP заявките, със съответните им функции*
    - *CourseController.java -> allCourses,showAddCourseForm, addCourse, enrollStudent, removeStudentFromCourse, isEnrolled*
    - *LoginController.java -> loginPage, determineTargetUrl*
    - *StudentController.java -> studentDashboard, showStudentProfile*
    - *TeacherController.java -> teacherDashboard, showTeacherProfile*
- *model/ -> съответно спрямо базата данни*

```
-- Таблица за ролите
CREATE TABLE role (
    id INT PRIMARY KEY,
    name VARCHAR(45) NOT NULL
);

-- Таблица за потребителите (преподаватели и студенти)
CREATE TABLE user (
    id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(45) NOT NULL,
    password VARCHAR(255) NOT NULL,
    email VARCHAR(45) NOT NULL,
    role_id INT,
    FOREIGN KEY (role_id) REFERENCES role(id)
);
```

```sql
-- Таблица за курсовете
CREATE TABLE course (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    type VARCHAR(45) NOT NULL,
    credits INT NOT NULL
);

CREATE TABLE enrollment_type (
    id INT PRIMARY KEY,
    name VARCHAR(255) NOT NULL
);


-- Таблица за записванията на студенти в курсове
CREATE TABLE enrollment (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT,
    course_id INT,
    enrollment_type_id INT,
    FOREIGN KEY (user_id) REFERENCES user(id),
    FOREIGN KEY (course_id) REFERENCES course(id),
    FOREIGN KEY (enrollment_type_id) REFERENCES enrollment_type(id)
);
```

- *repository/ -> взаимодействие с базата данни*
- *service/ -> UserService, представляващ съответно loadUserByUsername*
- *util/ -> Result, в който се намира логиката за таблица с критерии (таблица, която показва съответно колко кредита и избираеми дисциплини остават на съответния студент, за да завърши успешно)*

Визуализация на mapping-ите:



Визуализация на йерархията:

└── demo/

├── mvn

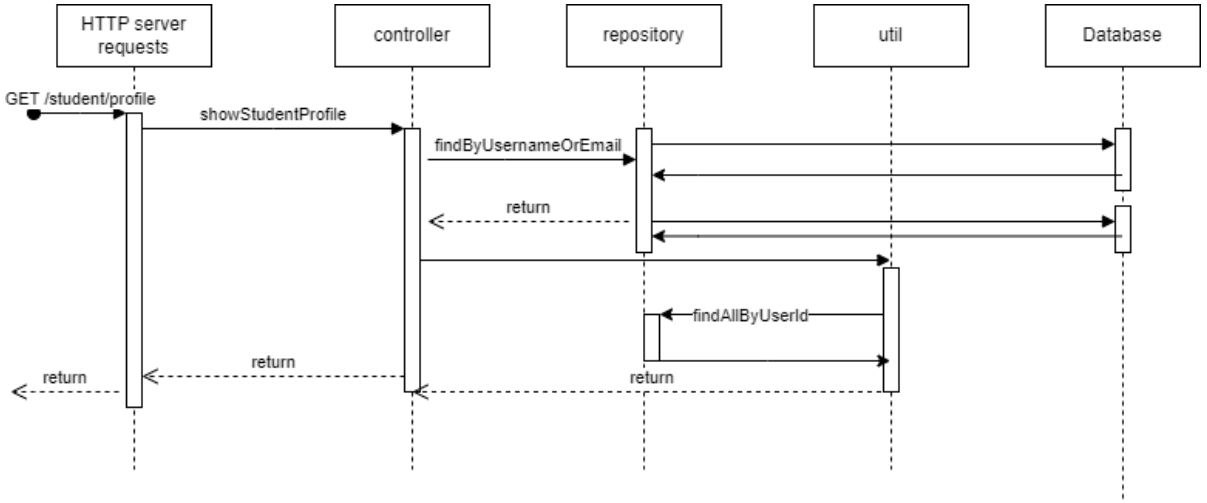
```
├── .vscode
└── src/
    └── main/
        ├── java/com/example/demo/
        │   ├── config/
        │   │   └── SecurityConfig.java
        │   ├── controller/
        │   │   ├── CourseController.java
        │   │   ├── HomeController.java
        │   │   ├── LoginController.java
        │   │   ├── StudentController.java
        │   │   └── TeacherController.java
        │   ├── model/
        │   │   ├── Course.java
        │   │   ├── Enrollment.java
        │   │   ├── EnrollmentType.java
        │   │   ├── Role.java
        │   │   └── User.java
        │   ├── repository/
        │   │   ├── CourseRepository.java
        │   │   ├── EnrollmentRepository.java
        │   │   ├── EnrollmentTypeRepository.java
        │   │   ├── RoleRepository.java
        │   │   └── UserRepository.java
        │   ├── service/
        │   │   └── UserService.java
        │   ├── util/
        │   │   └── Result.java
        │   └── DemoApplication.java
        ├── resources/
        │   └── templates/
        │       ├── course/
        │       │   ├── allcourses.html
        │       │   └── form.html
        │       ├── student/
        │       │   ├── dashboard.html
        │       │   └── profile.html
```

```
│       ├── teacher/
│       │   ├── dashboard.html
│       │   └── profile.html
│       ├── login.html
│       └── application.properties
└── test
```

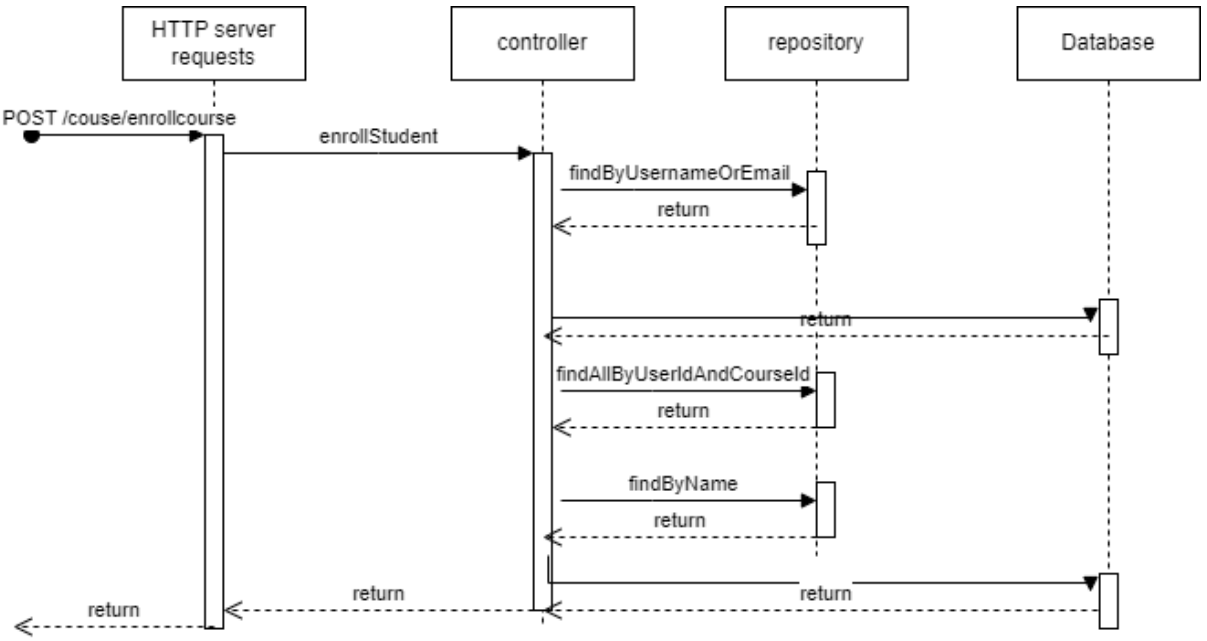
Използването на съответни класове и интерфейси, като например Course, Enrollment и User, предоставят ясно дефинирани модели и функционалности, които обслужват нуждите на системата. Това позволява и разширяемост и поддръжка.

Диаграми:

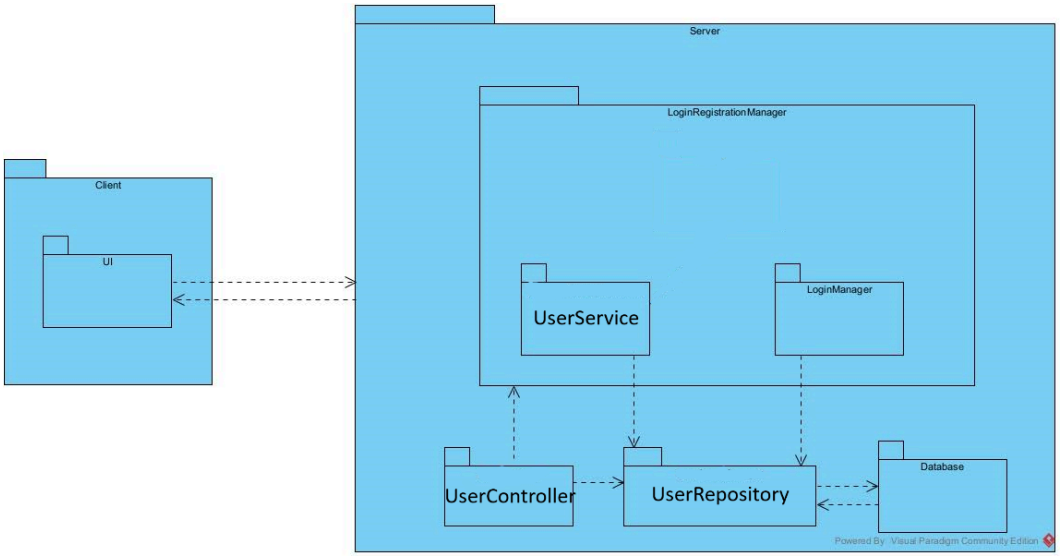
- *Sequence Diagram - Student profile*



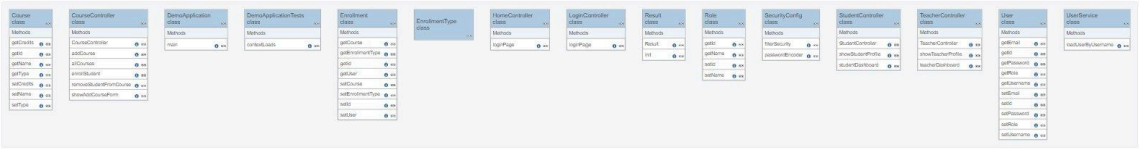
- *Sequence Diagram - Course enrollment*



- *Package Diagram - Компоненти, нужни за влизане*



- *Class Diagram*



4. Внедряване

Създава се база данни electivecoursesplatform.
В нея се изпълняват db.sql и след това populate.sql.
mvn package създава комбиниран jar на проекта. След като сме го създали, може да го изпълним с java -jar target\demo-0.0.1-SNAPSHOT.jar
Важно е да се спомене, за да се изпълни и компилира съответният проект, е нужно да имаме инсталирани и конфигурирани java и maven.

```
cd $(rootdir)
mvn package
java -jar target\demo-0.0.1-SNAPSHOT.jar
```

5. Тестване

За системата са създадени 2 integration tests и 2 unit tests.

За integration тестовете:

- *В /course/form създаваме курс и проверяваме дали сме се аутентикирали и съответно създали курс*

```java
@SpringBootTest
@AutoConfigureMockMvc
public class CourseControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @MockBean
    private CourseRepository courseRepository;

    @MockBean
    private EnrollmentRepository enrollmentRepository;

    @MockBean
    private UserRepository userRepository;

    @Test
    void testAddCourse() throws Exception {
        mockMvc.perform(post("/course/form")
                .param("name", "New Course TEST")
                .param("type", "Др.")
                .param("credits", "7")
                .with(user("msavov")))
                .andExpect(status().is3xxRedirection())
                .andExpect(redirectedUrl("/dashboard"));
    }
}
```

- *Проверява аутентикацията на студента и препращане му към правилния dashboard.*

```java
@SpringBootTest
@AutoConfigureMockMvc
public class StudentControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @Autowired
    private UserService userService;

    @Test
    void testStudentProfile() throws Exception {

        mockMvc.perform(get("/student/profile").with(user(userService.loadUserByUsername(
"nkrangelov"))))
                .andExpect(status().isOk())
                .andExpect(view().name("student/profile"))
                .andExpect(model().attributeExists("user"))
                .andExpect(model().attributeExists("userRole"))
                .andExpect(model().attributeExists("results"));
    }
}
```

За unit тестовете:

- *За course - тест, който проверява дали се връща правилното view при authentication с user Студент*

```java
    @Test
    void testStudentDashboard() {
        UserDetails mockUserDetails = mock(UserDetails.class);
        when(authentication.getPrincipal()).thenReturn(mockUserDetails);
        when(mockUserDetails.getUsername()).thenReturn("studentUser");

        User mockUser = new User();
        mockUser.setId(1);
        when(userRepository.findByUsernameOrEmail("studentUser", "")).thenReturn(java.util.Optional.of(mockUser));

        String viewName = studentController.studentDashboard(authentication, model);

        assertEquals("student/dashboard", viewName);
        verify(model, times(1)).addAttribute(eq("courses"), any());
    }
```

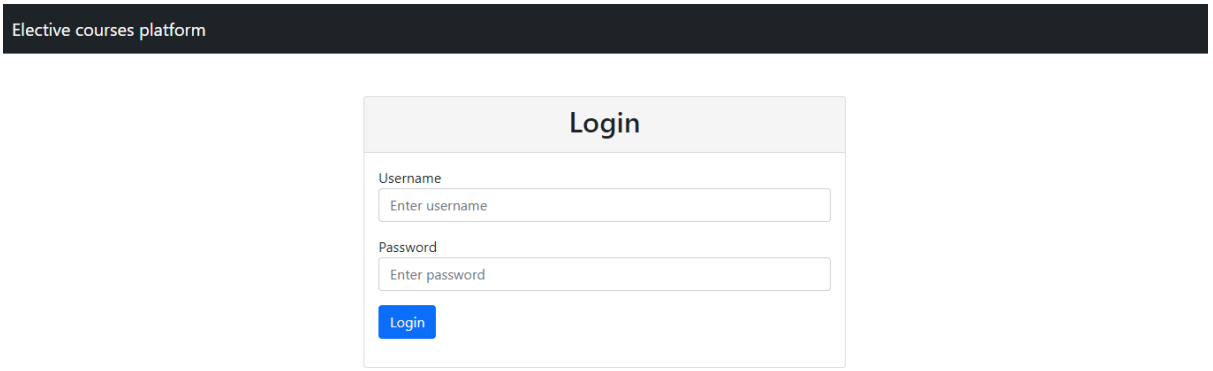
- *За преподавател - тест, който проверява дали се препращаме на правилния изглед след минаване през security config.*

```java
    @Test
    void testTeacherDashboard() {
        UserDetails mockUserDetails = mock(UserDetails.class);
        when(authentication.getPrincipal()).thenReturn(mockUserDetails);
        when(mockUserDetails.getUsername()).thenReturn("teacherUser");

        User mockUser = new User();
        mockUser.setId(1);
        when(userRepository.findByUsernameOrEmail("teacherUser", "")).thenReturn(java.util.Optional.of(mockUser));

        String viewName = teacherController.teacherDashboard(authentication, model);

        assertEquals("teacher/dashboard", viewName);
        verify(model, times(1)).addAttribute(eq("courses"), any());
    }
```

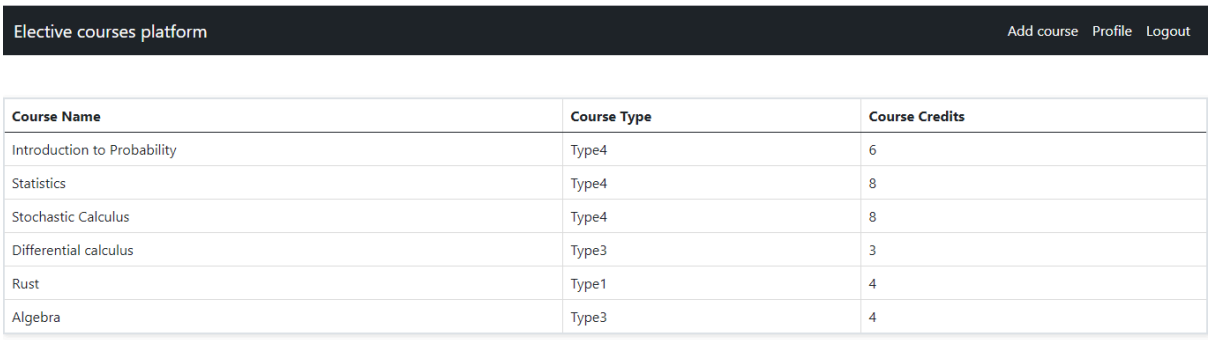
След пускане на тестовете всички минават успешно.

6. Преглед на приложението

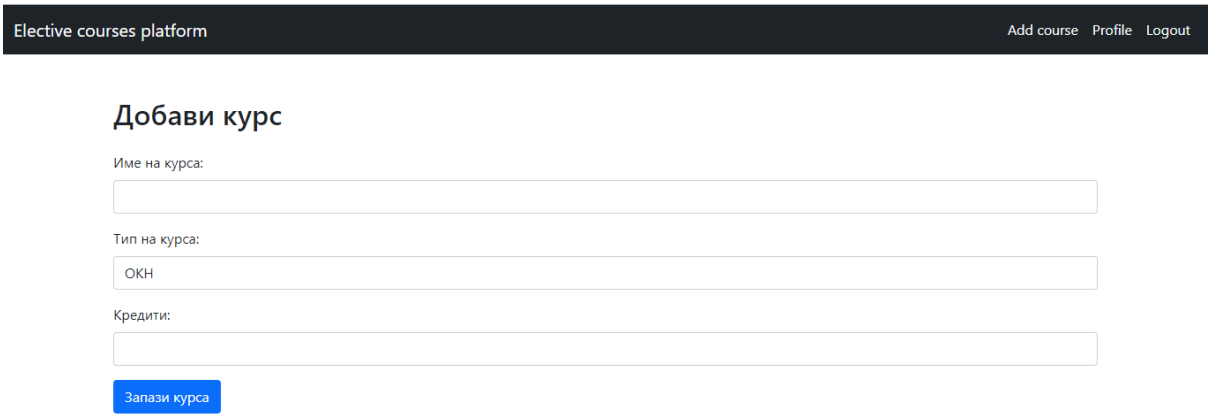
- *Влизане в системата с данните от СУСИ, съответно като Преподавател или Студент*



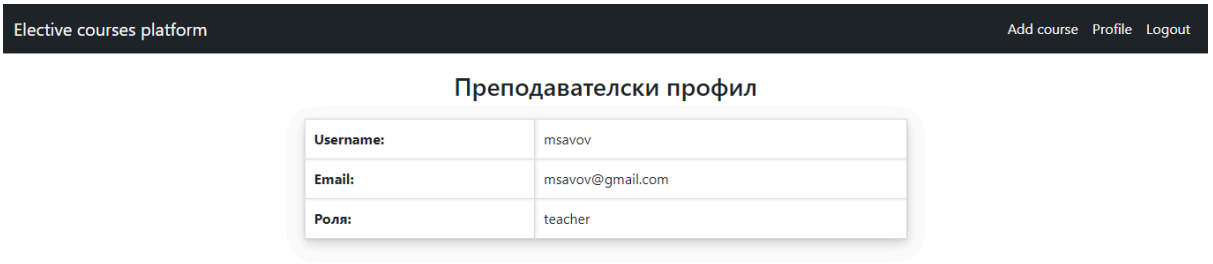
- *Преподавателски изглед: Създадените от него избираемите на началната страница (Elective Courses Platform - бутон за връщане към началната страница)*



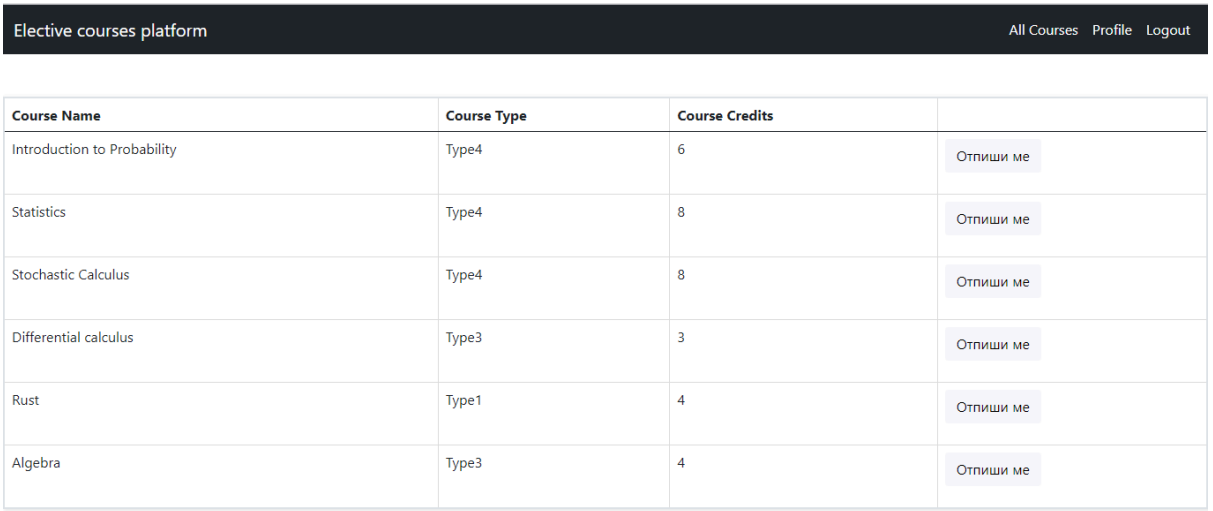
- *Преподавателски изглед: Възможност за добавяне на курс, избор от няколко типа и максимален брой кредити - 10*



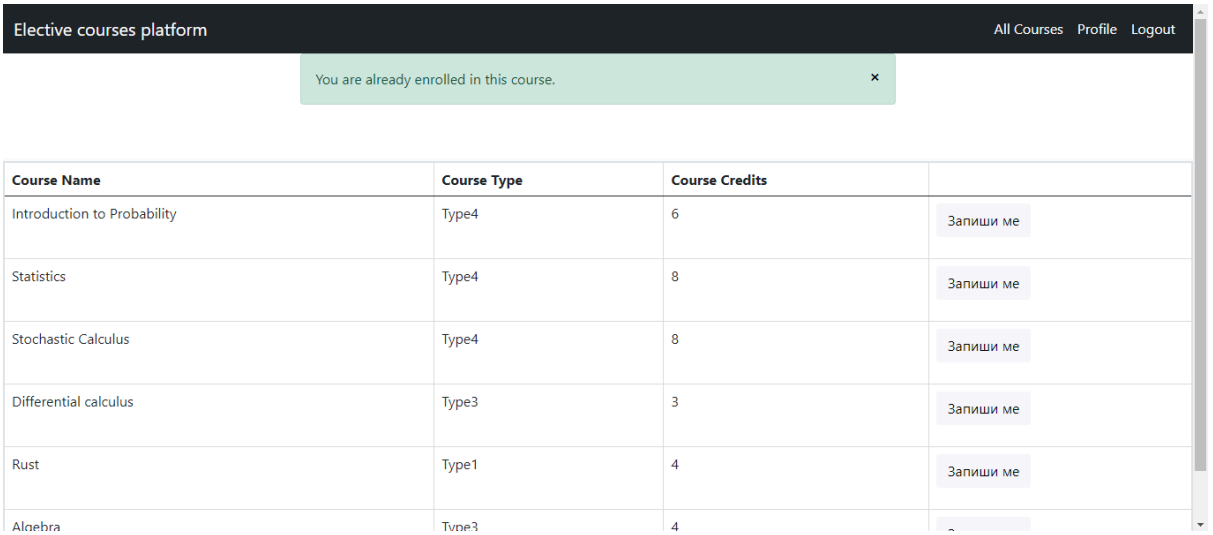
- *Преподавателски изглед: Профил*



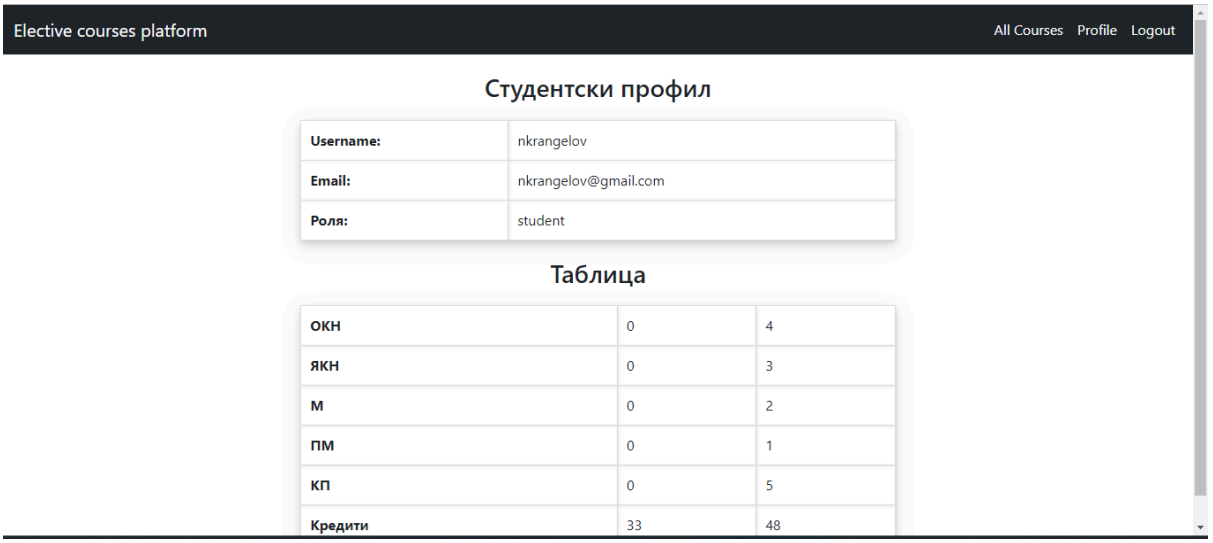
- *Студентски изглед: Записаните от него избираеми на началната страница с възможност за отписване (Elective Courses Platform - бутон за връщане към началната страница)*



- *Студентски изглед: Възможност за преглед на всички избираеми, както и записване. При вече записан избираем - съобщение за уведомяване, че този курс вече е записан*



- *Студентски изглед: Профил*



7. Заключение

С разработката на онлайн платформа за записване на избираеми дисциплини, използвайки Java Spring и Maven, създадохме функционалност, предоставяща значително улеснение на студентите и преподавателите във Факултета по математика и информатика. Проектът е създаден с оглед на ефективното управление на записванията и оптимизирането на процесите в рамките на академичната среда.

Въведението на стандарти като Java Spring и Maven предоставя стабилна основа за разработката на приложението. Изборът на Java Spring осигурява модулна и лесна за разширение архитектура, а интеграцията с Maven облекчава управлението на зависимостите и конфигурацията на проекта. Използването на JUnit за тестване на функционалностите гарантира надеждността и коректността на кода.

Проектът успешно създава баланс между използването на съвременни технологии и лесната поддръжка. Онлайн платформата предоставя необходимите функционалности за студентите и преподавателите, като подобрява качеството на образованието в университетската среда.