Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Владимирский государственный университет имени
Александра Григорьевича и Николая Григорьевича Столетовых»
(ВлГУ)

Институт ИИТР Кафедра ИСПИ

КУРСОВОЙ ПРОЕКТ по дисциплине «Распределенные программные системы» на тему: Проектирование и разработка программной системы информационной системы «Университет»

Выполнил:

студент группы

ПРИ-120 Грачев Д.А.

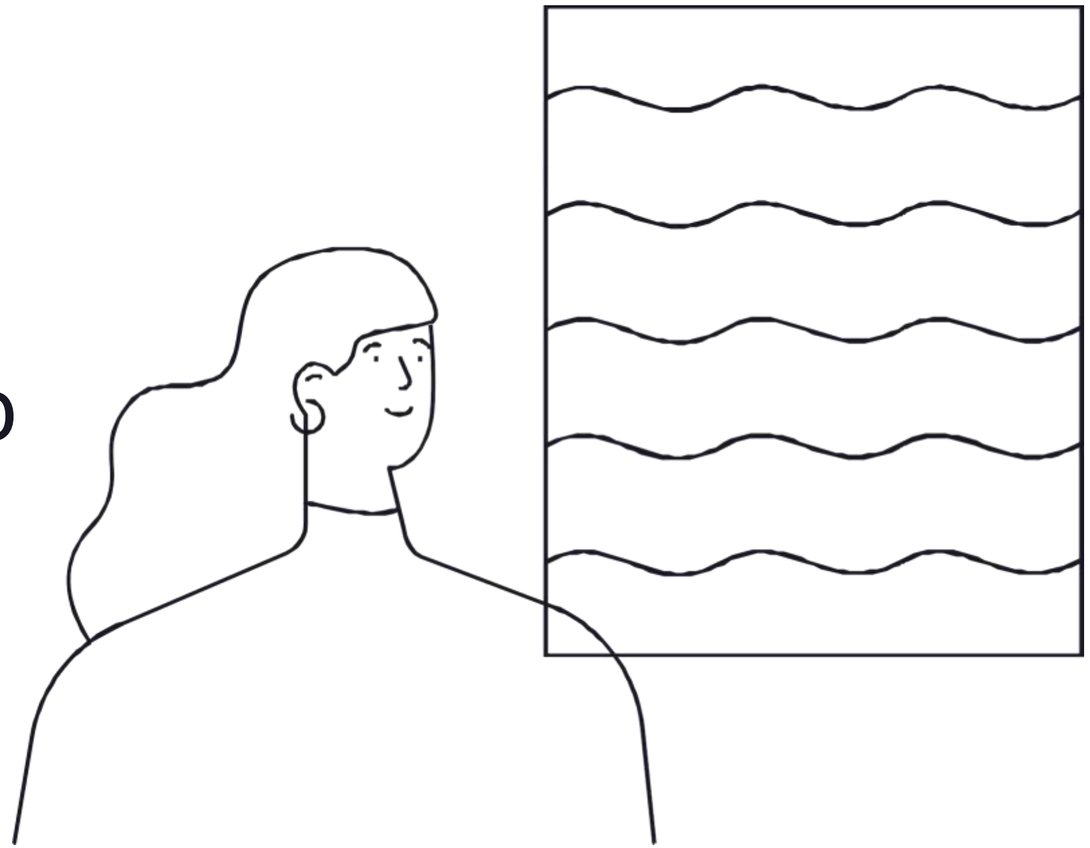
Принял:

доцент кафедры ИСПИ

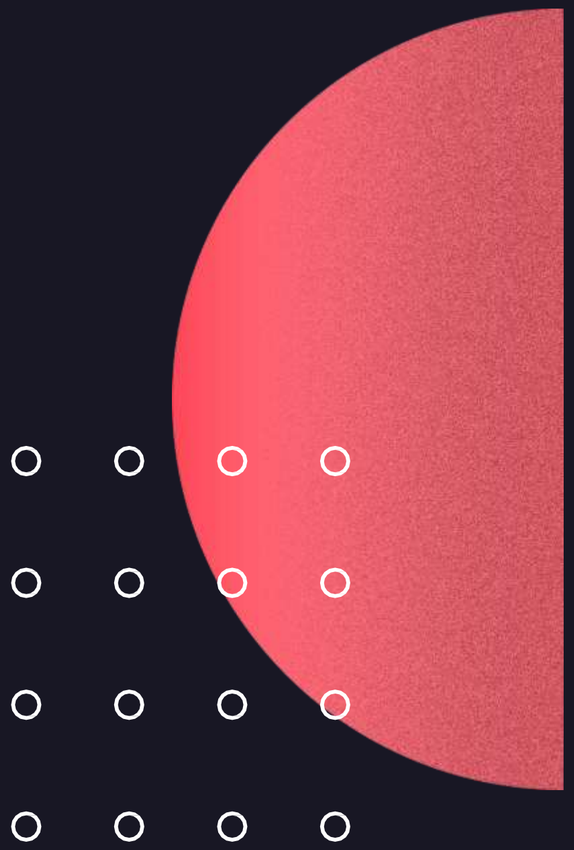
Проскурина Г.В.

Цель работы

Изучение процесса разработки сложных систем на примере программной системы, позволяющей проводить онлайн тестирование студентов, получать обратную связь от пользователей и регистрировать различных пользователей в системе под разными ролями



Описание предметной области

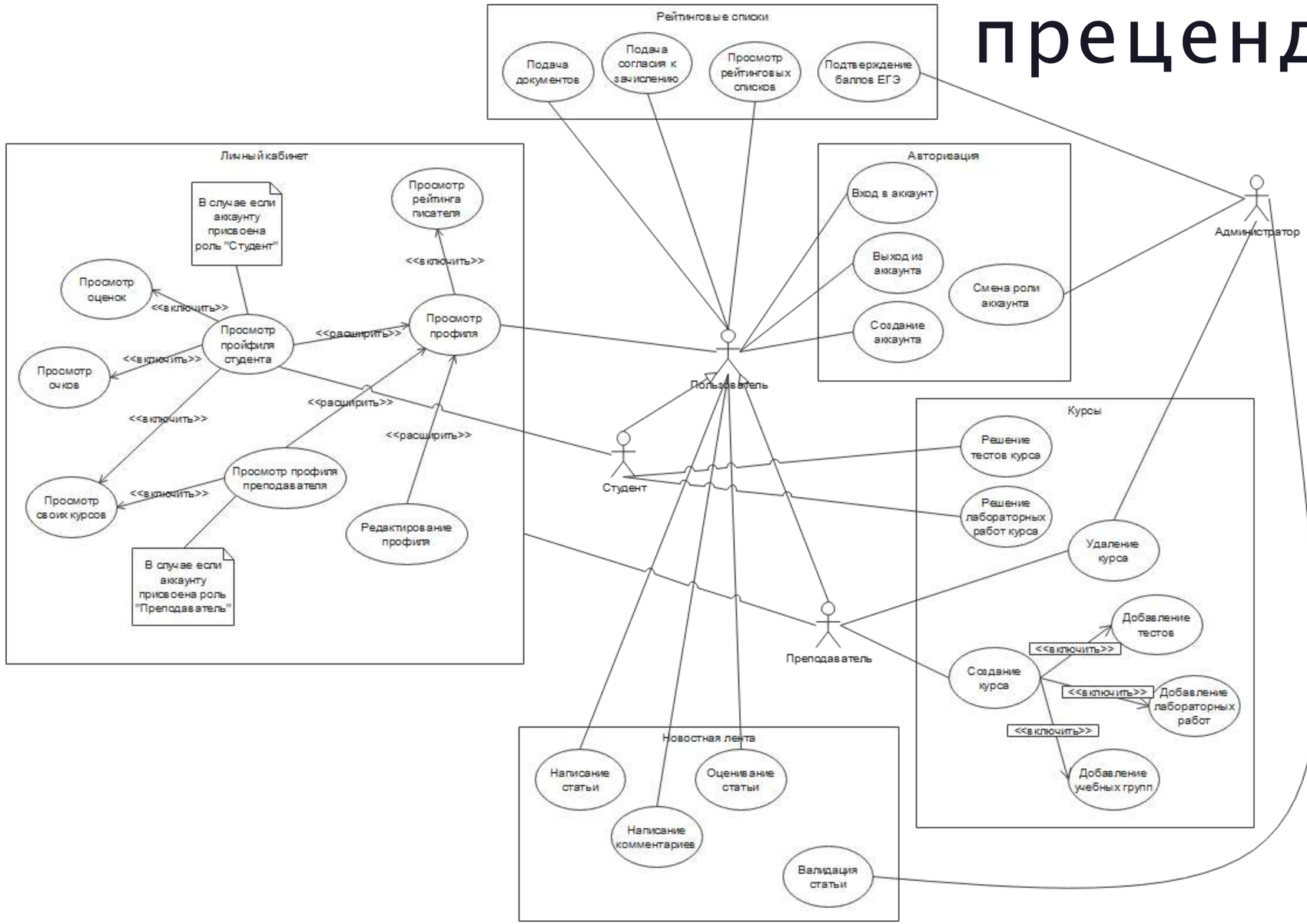


В университете, обучающем студентов и осуществляющем набор абитуриентов, важно иметь программную систему, обеспечивающую:

- **автоматизацию рейтинговых списков абитуриентов**
- **сдачу лабораторных работ в электронной форме и контрольных работ в онлайн формате с автоматическим указанием баллов**
- **новостную ленту с возможностью написания статей любым пользователем программной системы**

ЭТАП ПРОЕКТИРОВАНИЯ

Диаграмма прецендентов



Актеры:

- Пользователь
- Администратор
- Преподаватель
- Студент

Подсистемы:

- Авторизация
- Курсы
- Новости
- Личный кабинет
- Рейтинговые списки

Диаграмма классов

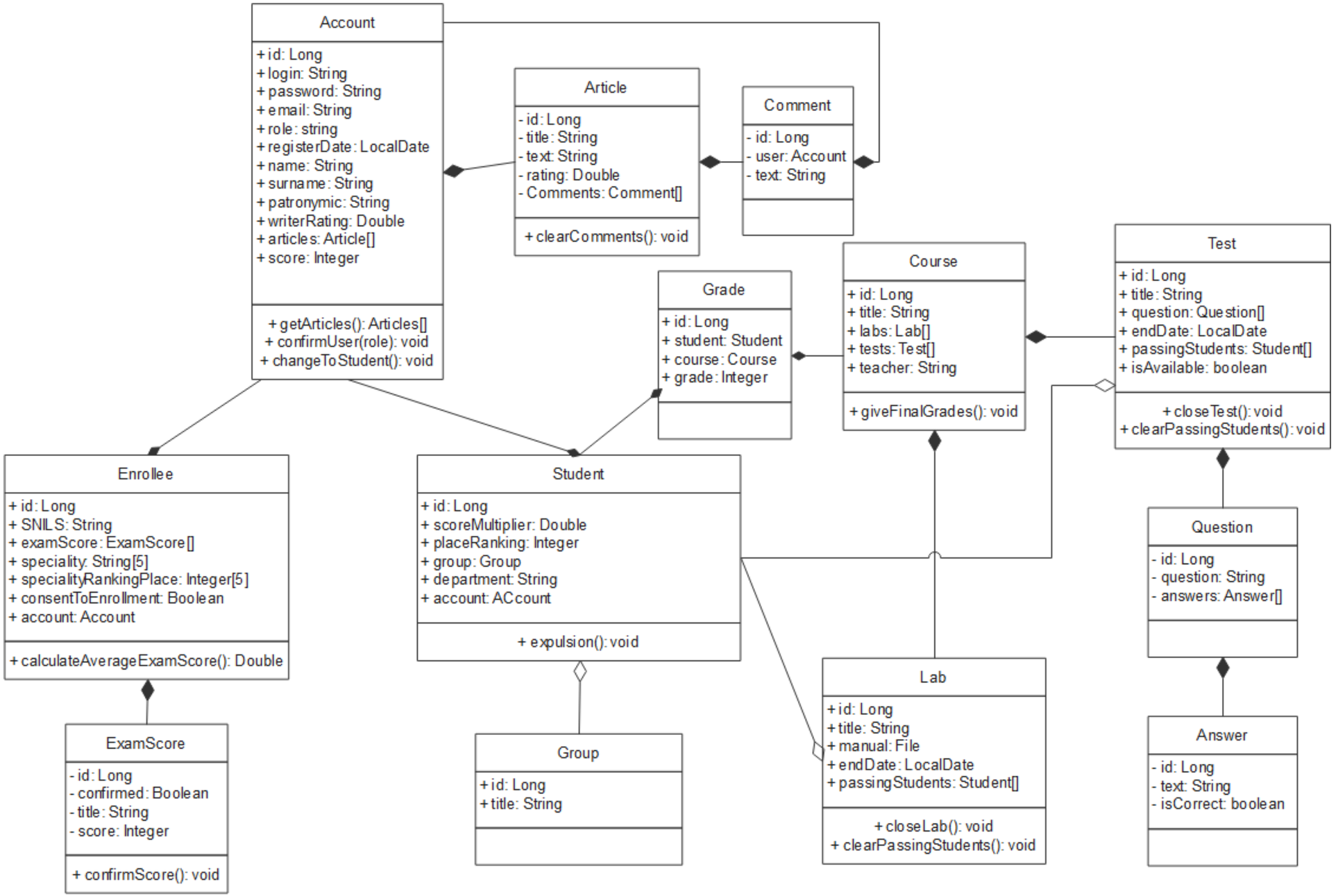
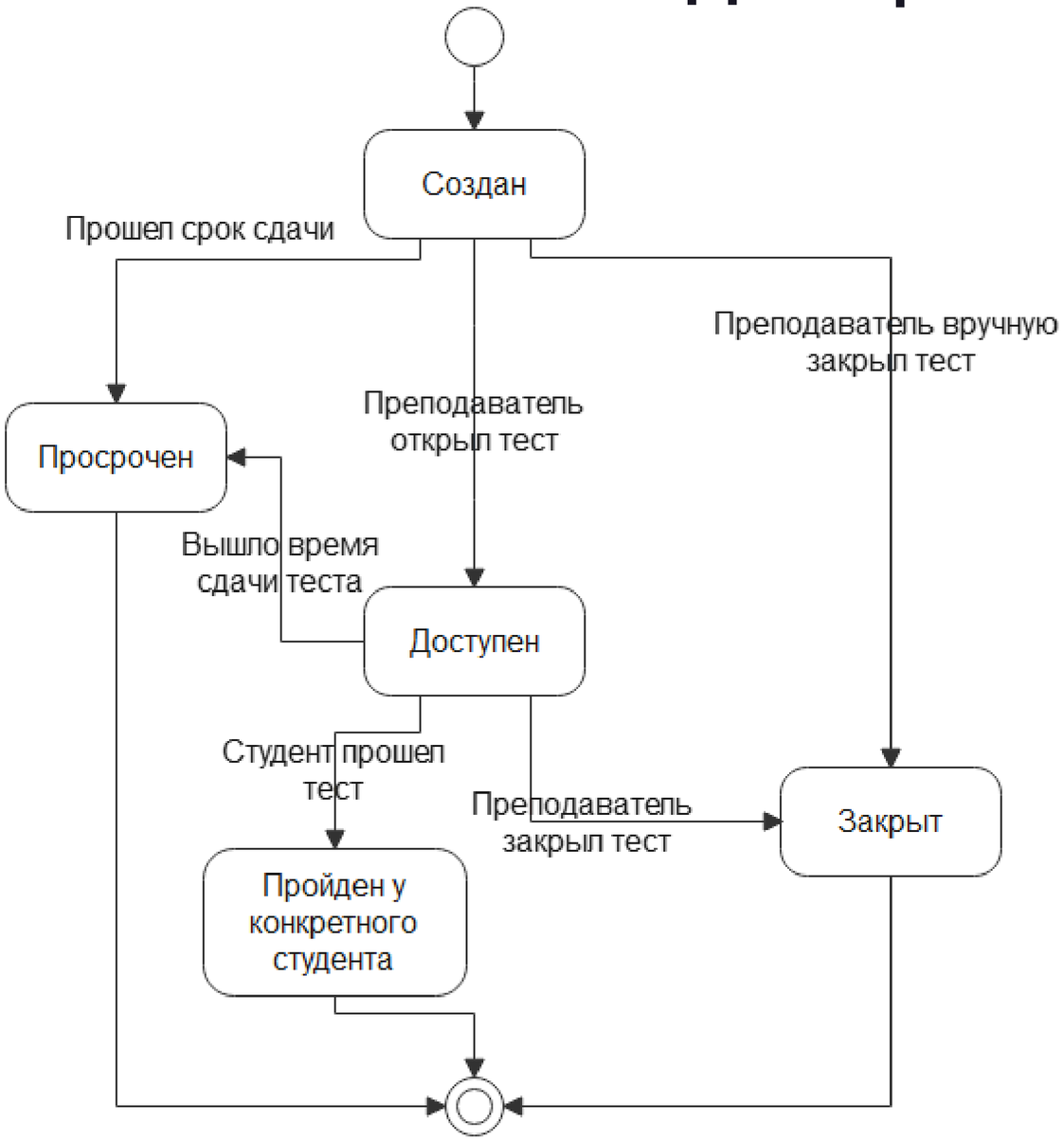


Диаграмма состояний



При создании теста преподаватель может указать дату окончания теста и доступность его для студентов сразу по созданию.

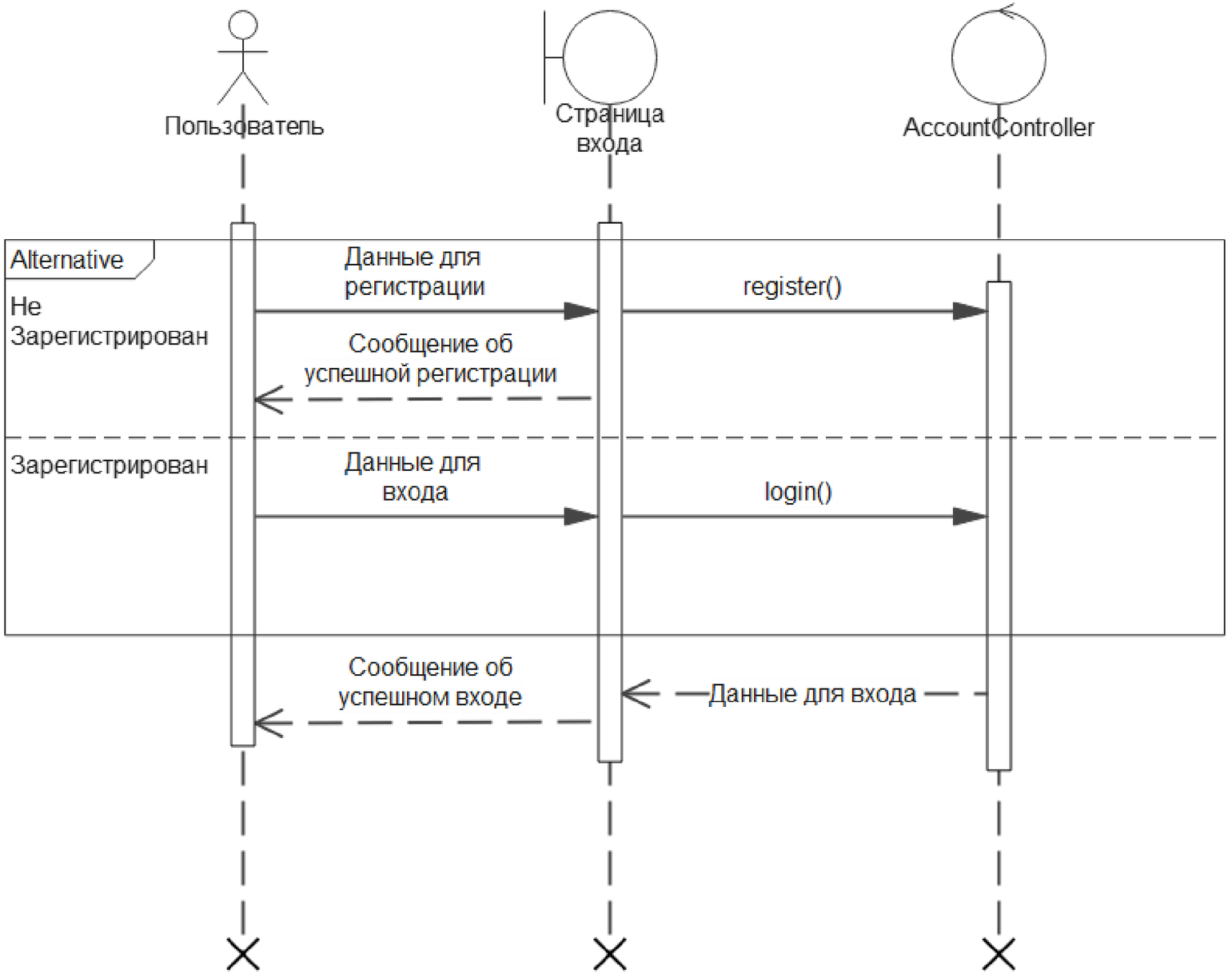
«Просрочен» - прошла дата окончания

«Закрыт» - преподаватель вручную явно установил маркер закрытости

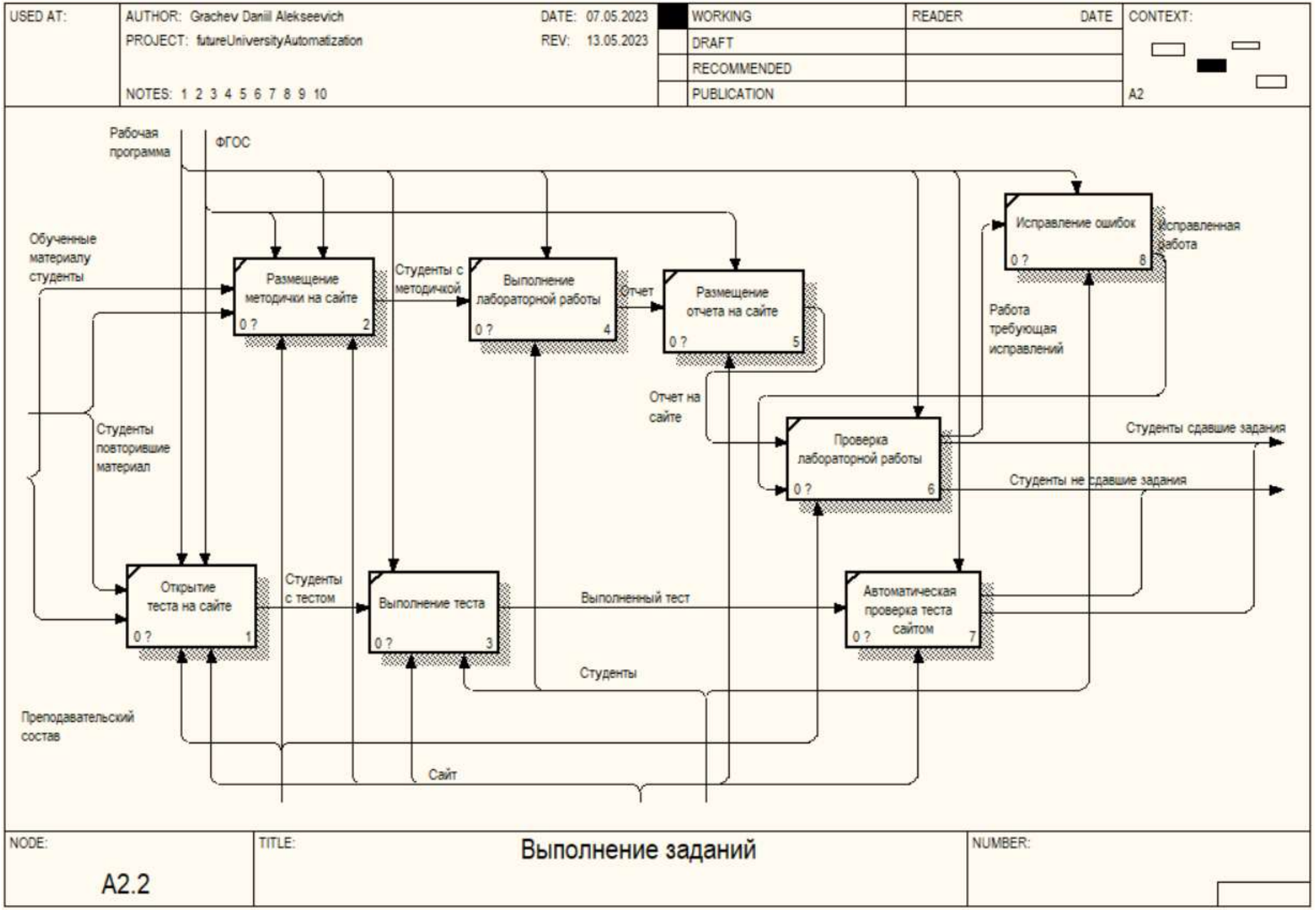
«Доступен» - не прошел срок и не закрыт явно

«Пройден» – маркер для конкретного студента, если он его прошел

Диаграмма последовательностей



IDEF0 диаграмма. «Выполнение заданий»



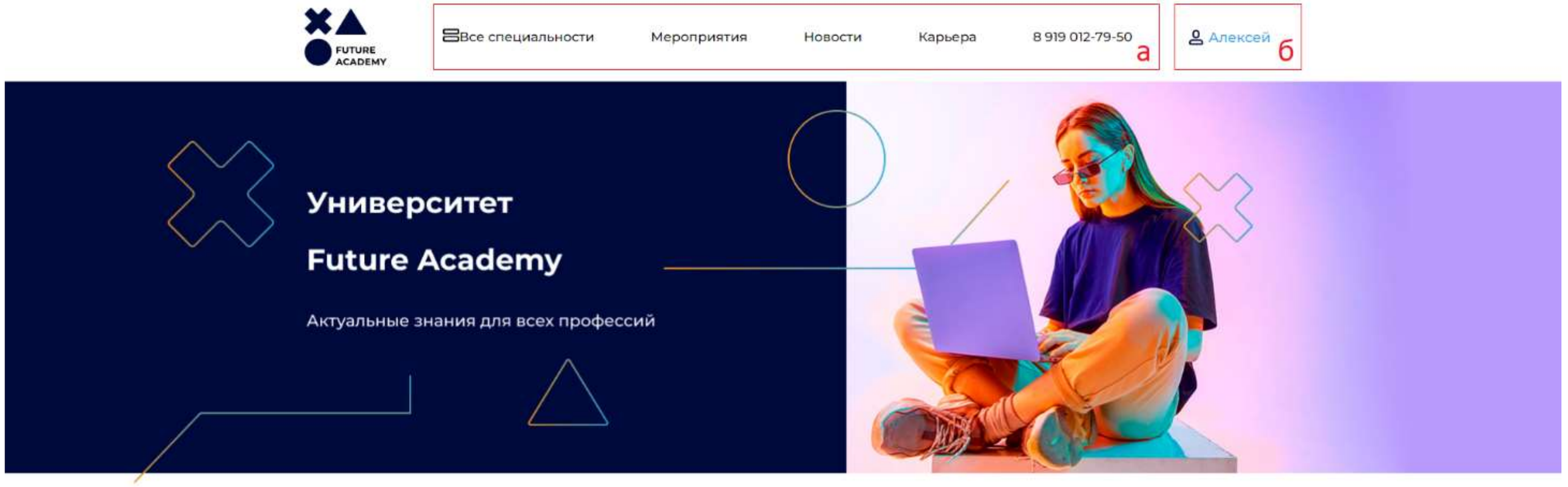
РЕАЛИЗАЦИЯ

Настройка доступа

```
return http
    .authorizeHttpRequests() AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcl
    .requestMatchers(
        ...patterns: "/api/account/register",
        "/api/account/existsemail",
        "/api/account/existslogin"
    ).permitAll()
    .requestMatchers( ...patterns: "/api/account/**").authenticated()

    .requestMatchers( ...patterns: "/api/admin/**").hasAuthority("admin")

    .requestMatchers( ...patterns: "/api/courses/fillcourses").hasAuthority("admin")
    .requestMatchers(
        ...patterns: "/api/courses/addtesttocourse",
        "/api/courses/changetestinfo"
    ).hasAuthority("teacher")
    .requestMatchers( ...patterns: "/api/courses/**") AuthorizeHttpRequestsConfigurer<...>.Auth
        .hasAnyAuthority( ...authorities: "student", "teacher", "admin") AuthorizeHttpRequ

    .requestMatchers(
        ...patterns: "/api/servicesinfo",
        "/api/appeal"
    ).permitAll()

    .requestMatchers( ...patterns: "/api/student/**").hasAuthority("student")

    .requestMatchers( ...patterns: "/**").permitAll()
    .and() HttpSecurity
```

```
.logout(logout -> logout.logoutUrl("/api/account/logout"))

.formLogin() FormLoginConfigurer<HttpSecurity>
.loginProcessingUrl("/api/account/login")
.successHandler((request, response, authentication) -> {
})
.failureHandler((request, response, authentication) -> response.setStatus(400))
.and() HttpSecurity

.exceptionHandling() ExceptionHandlingConfigurer<HttpSecurity>
.authenticationEntryPoint(new Http403ForbiddenEntryPoint())
.and() HttpSecurity

.cors() CorsConfigurer<HttpSecurity>
.and() HttpSecurity

.csrf() CsrfConfigurer<HttpSecurity>
.disable() HttpSecurity
.build();
```

Главная страница



а – навигация
б – метка пользователя
в – информация о университете

Страница профиля

Мой профиль

а

Грачев Алексей

a@a.a

Очки

300

Подробнее

Рабочий стол

Курсы

Настройки

Выйти

б

Общие сведения

Грачев Алексей Александрович

Основная информация

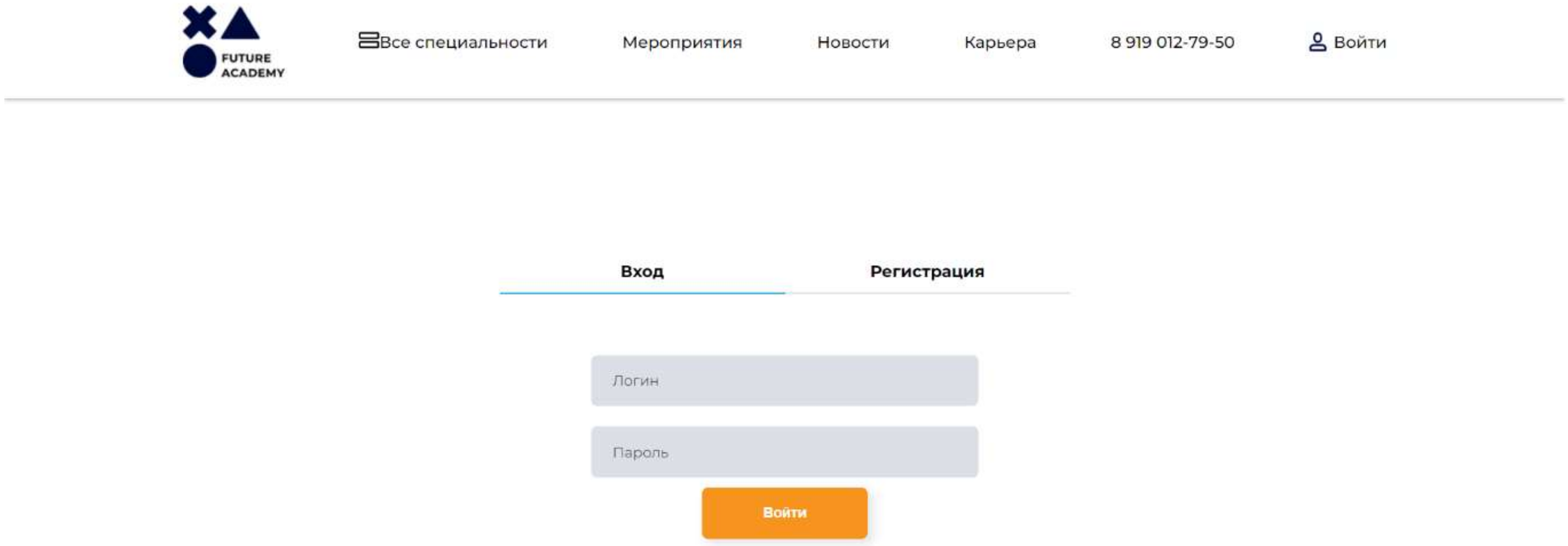
Группа: ПРИ-120

Кафедра: ИСПИ

Почта: a@a.a

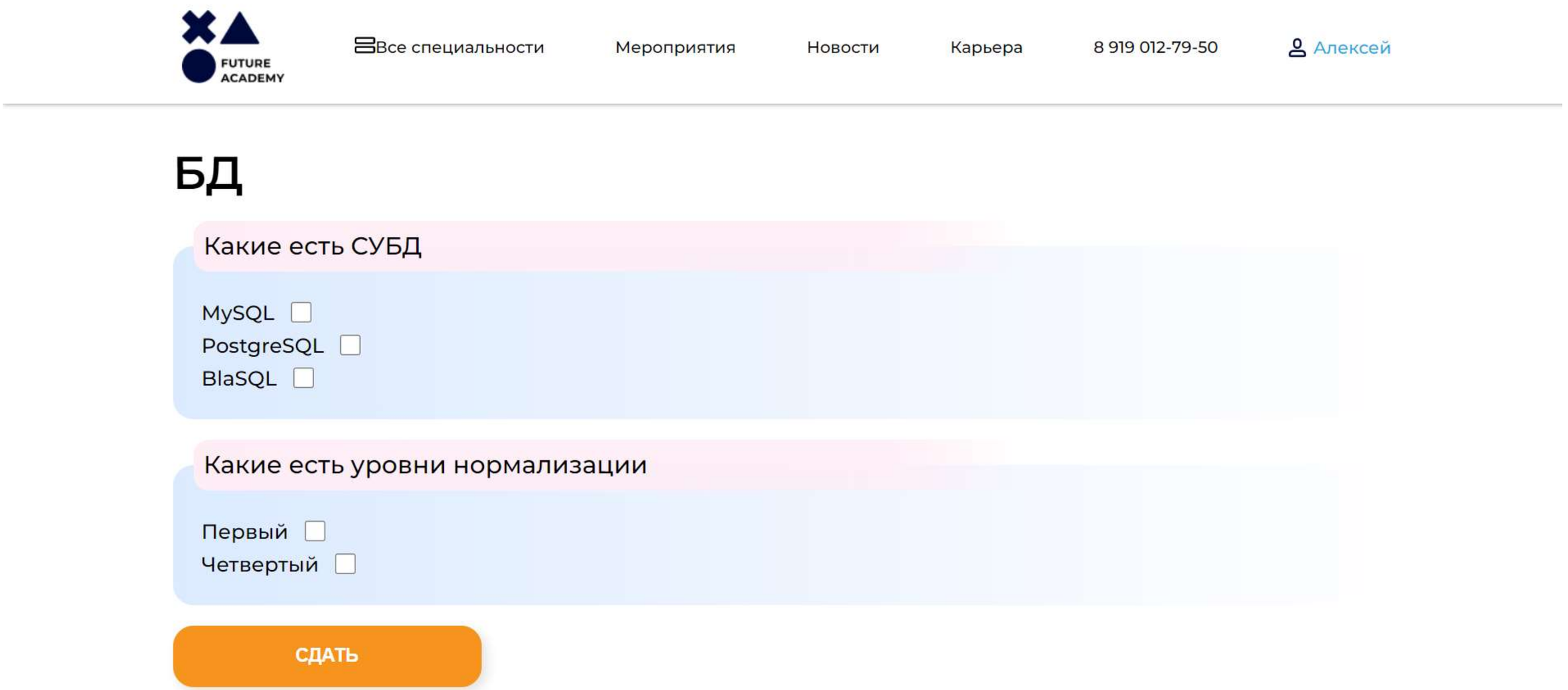
Роль: Студент

Страница входа/регистрации

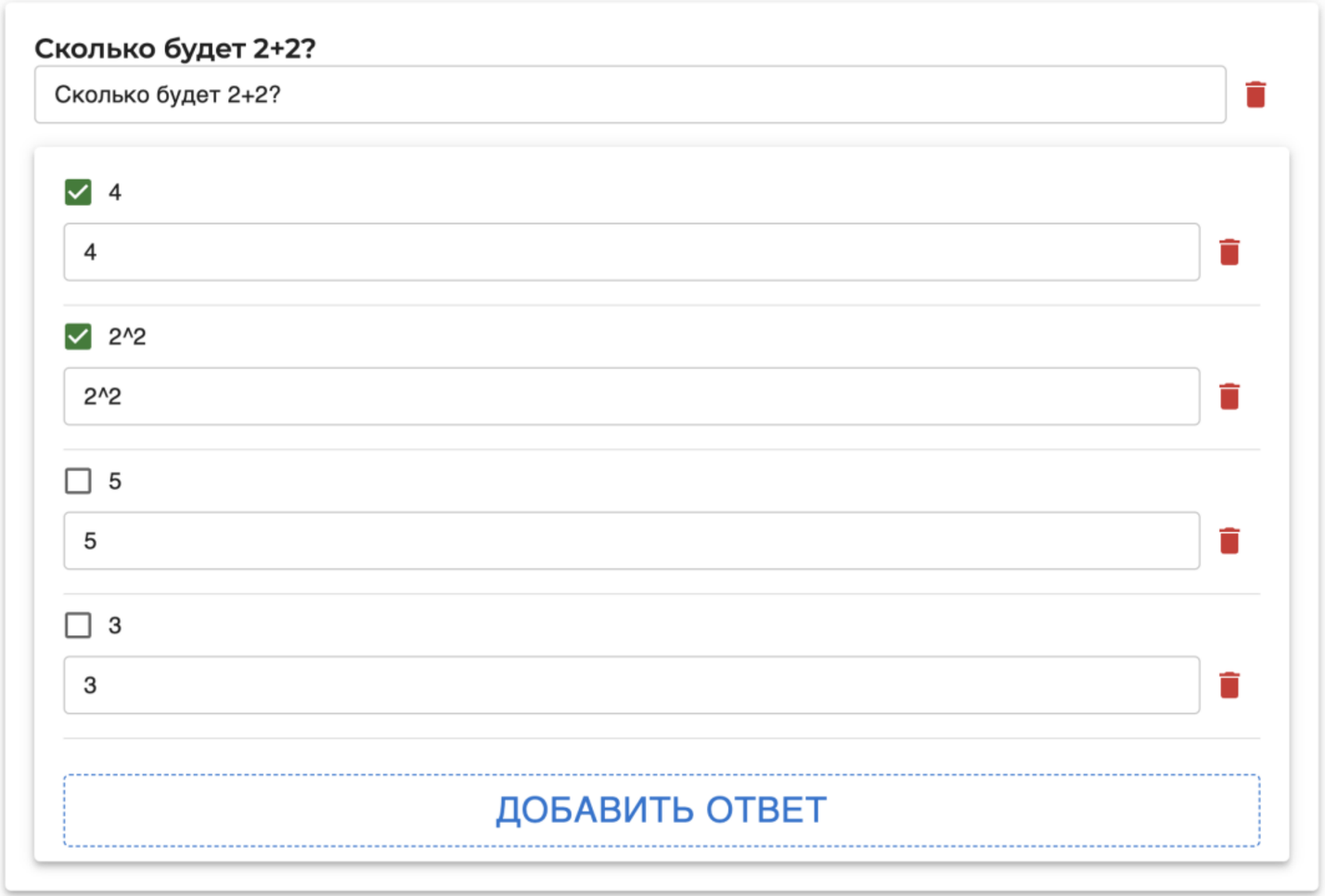


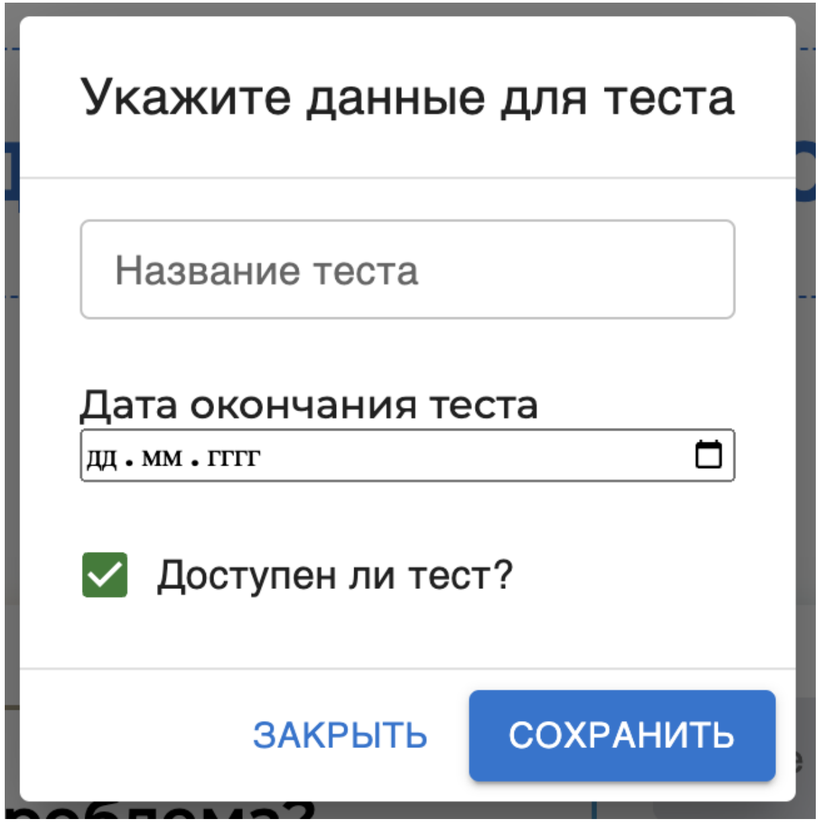
	id	login	password	
1	5	chase	$2a$10$siGGg8ejNaohxZcH7A1rzub…	a
2	7	chase2	$2a$10$3koxHEitBFsP0o2LegvTaOn…	c
3	6	chase1	$2a$10$ARNKgsxdrJoK5z79JD.tNOc…	b

Страница тестов для студента



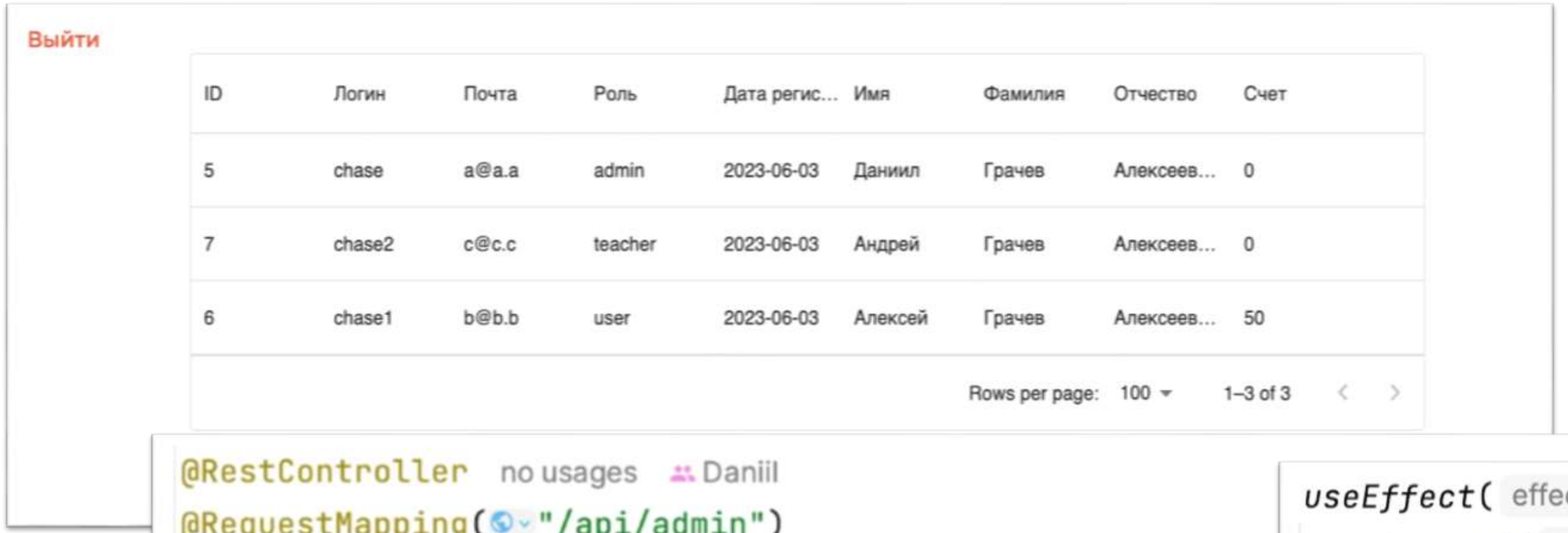
Страница тестов для преподавателя





ДОБАВИТЬ ВОПРОС

Соединение с помощью API




```java
@RestController  no usages  Daniil
@RequestMapping("/api/admin")
@RequiredArgsConstructor
public class AdminController {

    private final AdminServices adminServices;  2 usages

    @GetMapping("getallusers")  no usages  Daniil
    public List<Account> getAllUsers() { return adminServices.getAllUsers(); }

    @PostMapping("setuserrole")  no usages  Daniil
    public void setUserRoleByLogin(@RequestBody SetRoleModel model) { adminServices
}
```

```javascript
useEffect( effect: () => {
  axios.get( url: '/api/admin/getallusers')
    .then(({ data }) => {
      setUsers(data)
    })
}, deps: [])
```

Метод обработки теста

```
public int getTestGrade(HashMap<String, List<String>> test, String login) {  1 usage  Daniil
    var testId = test.get("testId").get(0);

    var DBTest = testRepository.findById(Long.valueOf(testId)).get();
    var student = studentRepository.findFirstByAccount_Login(login);

    var correctTest = new HashMap<String, List<String>>();

    int correctQuestionCount = 0;

    for (var question : DBTest.questions) {
        correctTest.put(question.id.toString(), new ArrayList<>());

        for (var answer : question.answers) {
            if (answer.isCorrect) {
                correctTest.get(question.id.toString()).add(answer.id.toString());
            } else {
                correctTest.get(question.id.toString()).add("");
            }
        }
        if (correctTest.get(question.id.toString()).equals(test.get(question.id.toString()))) {
            correctQuestionCount++;
        }
    }

    student.account.score += Math.round((float) correctQuestionCount / correctTest.size() * 100);
    DBTest.passedStudents.add(student);

    studentRepository.save(student);
    testRepository.save(DBTest);

    return student.account.score;

}
```

Для проверки правильности ответов студента из базы данных берется этот тест и преобразуется в словарь, после чего сравниваются правильные ответы с ответами от пользователя и в случае совпадения увеличивается счетчик правильных ответов у пользователя. После, из этого числа высчитывается балл по стобалльной системе, который прибавляется к очкам пользователя и позже возвращается клиенту. Также в этом методе происходит добавление студента в список сдавших тест, чтобы он не мог пройти тест дважды

Компонент «Course»

```tsx
const Test = () => {  3 usages  Daniil *
  const { testid } = useParams()
  const { data: test } = useTestById(testid as string)

  const navigate = useNavigate()

  function onsubmit (e: any) {  1 usage  Daniil *
    e.preventDefault()
    const result = { testId: [testid], ...serializer(e.target) }
    axios.post( url: '/api/courses/test', result).then(r => r.data)
    navigate( to: '/')
  }

  return (
    <div className={style.wrapper}>
      <h1 className={style.title__title}>{test?.theme}</h1>
      <form onSubmit={onsubmit}>
        {test?.questions.map( callbackfn: (item : QuestionTypes ) =>
          <fieldset key={`question${item.id}${item.question}`}>
            <legend>{item.question}</legend>
            {item.answers.map((answer : AnswerTypes ) =>
              <label key={answer.id}>
                {answer.text}
                <input type="checkbox" name={`${item.id}`} id={`question${item.id}_${answer.id}`} value={answer.id}/>
              </label>
            )}
          </fieldset>
        )}
        <input type="submit" value="Сдать"/>
      </form>
    </div>
  )
}
```

```java
@Data  22 usages  Daniil
@Entity
@Table(name = "course")
@AllArgsConstructor
@NoArgsConstructor
public class Course {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    public Long id;
    public String title;  1 usage
    public String teacher;  1 usage
    @OneToMany(mappedBy = "course")  8 usages
    public List<Test> tests;

    public Course(String title, String teacher, List<Test> tests) {  2 usages  Daniil
        this.title = title;
        this.teacher = teacher;
        this.tests = tests;
    }
}
```

Переключение вкладок профиля

```
{isShowByLogin &&
  <>
    <div className={style.pages}>
      <input type="radio" name="pages" id={style.desktop} className={style.pages__radio}
             defaultChecked={true}/>
      <label htmlFor={style.desktop} className={style.pages__label} id={style.desktop_label}>
        Рабочий стол
      </label>

      {UserStore.role !== '' &&
        <><input type="radio" name="pages" id={style.settings}
                 className={style.pages__radio}/>
          <label htmlFor={style.settings} className={style.pages__label}
                 id={style.settings_label}>Настройки</label>
        </>}
    </div>
    <button className={style.exitButton} onClick={
      () => {
        UserStore.clear()
        navigate( to: '/')
      }
    }>Выйти
    </button>
```

```
213     .leftBlock:has(.pages>input#desktop:checked) + .rightBlock > #desktopTab,
214     .leftBlock:has(.pages>input#course:checked) + .rightBlock > #courseTab,
215     .leftBlock:has(.pages>input#settings:checked) + .rightBlock > #settingsTab {
216       display: block;
217     }
```

```
89          <div className={style.rightBlock}>
90 >          <div id={clsx(style.desktopTab, !isShowByLogin ? style.tabActive : '')}...>
120          {isShowByLogin && <>
121 >          <div id={style.courseTab}...>
132 >          <div id={style.settingsTab}...>
244          </>}
245        </div>
246      </div>
```

Нагрузочное тестирование

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received K...	Sent KB/sec
login	18243	668	649	983	1102	1383	88	2059	0.00%	61.0/sec	27.11	16.71
getAccount...	18175	458	452	751	853	1102	1	1879	0.00%	60.9/sec	41.00	10.47
getTestInfo	18158	469	463	737	873	1126	1	1956	0.00%	60.9/sec	59.36	10.88
createAppeal	18124	447	436	749	882	1153	1	1936	0.00%	60.9/sec	31.96	18.79
TOTAL	72700	510	498	836	960	1231	1	2059	0.00%	243.1/sec	159.01	56.72

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received K...	Sent KB/sec
login	18308	1433	1389	2292	2778	3754	149	6975	0.00%	61.0/sec	27.11	16.67
getAccount...	18217	1290	1242	2134	2668	3602	1	7735	0.00%	61.2/sec	41.21	10.53
getTestInfo	18132	1319	1284	2118	2534	3459	50	6618	0.00%	61.5/sec	59.96	11.00
createAppeal	18019	1278	1251	2079	2505	3412	3	6302	0.00%	62.0/sec	32.55	19.12
TOTAL	72676	1330	1286	2160	2630	3574	1	7735	0.00%	242.2/sec	158.39	56.44

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received K...	Sent KB/sec
login	18397	3291	3084	4922	5737	7458	112	10366	0.00%	60.9/sec	27.06	16.58
getAccount...	18185	3114	2915	4684	5454	7066	0	9220	0.00%	60.4/sec	40.65	10.38
getTestInfo	18000	2785	2842	4024	4621	6120	1	10696	0.00%	59.9/sec	58.35	10.70
createAppeal	17774	2760	2787	3920	4450	5708	1	9492	0.00%	60.0/sec	31.51	18.51
TOTAL	72356	2990	2903	4392	5146	6769	0	10696	0.00%	239.5/sec	156.57	55.73

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received K...	Sent KB/sec
login	17849	13416	14289	21750	23063	24950	4	28859	0.01%	57.7/sec	25.64	15.49
getAccount...	17013	13305	14406	21472	22712	24410	155	28597	0.00%	55.1/sec	37.08	9.47
getTestInfo	16114	3545	1647	10138	14739	20085	145	25442	0.00%	52.6/sec	51.26	9.40
createAppeal	15976	3698	2221	9186	13287	18963	151	24895	0.00%	54.7/sec	28.73	16.87
TOTAL	66952	8693	6432	19789	21744	24032	4	28859	0.00%	216.3/sec	140.49	50.17

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была спроектирована и разработана программная система «Университет» для автоматизации взаимодействия университета с пользователями. Было разработано клиент-серверное приложение, в котором было налажено сообщение с помощью HTTP-запросов. Для отрисовки пользовательского интерфейса использовался React, для обработки запросов к API использовался ASP.NET Core 6, для взаимодействия с базой данных Entity Framework 6, в качестве СУБД использовался MySQL.