



**МИНОБРНАУКИ РОССИИ**

**федеральное государственное автономное образовательное учреждение  
высшего образования**

**«Московский государственный технологический университет «СТАНКИН»  
(ФГАОУ ВО «МГТУ «СТАНКИН»)**

---

**Институт  
информационных  
технологий**

**Кафедра информационных  
технологий и  
вычислительных систем**

**Романов Илья Олегович**

Выпускная квалификационная работа  
по направлению подготовки  
09.04.04 «Программная инженерия»

профиль «Технологии разработки интеллектуальных систем и программных комплексов»

**ИССЛЕДОВАНИЕ И РАЗРАБОТКА СИСТЕМЫ ОЦЕНКИ  
КАЧЕСТВА ПРОВЕДЁННЫХ ЗАНЯТИЙ**

Регистрационный номер № \_\_\_\_\_

Зав. кафедрой ИТиВС \_\_\_\_\_

к.т.н., доц.

Новоселова О.В.

Научный руководитель \_\_\_\_\_

ст. п.

Бердюгин А.В.

Консультант \_\_\_\_\_

к.э.н.

Карплюк Ю.А.

Обучающийся \_\_\_\_\_

Романов И.О.

**Москва 2025 г**



**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение  
высшего образования**

**«Московский государственный технологический университет «СТАНКИН»  
(ФГБОУ ВО «МГТУ «СТАНКИН»)**

---

**Институт  
информационных  
технологий**

**Кафедра информационных  
технологий и  
вычислительных систем**

**«УТВЕРЖДАЮ»**  
Заведующий кафедрой ИТиВС  
к.т.н., доц. Новоселова О.В.

«\_\_\_» \_\_\_\_\_ 2024 г.

### **Задание**

на выполнение выпускной квалификационной работы  
по направлению подготовки  
09.04.04 «Программная инженерия»  
профиль «Технологии разработки интеллектуальных систем и программных комплексов»

Студент группы ИДМ-23-08  
Романов И.О.

Научный руководитель  
ст. п.  
Бердюгин А.В.

**Тема: «Исследование и разработка системы оценки качества проведённых  
занятий»**

Тема утверждена приказом от «\_\_\_» \_\_\_\_\_ 202\_ г. № \_\_\_\_\_.  
Срок сдачи ВКР на кафедру «\_\_\_» \_\_\_\_\_ 202\_ г.

## **1. Описание задания на выполнение ВКР**

**1.1. Тип ВКР** – исследовательская работа.

**1.2. Цель исследования** – повышение эффективности оценивания занятия в университете за счёт разработки автоматизированной системы, обеспечивающей сбор статистических данных о качестве проводимых занятий.

**1.3. Объект исследования** – управление образовательными и научно-методическими процессами

**1.4. Предмет исследования** – информационно-программные средства для поддержки образовательных процессов.

**1.5. Методы исследования** – системный анализ, методы обработки данных, практическое моделирование, разработка автоматизированных систем.

**1.6. Задачи исследования:**

1.6.1. Исследовать системы оценки качества проведённых занятий.

1.6.2. Разработать модель целевой системы оценки качества проведённых занятий.

1.6.3. Выбрать и обосновать средства реализации.

1.6.4. Разработать систему оценки качества проведённых занятий с предоставлением целевых метрик.

## **2. Требования к выполнению ВКР**

### **2.1. Соблюдение требований законодательной базы и стандартов**

2.1.1. Образовательная программа высшего образования в магистратуре ФГБОУ ВО «МГТУ «СТАНКИН» по направлению подготовки 09.04.04 «Программная инженерия» для направленности (профиля) подготовки «Технологии разработки интеллектуальных систем и программных комплексов» (утв. 13.04.2022).

2.1.2. Внутренний нормативный документ. П 01-04/264/2017. Положение о государственной итоговой аттестации по образовательным программам высшего образования – программам бакалавриата, программам специалитета и программам магистратуры: [утверждено приказом ректора от 31.08.2017 г. №431/1, одобрено решением ученого совета Университета от 31.08.2017 г., протокол № 10/17] – ФГБОУ ВО «МГТУ «СТАНКИН».

2.1.3. Внутренний нормативный документ. П 01-04/438/2021. Положение о выпускной квалификационной работе обучающихся по образовательным программам высшего образования – программам бакалавриата, программам специалитета, программам магистратуры: [утверждено приказом ректора от 30.03.2021

г. №177/1, одобрено решением ученого совета Университета от 25.12.2020 г., протокол № 11/20] – ФГБОУ ВО «МГТУ «СТАНКИН».

## **2.2. Дополнительные требования**

2.2.1. Оформление раздела «Список литературы» по национальному стандарту ГОСТ Р 7.0.100-2018 “Библиографическая запись. Библиографическое описание. Общие требования и правила составления”.

2.2.2. Результаты исследования должны быть опубликованы в виде научных статей и тезисов докладов (не менее 4), а также доложены на научно-технических конференциях и семинарах.

## **2.3. Срок сдачи оформленной квалификационной работы на кафедру – вторая декада мая 2025 г.**

Исполнитель

Романов И. О.

Научный руководитель,  
ст. п. каф. ИТиВС

Бердюгин А.В.

## **АННОТАЦИЯ**

Выпускной квалификационной работы

Студента группы ИДМ-23-08 ФГАОУ ВО «МГТУ «СТАНКИН»

**Романова Ильи Олеговича**

по направлению подготовки

**09.04.04 «Программная инженерия»**

на тему

### **«Исследование и разработка системы оценки качества проведённых занятий»**

Актуальность работы. В современной российской системе образования возрастает потребность в объективной и комплексной оценке качества учебных занятий. Традиционные методы, такие как наблюдение и анкетирование, дополняются цифровыми инструментами, которые позволяют автоматизировать сбор и анализ данных.

Однако, готовые цифровые инструменты, существующие на рынке, которые позволяют автоматизировать сбор и проводить анализ собранных данных, не могут в полной мере удовлетворить потребность агрегации данных по различным административным единицам учреждений высшего образования.

В этой связи, для улучшения качества предоставления услуг образования, анализа собранных данных и аудита, планируется разработать программное решение, которое обеспечит единую точку сбора метрик, их анализа и инструментов агрегации по целевым признакам и административным единицам.

Целью данной выпускной квалификационной работы является улучшение качества занятий учреждений высшего образования, путем сбора, агрегации и анализа метрик оценки качества проведённых занятий.

Для достижения поставленной цели в работе были поставлены и решены следующие задачи:

1. Провести обзор существующих систем сбора оценок качества.
2. Обосновать требования к разрабатываемой системе для оценки качества проведённых занятий в учреждениях высшего образования
3. Провести проектирование моделей системы, включающих в себя сценарии использования и схемы данных.
4. Разработать макетный вариант системы оценки качества проведённых занятий.

Научная новизна состоит в создании единой точки сбора оценок качества проведённых занятий и исследовании инструментов агрегации полученных метрик.

Практическая ценность работы состоит в повышении качества проведённых занятий, которое является следствием регулярного анализа оценок качества.

Структура работы. ВКР состоит из Введения, пяти глав, Заключения, Списка литературы (12 наименований) и Приложения (стр.). Выпускная квалификационная работа содержит 74 страниц текста, рисунков – 32, таблиц – 6, использованных источников – 12.

Апробация работы. Полученные научные и практические результаты докладывались автором на студенческих научно-практических конференциях «Автоматизация и информационные технологии»: АИТ-2025 (1 тур) и Международной научно-практической конференции «Развитие науки и практики в глобально меняющемся мире в условиях рисков».

Выводы по работе:

1. В ходе работы было проведено исследование существующих решений сбора и анализа метрик оценки качества.
2. Были спроектированы модели типовых сценариев оценки качества проведённых занятий в учреждениях высшего образования.
3. На основе сценариев была разработана функциональная модель и схема данных для базы данных.

4. Разработан макетный вариант системы оценки качества проведённых занятий.

Список работ, опубликованных по теме ВКР:

1. Романов И.О. Современные средства оценки качества проведённых занятий// Материалы Международной студенческой научно-практической конференции «Автоматизация и информационные технологии» (АИТ-2025). – М.: МГТУ «Станкин», 2025;
2. Романов И.О. Современные средства оценки качества проведённых занятий// Материалы Международной научно-практической конференции «Развитие науки и практики в глобально меняющемся мире в условиях риска» (МКРНП-2025).

## ОГЛАВЛЕНИЕ

<b>ВВЕДЕНИЕ.....</b>	<b>10</b>
<b>ГЛАВА 1. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ .....</b>	<b>11</b>
<b>1.1. ПОНЯТИЙНЫЙ АППАРАТ.....</b>	<b>11</b>
1.1.1. Качество проведённых занятий .....	11
1.1.2. Проведённое занятие .....	11
1.1.3. Целевые метрики .....	11
1.1.4. Полное наименование системы и её условное обозначение .....	12
1.1.5. Назначение и цели создания системы .....	12
1.1.6. Цели создания системы .....	12
<b>1.2. СУЩЕСТВУЮЩИЕ РЕШЕНИЯ.....</b>	<b>12</b>
1.2.1. WebAsk.io .....	13
1.2.2. Электронная образовательная среда (СЭО ЗКЛ Русский Moodle).....	14
1.2.3. Stepik.....	16
<b>1.3. ФОРМУЛИРОВАНИЕ ТРЕБОВАНИЙ К БУДУЩЕЙ СИСТЕМЕ .....</b>	<b>17</b>
<b>1.4. ВЫВОДЫ ПО ГЛАВЕ 1 .....</b>	<b>18</b>
<b>ГЛАВА 2. РАЗРАБОТКА МОДЕЛИ СИСТЕМЫ ОЦЕНКИ КАЧЕСТВА ПРОВЕДЁННЫХ ЗАНЯТИЙ .....</b>	<b>19</b>
<b>2.1. СХЕМА ДАННЫХ.....</b>	<b>20</b>
2.1.1. Процесс «Расписание».....	21
2.1.2. Процесс «Аутентификация и авторизация» .....	24
2.1.3. Процесс «Тестирование» .....	26
2.1.4. Процесс «Оценка качества проведённого занятия» .....	27
<b>2.2. ДАННЫЕ СТУДЕНТА. МОДУЛЬНЫЙ ЖУРНАЛ .....</b>	<b>28</b>
<b>2.3. ВЫВОДЫ ПО ГЛАВЕ 2 .....</b>	<b>29</b>
<b>ГЛАВА 3. ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ СРЕДСТВ РЕАЛИЗАЦИИ .....</b>	<b>30</b>
<b>3.1. ОПИСАНИЕ ТРЕБОВАНИЙ К СРЕДСТВАМ РЕАЛИЗАЦИИ РАЗРАБАТЫВАЕМОЙ СИСТЕМЫ .....</b>	<b>30</b>
3.1.1. Пользовательский интерфейс .....	30
3.1.2. Хранение данных .....	31
3.1.3. Агрегация данных.....	32
3.1.4. Выбор инструмента агрегации данных .....	33
3.1.5. Сравнение Elasticsearch и PostgreSQL .....	33
<b>3.2. ТРЕБОВАНИЕ К ИНСТРУМЕНТУ РЕАЛИЗАЦИИ СЕРВЕРНОГО ПРИЛОЖЕНИЯ .....</b>	<b>35</b>
3.2.1. Выбор языка программирования .....	35
3.2.2. Сравнение Java и Kotlin .....	35
<b>3.3. ВЫВОДЫ ПО ГЛАВЕ 3 .....</b>	<b>37</b>
<b>ГЛАВА 4. РЕАЛИЗАЦИЯ .....</b>	<b>38</b>
<b>4.1. РЕАЛИЗАЦИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА .....</b>	<b>39</b>
4.1.1. Расписание занятий.....	40
4.1.2. Тестирование и опросы .....	42
4.1.3. Модульный журнал. OAUTH 2.0 провайдер.....	47
<b>4.2. РЕАЛИЗАЦИЯ СЕРВЕРНОЙ ЧАСТИ.....</b>	<b>50</b>
4.2.1. Авторизация и аутентификация .....	51
4.2.2. Сетевое взаимодействие серверного приложения с пользовательским интерфейсом .....	53
4.2.3. Сетевое взаимодействие серверного приложения с СУБД PostgreSQL .....	55



4.2.4. Преобразования файлов расписания в целевой формат хранения.....	60
4.2.5. Сбор метрик тестирования студентов и оценивания занятий.....	63
4.2.6. Результат агрегации и генерации метрик оценки качества проведённых занятий .....	64
<b>4.3. ВЫВОДЫ ПО ГЛАВЕ 4 .....</b>	<b>65</b>
<b><i>ГЛАВА 5. ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ СИСТЕМЫ ОЦЕНКИ КАЧЕСТВА ПРОВЕДЁННЫХ ЗАНЯТИЙ «СОКПЗ» .....</i></b>	<b><i>66</i></b>
5.1. ОЦЕНКА ЗАТРАТ ПРОЕКТА.....	66
5.2. ОБОСНОВАНИЕ ЭКОНОМИЧЕСКОЙ ВЫГОДЫ ПРОЕКТА .....	69
5.3. ВЫВОДЫ ПО ГЛАВЕ 5 .....	71
<b>ЗАКЛЮЧЕНИЕ .....</b>	<b>71</b>
<b>СПИСОК ЛИТЕРАТУРЫ .....</b>	<b>73</b>

## **ВВЕДЕНИЕ**

Темой выпускной квалификационной работы является «Исследование и разработка системы оценки качества проведённых занятий». Данная тема подразумевает под собой проектирование и реализацию программного продукта, полностью удовлетворяющего требованиям конечного потребителя, а именно – учреждение высшего образования. Цель данного программного продукта – автоматизация процесса оценки качества проведённых занятий с предоставлением целевых метрик.

Первоначально следует исследовать рынок программного обеспечения на наличие уже готовых программных решений, с помощью которых может быть удовлетворена потребность учреждений высшего образования в программном продукте.

## **ГЛАВА 1. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

### **1.1. ПОНЯТИЙНЫЙ АППАРАТ**

Перед началом разработки требований к программному продукту, необходимо определить понятийный аппарат в выбранной предметной области.

#### **1.1.1. Качество проведённых занятий**

Под качеством проведённых занятий, в данной предметной области, подразумевается объективная оценка, основанная на численных методах. В данном случае оценка может быть двунаправленная – как роль «студент» может оценить качество проведённого занятия, так и роль «преподаватель» может выставить свою оценку для конкретной группы или потока.

#### **1.1.2. Проведённое занятие**

Под проведённым занятием, в данной предметной области, подразумевается событие, обозначенное в учебном плане, связанное с Обществом, в котором участвуют роли «Студент» и «Преподаватель».

#### **1.1.3. Целевые метрики**

Под целевыми метриками, в данной предметной области, подразумевается набор численных агрегатов или словарно-семантических диаграмм, собранных в ходе выполнения целевых бизнес-процессов, наглядно иллюстрирующих объективную, с точки зрения статистики, но субъективную с точки зрения методологии, оценку качества проведённых занятий.

#### **1.1.4. Полное наименование системы и её условное обозначение**

Полное наименование – «Система оценки качества проведённых занятий». Шифр – «СОКПЗ»

#### **1.1.5. Назначение и цели создания системы**

Разрабатываемая система предназначена для оценки качества проведённых занятий. СОКПЗ должна создаваться в качестве единой точки сбора, агрегации и предоставления агрегированных данных в виде аналитических отчётов оценок качества проведённых занятий.

СОКПЗ служит для обеспечения:

1. Предоставления данных о качестве проведенных занятий Общества.
2. Хранение и агрегация данных в рамках системы.
3. Единую точку входа для как обучающихся общества, имеющих роль «Студент», так и для роли «Преподаватель», для возможности выставления оценки качества проведения занятий.

#### **1.1.6. Цели создания системы**

В качестве основных целей создания системы можно выделить:

1. Формирование статистических данных о качестве проведения занятий.
2. Агрегация и расчёт статистических данных по институтам, кафедрам и иным административным единицам общества, для дальнейшего прикладного применения полученных сведений.

### **1.2. СУЩЕСТВУЮЩИЕ РЕШЕНИЯ**

Чтобы сформулировать требования для актуальной системы оценки качества проведённых занятий со сбором, агрегацией и анализом целевых метрик, необходимо провести анализ существующих решений для того, чтобы

при разработке функциональных и нефункциональных требований к системе учитывать все актуальные тенденции на рынке.

Далее будут рассмотрены самые распространённые, по общей функциональности, системы оценок.

### **1.2.1. WebAsk.io**

WebAsk.io – это российский сервис проведения онлайн-тестирований, анкетирования, маркетинговых и продуктовых исследований [13].

WebAsk.io решает следующие задачи:

1. Изучение клиентского опыта – анализ конкурентов, изучение потребностей покупателей, сбор обратной связи, оценка уровня лояльности.
2. Анализ продукта – оценка логотипа, макетов, дизайна упаковок, UX-тестирование, сравнение с конкурентами.
3. Проверка гипотез – тестирование концептов, изучение спроса.
4. Маркетинговые исследования - исследование рынка, проверка креативов, оценка эффективности рекламных кампаний, изучение ЦА, измерение уровня знания бренда.
5. Работа с персоналом - оценка удовлетворенности сотрудников, аттестации персонала, сбор обратной связи, тестирование персонала, опросы сотрудников.
6. Образовательные учреждения - психологические тестирования, исследование целей и ценностей обучающихся, тестирование учащихся, оценка качества занятий.

WebAsk.io предоставляет инструментарий для создания открытых и закрытых опросов. По проведению опроса, реализован функционал просмотра итоговой статистики. Данное программное обеспечение является платным, и предоставляется по подписке. В бесплатном тарифе есть ряд существенных ограничений – доступно только 3 опроса на аккаунт, 100 ответов в месяц и 10 элементов на опрос.

### **1.2.2. Электронная образовательная среда (СЭО 3KL Русский Moodle)**

В качестве примера электронной образовательной среды автор предлагает рассмотреть электронную образовательную среду ФГБОУ ВО МГТУ «СТАНКИН». Данная электронная образовательная среда реализована на базе СЭО (среда электронного обучения) 3KL (Русский Moodle). 3KL (Русский Moodle) – среда электронного образования, которая предоставляет инструменты для создания учебных курсов и организации учебного процесса.

Данная среда адаптирована под нужды российских заказчиков и применяется в корпоративном секторе, образовании, сфере дополнительного образования.

СДО 3KL Русский Moodle является профессиональной версией базовой системы Moodle и имеет с ней полную совместимость, распространяется компанией «Открытые Технологии» [7].

В отличие от базовой версии, СЭО 3KL Русский Moodle поставляется с техподдержкой и содержит более 50-ти дополнительных модулей, разработанных компанией «Открытые Технологии»

Профессиональная версия СЭО 3KL® Русский Moodle включает в себя широкий спектр функционала.

Панель управления СЭО 3KL - инструмент администрирования, предназначенный для работы с основными объектами системы: курсами, пользователями, глобальными группами. Позволяет выполнять массовые действия с объектами системы, например, записывать/отчислять глобальные группы пользователей на выбранные курсы. Предоставляет широкие возможности для анализа (пользовательской активности, использования и востребованности курсов и т.п.) при помощи удобного интерфейса редактируемых вкладок, содержащих настраиваемые таблицы параметров объектов СЭО 3KL. Создаваемые вкладки и их настройки индивидуальны для каждого пользователя. Предусматривает выгрузку данных из вкладок в файл

с возможностью выбора формата сохранения, для последующего формирования статистической и аналитической отчетности.

Модуль «Электронный деканат» - управляющая надстройка системы для управления учебными процессами на основе учебных планов, параллелей и академических групп. Позволяет распределить объекты учебного процесса по иерархической структуре подразделений с разграничением прав доступа. Включает необходимую электронную документацию и отчетность, автоматизацию взаимодействия участников и индивидуальные траектории, значительно экономит время регламентных процедур, сохраняет историю всех процессов и событий.

Витрина курсов - графический каталог с широчайшими возможностями настройки оформления как списка (плиток) курсов и категорий, так и описательной страницы отдельно взятой дисциплины (курса). Предназначена для привлечения слушателей и онлайн-продаж курсов. Может использоваться для организации массовых онлайн-курсов. Каталог содержит дерево категорий курсов, списки курсов, страницу публичной информации о курсе, вывод метаинформации, плитки и иконки, встроенный поиск и фильтрацию курсов по названиям, датам, ценам, преподавателям и любым настраиваемым полям. Витрина может отображаться на любых страницах портала в любом удобном пользователю регионе интерфейса СЭО ЗКЛ.

Система учета достижений и целей «Портфолио» позволяет организовать индивидуальный план развития персонала, формировать личные списки достижений пользователя, осуществлять целевое планирование в обучении и использоваться как инструмент для формирования кадрового резерва. Учет достижений включает подсистемы расчета рейтинга, модерации и фильтрации. Соответствует стандарту ФГОС 3++.

Данная платформа предлагает обширный функционал для ведения образовательной деятельности в цифровом формате.

Для эксплуатации данного программного обеспечения необходимо приобрести лицензию. В стоимость лицензии входит установка, техническая

поддержка и обновление продукта, а также доступ к базе знаний и обучающим вебинарам.

### 1.2.3. Stepik

Stepik - российская образовательная платформа и конструктор бесплатных и платных открытых онлайн-курсов и уроков [11]. Данная платформа предоставляет интерфейс, который позволяет любому зарегистрированному создавать интерактивные обучающие онлайн-курсы, используя аудио, видео, текст и разнообразные задачи с автоматической проверкой и моментальной обратной связью.

В процессе обучения пользователи могут вести обсуждения между собой и задавать вопросы преподавателю на форуме. Основные охватываемые курсами дисциплины — программирование, математика, биоинформатика и биология, экономика; основной язык курсов — русский, есть курсы на английском языке. По состоянию на 2020 год на платформе зарегистрировано 5 миллионов пользователей.

На основе сведений о существующих решениях, приведена таблица 1.1 представляющая из себя сравнительную характеристику функциональных свойств этих решений.

Таблица 1.1

Сравнительная характеристика существующих решений

Название	Анализ вовлеченности студентов	Оценка проведённого занятия ролью «Преподаватель»	Форма для создания тестовых заданий	Агрегация метрик по различным административным единицам
Google forms	-	-	+	-
СЭО ЗКЛ Русский Moodle	-	-	+	-
Stepik	-	-	+	-



Оценка проведённого занятия - прошедшие занятие участники с ролью «студент» могут оставить свою оценку/отзыв о качестве проведённого занятия.

Оценка группы - участник процесса с ролью «Преподаватель» может оценить конкретное множество участников с ролью «студент» под названием «группа» по набору конкретных характеристик, например: количество, вовлеченность и результат «завершающего» краткого тестирования.

Форма для создания тестовых заданий - характеристика, описывающая наличие удобной пользовательской формы для создания «завершающего» тестового задания, с целью прохождения этого задания студентами.

Агрегация метрик по различным административным единицам - характеристика, определяющая возможность агрегации метрик по административным единицам. В предметной области «Образование» такими единицами могут стать: предмет, факультет, кафедра, институт, поток, группа, подгруппа и т.д..

### **1.3. ФОРМУЛИРОВАНИЕ ТРЕБОВАНИЙ К БУДУЩЕЙ СИСТЕМЕ**

На основе сравнительной характеристики существующих решений, сформулированные следующие функциональные требования к системе:

1. Система должна предоставлять единую точку входа для пользователей с различными ролями.
2. Система должна обеспечить способ получения, хранения, агрегации, отображения, и формирования отчётов по итогам оценивания качества проведённых занятий по различным административным единицам.
3. Процессы авторизации и аутентификации пользователей системы должны быть реализованы средствами Системы.
4. Технологические операции, выполняемые на технических средствах Системы должны быть полностью автоматизированы.

5. Результатом формирования отчётов системы должны быть заранее согласованного с заказчиком формата.
6. В системы должны быть включены роли студента, преподавателя и аудитора.
7. Роль «Аудитор» включает в себя роли «Студент» и «Преподаватель».

#### **1.4. ВЫВОДЫ ПО ГЛАВЕ 1**

В настоящей главе был рассмотрен понятийный аппарат предметной области оценки качества проведённых, рассмотрены существующие решение на рынке, предлагающие похожий функционал для решения поставленной задачи, приведена сравнительная характеристика таких решений, а также исходя из приведённой характеристики были сформированы нефункциональные требования к системе.

## ГЛАВА 2. РАЗРАБОТКА МОДЕЛИ СИСТЕМЫ ОЦЕНКИ КАЧЕСТВА ПРОВЕДЁННЫХ ЗАНЯТИЙ

В данной главе описаны независимые от средств реализации аспекты проектирование программной системы - диаграмма прецедентов, схемы данных, внешние интеграции и т. п. На рисунке 2.1 приведена диаграмма прецедентов информационной системы для оценки качества проведённых занятий. Данная диаграмма отражает взаимодействие между участниками процесса и функциональными возможностями системы.

Основными акторами системы являются преподаватель, студент и внешний аудит. Преподаватель выполняет такие действия, как составление критериев оценки занятий и тестов, анализ статистики данных, просмотр обратной связи от обучающихся, а также генерацию отчётов. Обучающийся взаимодействует с системой путём выполнения тестов, предоставления письменной обратной связи и выставления оценок за занятия. Аудитор выступает в роли внешнего участника, взаимодействующего с системой, например, для получения отчётов или анализа данных.

Ключевые прецеденты включают авторизацию, которая является отправной точкой для работы с системой. Преподаватель может составлять критерии оценки занятий и тесты, что расширяет базовый процесс авторизации. Обучающиеся имеют возможность просматривать критерии, оставлять обратную связь и проходить тестирование. Система также поддерживает генерацию отчётов на основе собранных данных, анализ статистики и анонимных выборок для оценки эффективности занятий.

Связи между прецедентами обозначены с помощью зависимостей `<<include>>` и `<<extend>>`. Например, анализ эффективности занятий обязательно включает ознакомление с критериями, а составление критериев оценки расширяет процесс авторизации.

Диаграмма наглядно демонстрирует структуру взаимодействия пользователей с системой, обеспечивающей оценку качества учебных занятий, сбор обратной связи и анализ результатов.

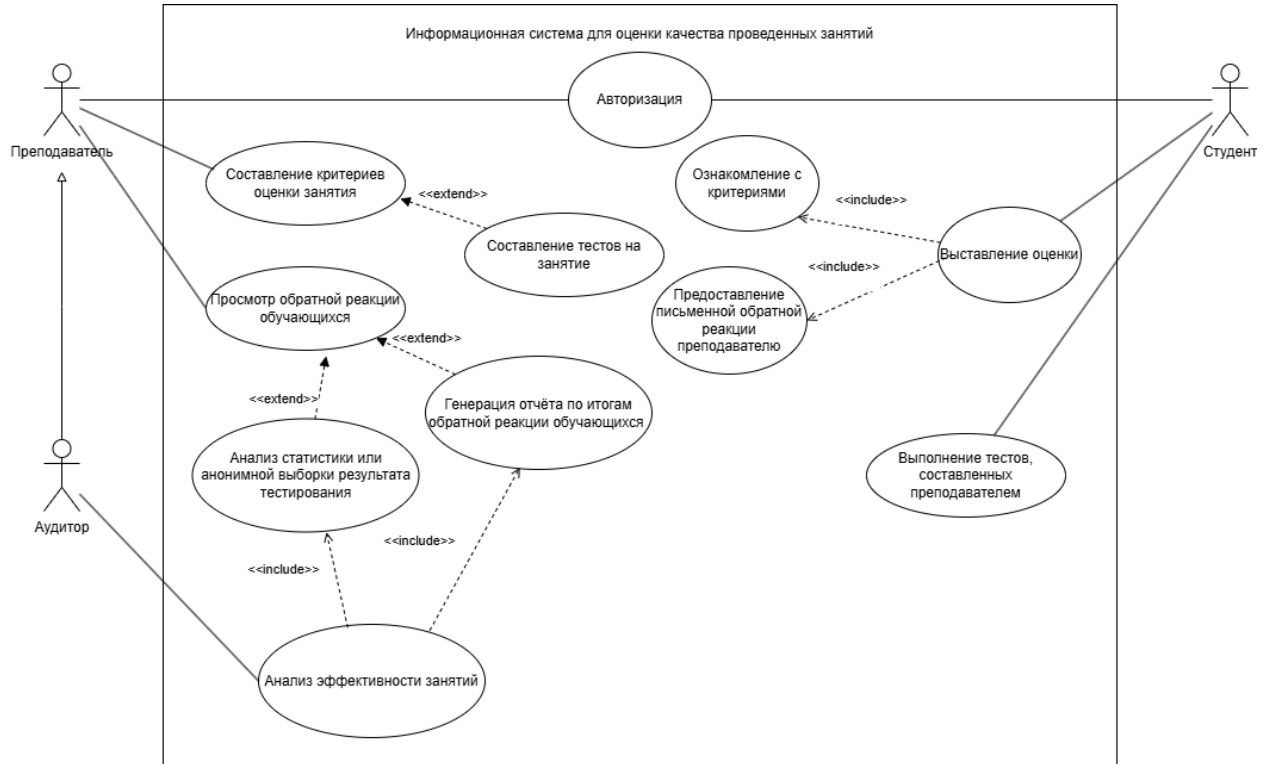


Рис. 2.1. Диаграмма прецедентов СОКПЗ

## 2.1. СХЕМА ДАННЫХ

На рисунке 2.2 представлена полная схема данных СОКПЗ.

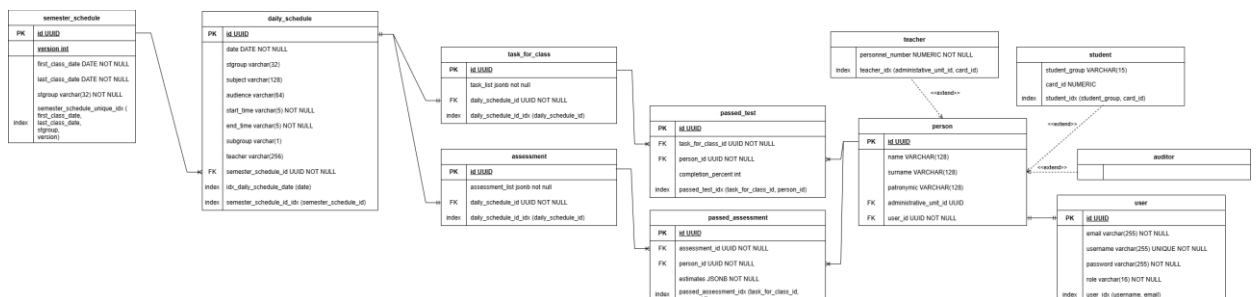


Рис. 2.2. Схема данных для процесса СОКПЗ

В следующих пунктах текущего раздела будут детально описаны процессы, представленные на схеме данных.

### 2.1.1. Процесс «Расписание»

Схема базы данных на рисунке 2.3 для процесса создания расписания включает четыре основные таблицы, связанные между собой.

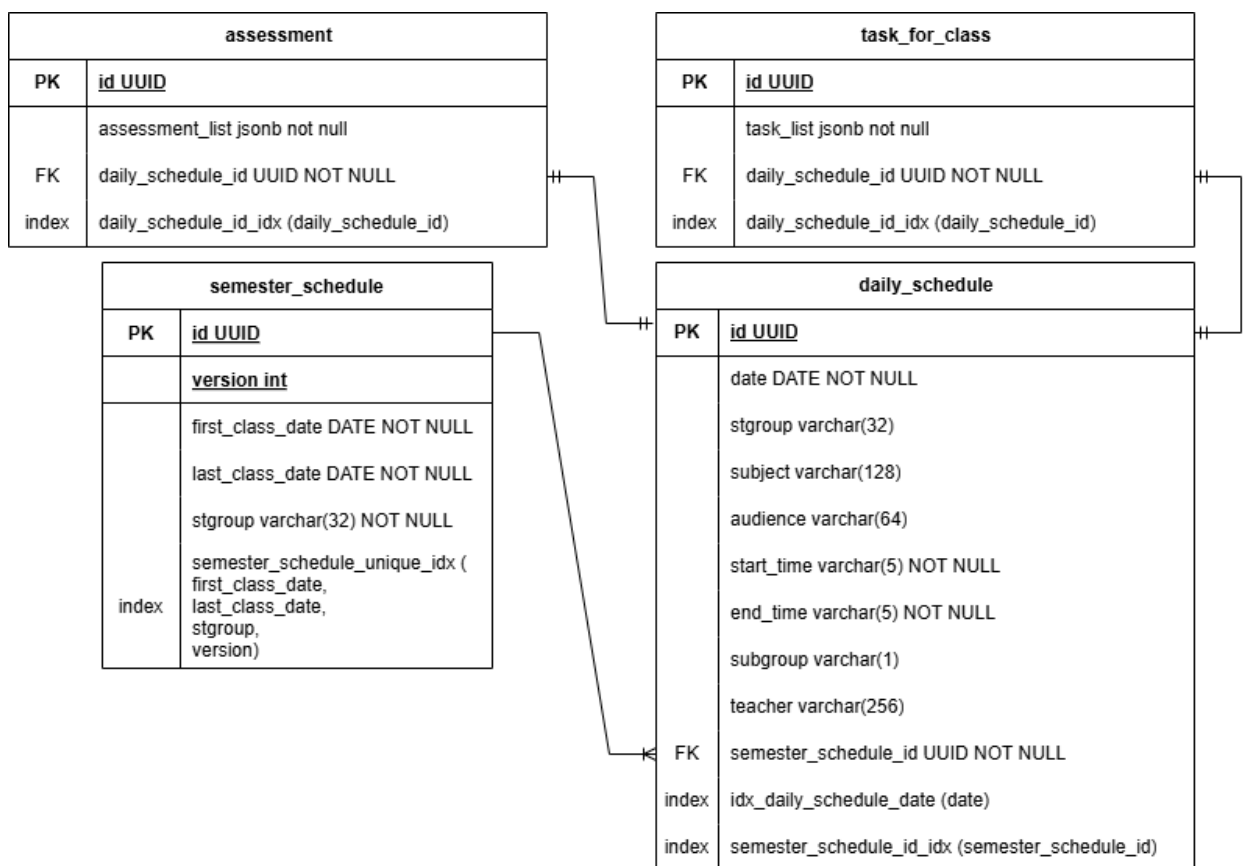


Рис. 2.3. Схема данных для процесса «Расписание»

Процесс "Расписание" в системе оценки качества учебных занятий представляет собой многоуровневую структуру, обеспечивающую планирование, проведение и анализ эффективности образовательного процесса. В его основе лежит семестровое расписание, которое определяет временные рамки учебного периода для конкретной группы студентов, включая даты начала и окончания занятий. Это общее расписание детализируется через ежедневные планы занятий, содержащие информацию о

времени проведения, аудитории, преподавателя, типа занятия и учебном предмете.

Каждое занятие может быть связано с набором учебных заданий, представленных в структурированном формате, что позволяет адаптировать требования под конкретный урок. Для оценки качества проведённых занятий используется система критериев, также хранящаяся в гибком формате, что даёт возможность настраивать параметры оценивания в зависимости от дисциплины или методики преподавания.

Сбор данных о результатах обучения осуществляется через фиксацию выполнения заданий студентами, включая процент освоения материала, а также через качественные оценки по установленным критериям. Это позволяет анализировать эффективность занятий как с точки зрения усвоения знаний, так и с позиции организации учебного процесса. Взаимосвязь между семестровым планированием, ежедневными занятиями, учебными заданиями и системой оценки создаёт целостный механизм для управления образовательным процессом и постоянного повышения его качества.

Таблица `semester_schedule` хранит информацию о семестровом расписании. Она содержит уникальный идентификатор (`id` UUID), версию расписания (`version` int), даты первого и последнего занятий (`first_class_date` и `last_class_date` типа `DATE` с ограничением `NOT NULL`), а также идентификатор группы (`stgroup` `varchar(32)` `NOT NULL`). Для обеспечения целостности данных в таблице определен уникальный индекс `semester_schedule_unique_idx`, включающий даты занятий, идентификатор группы и версию расписания.

Таблица `daily_schedule` содержит детализированную информацию о каждом занятии. Первичный ключ таблицы - `id` UUID. Для связи с семестровым расписанием используется поле `semester_schedule_id` `UUID NOT NULL`. В таблице хранятся такие данные как дата занятия (`date` `DATE NOT NULL`), идентификатор группы (`signoup` `varchar(32)`), название предмета (`subject` `varchar(128)`), аудитория (`audience` `varchar(64)`), время начала и

окончания (start\_time и end\_time varchar(5) NOT NULL), подгруппа (subgroup varchar(1)) и преподаватель (teacher varchar(256)). Для оптимизации запросов созданы индексы: idx\_daily\_schedule\_date по дате занятия и semester\_schedule\_id\_idx по идентификатору семестрового расписания

Таблица task\_for\_class предназначена для хранения тестовых заданий к занятиям. Первичный ключ - id UUID. Связь с ежедневным расписанием осуществляется через поле daily\_schedule\_id UUID NOT NULL с соответствующим индексом daily\_schedule\_id\_idx. Список тестовых заданий хранится в формате JSON в поле task\_list jsonb NOT NULL.

Таблица assessment является ключевым элементом системы мониторинга качества учебного процесса, обеспечивающим хранение и обработку критериев оценки проведенных занятий. В её структуре используется уникальный идентификатор id типа UUID, выполняющий роль первичного ключа и гарантирующий однозначную идентификацию каждой записи. Основное содержание таблицы представлено полем assessment\_list формата jsonb, которое содержит гибко настраиваемый набор оценочных критериев, включая параметры методической эффективности, уровень вовлеченности студентов и соответствие образовательным стандартам.

Связи между таблицами организованы следующим образом: таблица daily\_schedule связана с semester\_schedule через поле semester\_schedule\_id, а task\_for\_class и assessment связана с daily\_schedule через daily\_schedule\_id. Такая структура позволяет эффективно организовать данные о расписании занятий и связанных с ними тестовых заданиях, обеспечивая целостность данных и быстрый доступ к необходимой информации.

### 2.1.2. Процесс «Аутентификация и авторизация»

На рисунке 2.4 представлена схема данных процесса «Авторизация и аутентификация»

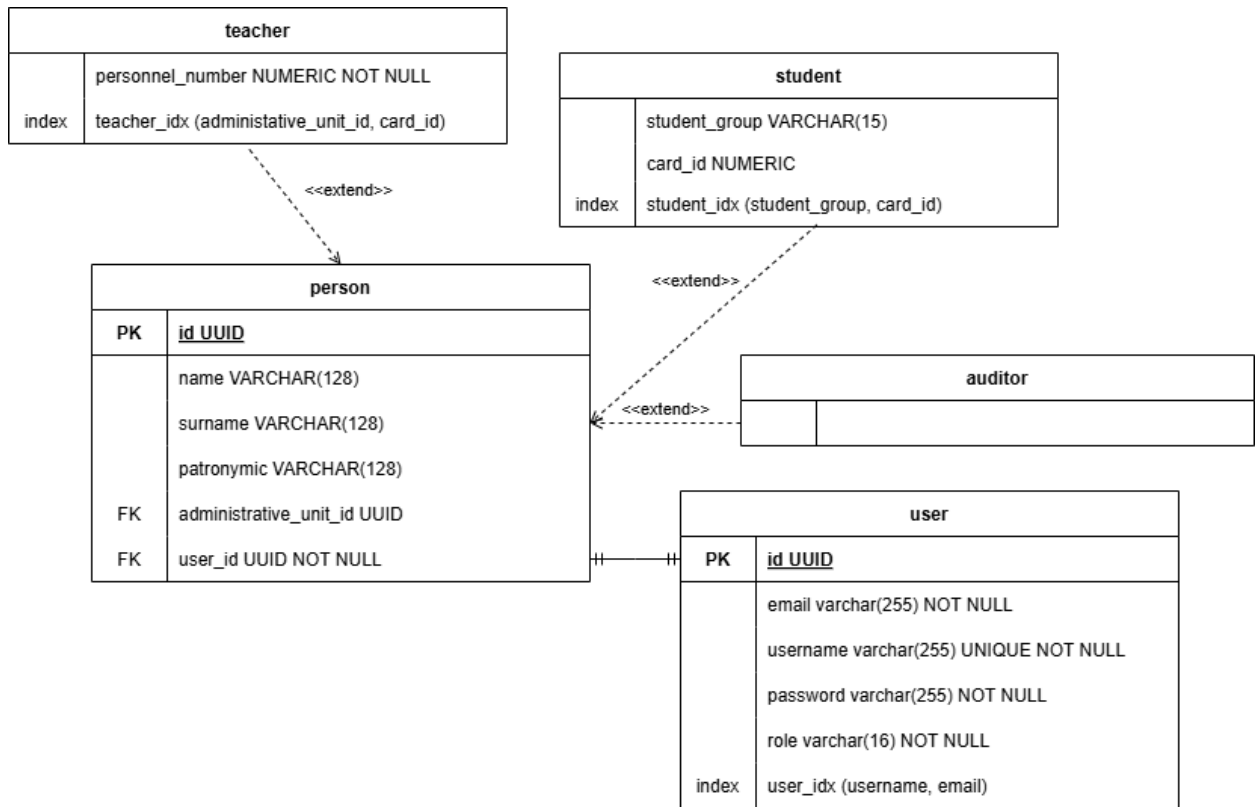


Рис. 2.4. Схема данных процесса «Авторизация и аутентификация»

Схема базы данных для платформы оценки качества учебных занятий, изображённая на рисунке, построена на основе наследования сущностей, где таблицы **auditor** (аудитор) и **student** (студент) и **teacher** (преподаватель), расширяют базовую логику таблицы **person** (базовый участник учебного процесса) используя одинаковый набор атрибутов. Центральным элементом системы является таблица **user**, содержащая общие учетные данные для всех типов пользователей.

Таблица **user** включает обязательные поля для авторизации: уникальный идентификатор (`id` UUID) как первичный ключ, электронную почту (`email` varchar(255) с ограничением NOT NULL, уникальное имя пользователя



(username varchar(255) с ограничениями UNIQUE и NOT NULL, зашифрованный пароль (password varchar(255) NOT NULL, а также роль пользователя (role varchar(16)) NOT NULL, которая определяет тип учетной записи (преподаватель, студент или аудитор). Для ускорения поиска создан составной индекс user\_idx по полям username и email.

Таблицы teacher, auditor и student имеют идентичную структуру, что реализует паттерн наследования в реляционной модели. Каждая из этих таблиц содержит: первичный ключ (id UUID), персональные данные (name, surname и patronymic типа VARCHAR (128)), служебный идентификатор (personal\_id NUMERIC), ссылку на структурное подразделение (administrative\_unit UUID NOT NULL) и обязательную связь с учетной записью через поле user\_id UUID NOT NULL. В таблице student дополнительно присутствуют поля для работы с учебными группами (student\_group VARCHAR (15)) и системами идентификации (card\_id).

Связи между таблицами организованы следующим образом: каждая из таблиц-наследников (teacher, auditor, student) связана с основной таблицей user через поле user\_id, которое является внешним ключом, ссылающимся на id в таблице user. Такая архитектура позволяет:

- обеспечить единый механизм авторизации для всех типов пользователей;
- сохранять специализированные данные для каждой роли;
- поддерживать целостность данных через внешние ключи;
- реализовать разграничение прав доступа на основе ролей.

Для оптимизации запросов в таблицах-наследниках созданы соответствующие индексы, включая student\_idx для работы со студенческими группами.

### 2.1.3. Процесс «Тестирование»

На рисунке 2.5 представлена схема данных процесса «Тестирование».

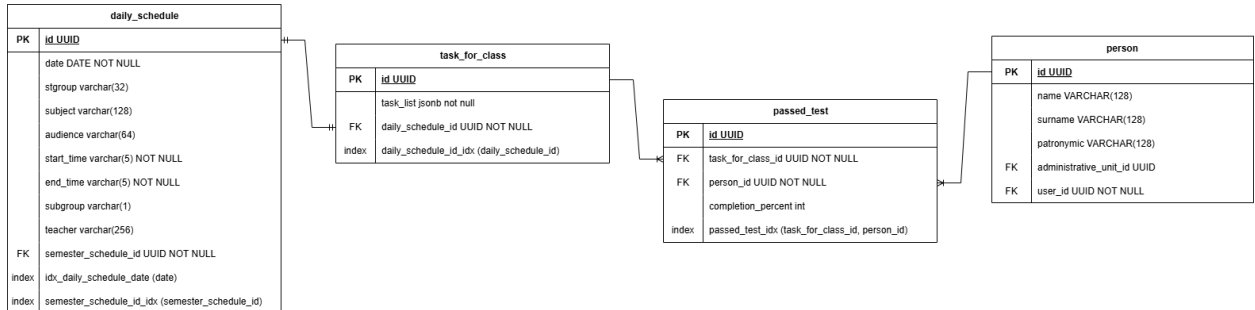


Рис. 2.5. Схема данных процесса «Тестирование»

Процесс тестирования представляет собой многоуровневую систему оценки знаний студентов, интегрированную в общую структуру учебного расписания. Основой процесса выступает таблица `daily_schedule`, содержащая информацию о запланированных учебных занятиях с указанием даты, времени, аудитории, преподавателя и других параметров проведения урока. Каждое занятие может быть связано с набором учебных заданий через таблицу `task_for_class`, где в структурированном json-формате хранится перечень задач и тестовых материалов, подготовленных для студентов.

Результаты прохождения тестов фиксируются в таблице `passed_test`, которая связывает конкретного студента из таблицы `person` с набором заданий. Поле `completion_percent` количественно отражает степень выполнения тестовых заданий, позволяя оценить уровень усвоения материала. Связи между таблицами организованы через систему внешних ключей с каскадным удалением, что обеспечивает целостность данных при изменении расписания или удалении пользователей.

Таблица `person` содержит полные анкетные данные участников учебного процесса, включая ФИО, номер группы и идентификационные данные, что позволяет однозначно соотносить результаты тестирования с конкретными студентами. Особенностью системы является возможность привязки

пользователей к учетным записям через связь с таблицей пользователей, что обеспечивает интеграцию процессов тестирования в общую систему управления образовательным процессом. Таким образом, процесс тестирования представляет собой замкнутый цикл от планирования учебных занятий через назначение заданий к фиксации и анализу результатов выполнения.

#### 2.1.4. Процесс «Оценка качества проведённого занятия»

На рисунке 2.6 представлена схема данных процесса «Оценка качества проведённого занятия»

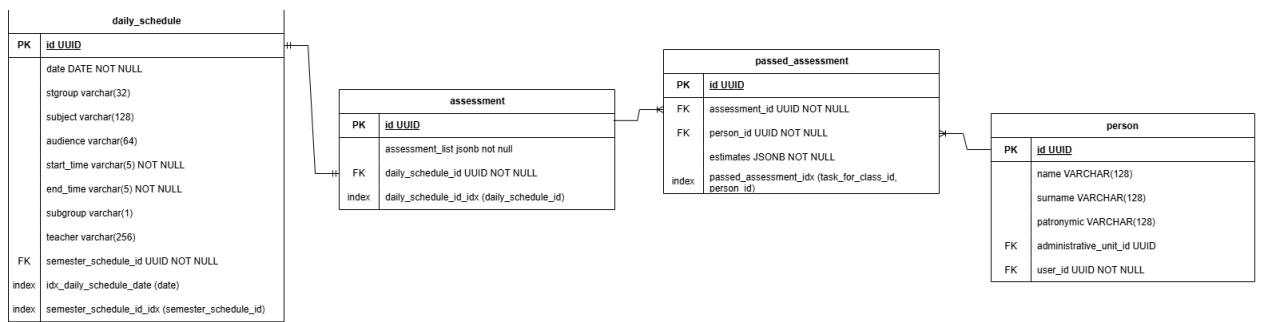


Рис. 2.6. Схема данных процесса «Оценка качества проведённого занятия»

Основой процесса служит таблица `daily_schedule`, содержащая полные данные о запланированных занятиях, включая временные параметры, состав участников и организационные детали. Ключевым элементом оценки является таблица `assessment`, хранящая в формате JSON набор критериев для предстоящей студентам оценки качества занятия.

Процедура оценки предполагает двустороннюю связь между расписанием и оценочными критериями: с одной стороны, через обязательное поле `daily_schedule_id` в таблице оценок, с другой - через опциональную ссылку `assessment_id` в самом расписании. Это позволяет как прикреплять различные оценочные методики к занятиям, так и оперативно находить все оценки для конкретного урока. Результаты оценивания фиксируются в таблице `passed_assessment`, где каждый отзыв привязывается к конкретному

студенту из таблицы person и содержит детализированные оценки в структурированном JSON-формате.

Особенностью системы является гибкость оценочного механизма: использование JSON-формата позволяет адаптировать критерии оценки под разные дисциплины и типы занятий без изменения структуры базы данных. Каскадное удаление связанных записей обеспечивает целостность данных при обновлении расписания или удалении пользователей. Собранные данные позволяют проводить многомерный анализ качества преподавания, учитывающий как объективные показатели эффективности занятий, так и субъективные оценки участников образовательного процесса.

## **2.2. ДАННЫЕ СТУДЕНТА. МОДУЛЬНЫЙ ЖУРНАЛ**

Для того, чтобы обеспечить прозрачность процесса оценки качества занятий, возможность оценивания предоставляется только авторизованным через модульный журнал пользователям. Для этого в системе предусмотрен функционал авторизации через Модульный журнал как OAuth2.0-провайдер [15].

Это означает, что система никак не хранит и не обрабатывает данные о логине и пароле студента. СОКПЗ перенаправит на целевую страницу аутентификации пользователя в модульном журнале, и после успешного прохождения процедуры вернёт токен, с помощью которого можно будет получить актуальные данные студента.

Для этого автор связался с разработчиками данного программного продукта и предоставил данные для возможности такого способа авторизации.

Детально данный процесс будет описан в главе, посвящённой средствам реализации.

### **2.3. ВЫВОДЫ ПО ГЛАВЕ 2**

В результате разработки модели системы оценки качества проведённых занятий были спроектированы схемы данных для процессов авторизации и аутентификации, тестирования, оценки качества проведённых занятий и центрального процесса расписания. Полученные схемы данных будут применены на этапе разработки прототипа программного продукта.

Также была описана диаграмма прецедентов, содержащая целевые бизнес-процессы системы оценки качества проведённых занятий.

## **ГЛАВА 3. ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ СРЕДСТВ РЕАЛИЗАЦИИ**

### **3.1. ОПИСАНИЕ ТРЕБОВАНИЙ К СРЕДСТВАМ РЕАЛИЗАЦИИ РАЗРАБАТЫВАЕМОЙ СИСТЕМЫ**

Разрабатываемая система, с точки зрения функциональности, должна удовлетворять ряду требований, чтобы выполнять поставленную задачу. Среди таких требований:

- предоставление удобного пользовательского интерфейса;
- долговременное хранение данных о пользователях, ролях, занятиях и связанных с ними событиях;
- API (Application Programming Interface) для анализа и агрегации данных и т. д.;
- система аутентификации и авторизации пользователей с ролевой моделью;
- производительная серверная часть.

Для каждой, из описанных задач далее будет приведён список средств реализаций, который удовлетворяет требованиям системы.

#### **3.1.1. Пользовательский интерфейс**

Для инструментов реализации пользовательского интерфейса, в связи с требуемой функциональностью, сформировался ряд требований, среди которых:

- поддержка маршрутизации «из коробки» - технология должна предоставлять API, для удобного создания многостраничных веб-приложений;
- поддержка асинхронного взаимодействия с сервером — для улучшения пользовательского опыта;
- низкий порог входа.

В связи с описанными требованиями, автором было принято решение, реализовывать пользовательский веб-интерфейс используя стек технологий, основанный на TypeScript и его серверном фреймворке Next.js.

TypeScript – расширение языка JavaScript, вводящее ограничения в систему типов языка JavaScript [12]. Благодаря данному расширению контроль над типами производится на этапе компиляции, а ошибки, связанные с несовместимостью операций над различными типами, можно выявить на более раннем этапе.

Next.js – это основанный на React фреймворк, предназначенный для разработки веб-приложений, обладающих функционалом, выходящим за рамки SPA (Single Page Application), т.е. так называемых одностраничных приложений [5]. Выбор пал в пользу данного инструмента, благодаря встроенной поддержке создания URL на основе исходных путей в файловой системе проекта, что позволяет гибко и просто организовывать иерархию адресов в веб-приложении.

### **3.1.2. Хранение данных**

В рамках текущей системы, автором определены три категории данных, которые необходимо хранить и обрабатывать.

- пользователь и смежные с ним данные (Роли, принадлежность к кафедре, институту, группе и т. п.). Данные определённого формата, которые в дальнейшем с наименьшей вероятностью будут меняться;
- данные для веб-интерфейса – это созданные самими пользователями структуры данных, для дальнейшего отображения в веб-интерфейсе, например: json-объект, содержащий данные для процесса «Тестирование»;
- данные о качестве проведённых занятий. Такими данными могут выступать оценки группе/преподавателю или текстовый отзыв.

Для каждой из этих категорий, подобран инструмент, удовлетворяющий требованиям.

Для хранения данных о пользователе средством реализации будет служить PostgreSQL, т. к. он удовлетворяет ACID (Атомарность, Согласованность, Изолированность, Надёжность) [16]. Это означает, что данные в рамках использования PostgreSQL будут согласованы и надёжно сохранены;

Для хранения данных для рендеринга пользовательского интерфейса будет использован так же PostgreSQL, т.к. его типы данных «json» и «jsonb» обеспечивают необходимую скорость и селективность данных;

Для того чтобы хранить, агрегировать и анализировать данные, не изобретая свой собственный инструмент, автором принято решение, для этих целей, использовать Elasticsearch. Elasticsearch – поисковой инструмент, позволяющий производительно вести полнотекстовой поиск по данным больших объёмов [3]. Этот механизм реализован благодаря встроенным механизмам токенизации и обратной индексации. Данный инструмент так же позволяет составлять запросы на агрегацию данных различных форматов, что в рамках задачи системы, позволит быстро проводить аналитику;

### **3.1.3. Агрегация данных**

Современные информационные системы сталкиваются с большими объемами данных, поступающих из различных источников. Для обработки и анализа таких данных требуется применение технологий агрегаций, которые позволяют объединять, структурировать, обогащать и преобразовывать информацию различных форматов и типов данных для дальнейшего использования.

В данной главе описан результат исследования инструментов и технологий реализации системы оценки качества проведённых

В следующем пункте рассматриваются основные методы и инструмента агрегации данных, их преимущества и ограничения.



### 3.1.4. Выбор инструмента агрегации данных

Агрегация данных – это процесс последовательного преобразования информации с целью получения сводных данных в определённом, заранее задекларированном формате.

Основные задачи агрегации включают:

- уменьшение объёма данных без потери значимой информации;
- повышение скорости обработки запросов;
- улучшение качества данных за счёт устранения дубликатов;
- преобразование данных в целевой формат хранения, обработки и анализа.

В предметной области оценки качества проведённых занятий это означает, что целевой инструмент агрегации данных должен обладать следующими свойствами:

- полнотекстовой поиск;
- агрегация по временным интервалам;
- статическая агрегация (суммирование, усреднение, нахождение минимума/максимума).

Поскольку данные планируется хранить в реляционной СУБД было принято решение сузить область исследований до двух наиболее доступных, популярных и востребованных на рынке технологий агрегации.

### 3.1.5. Сравнение Elasticsearch и PostgreSQL

Elasticsearch и PostgreSQL по своей сути разные технологии с различными сценариями типового использования, однако автор предлагает рассмотреть эти технологии в разрезе инструментов агрегации данных.

Elasticsearch – распределённый поисковый и аналитический инструмент, основанный на Apache Lucene.

Среди основных преимуществ Elasticsearch выделяют:

- полнотекстовой поиск;
- агрегация данных, получение метрик, гистограмм и сложных группировок;
- NoSql-подход и возможность работать с JSON-документами;
- поддержка горизонтального масштабирования для высоких нагрузок.

PostgreSQL в свою очередь является реляционной СУБД.

Преимущества данной технологии являются:

- поддержка ACID – аббревиатуры, обозначающей атомарность, согласованность, изолированность и надёжность транзакций [2];
- полная поддержка стандарта SQL;
- широкий спектр функций агрегаций.

Для технологий PostgreSQL и Elasticsearch образуются два целевых сценария использования в предметной области оценки качества приведённых занятий:

- PostgreSQL предпочтителен тогда, когда критически важна согласованность данных и нагрузка является умеренной;
- Elasticsearch предпочтителен в случаях, когда запланирована высокая нагрузка, т.к. позволяет горизонтально масштабироваться, и необходим полнотекстовой поиск.

Для решения задачи разработки СОКПЗ, автор предпочёл внедрить гибридный подход. То есть могут быть реализованы оба варианта, в случае если необходимо масштабировать решения в условиях большой нагрузки.

Базовое состояние такого подхода подразумевает, что данные будут храниться и агрегироваться СУБД PostgreSQL, но, если необходимо, будет представлена возможность подключить инструмент миграции данных в Elasticsearch для большей нагрузки.

## **3.2. ТРЕБОВАНИЕ К ИНСТРУМЕНТУ РЕАЛИЗАЦИИ СЕРВЕРНОГО ПРИЛОЖЕНИЯ**

К инструменту реализации серверной части, автором описаны следующие требования:

- статическая типизация со строгой системой типов;
- высокая производительность;
- широкий и доступный набор инструментов, библиотек и фреймворков;
- открытый исходный код.

### **3.2.1. Выбор языка программирования**

После составления нефункциональных требований к системе, проектирования диаграммы последовательностей и схемы данных целевого решения, следующим шагом разработки программного продукта является выбор средств реализации серверного приложения для СОКПЗ.

Поскольку автор является разработчиком на платформе Java, выбор серверного языка программирования ограничивается двумя самыми популярным решениями в этой области – Java и Kotlin

### **3.2.2. Сравнение Java и Kotlin**

Java и Kotlin – два основных языка программирования для разработки на платформе JVM [17]. Эти языки позволяет разрабатывать Android приложения и серверные решения, а компания Google и вовсе сделала Kotlin основным языком разработки для Android-систем[<https://www.forbes.ru/tehnologii/376507-google-po-russki-pochemu-kompaniya-perevela-android-na-yazyk-ot-rossiyskih>].

Язык Kotlin по своей сути является наследником языка Java, привнес в него изменения, напрашивающиеся для современного языка программирования. Среди таких изменений можно выделить:

- null-безопасность – NullPointerException, ошибка нулевой ссылки на объект в виртуальной области памяти JVM, является одной из самых частых ошибок [4]. Защита от этого исключения появилась на уровне компилятора в языке Kotlin;
- лаконичный синтаксис – язык программирования Kotlin внедрил в языковые средства множество упрощений языковых структур, называемых «синтаксическим сахаром». Это позволяет писать короткий, содержательный человекочитаемый код, не жертвуя при этом производительностью;
- мультипарадигменность – несмотря на то, что в языке программирования Kotlin, по чисто техническим причинам, не может быть реализована хвостовая рекурсия, что не позволяет его считать функциональным языком программирования, Kotlin позволяет описывать функциональные структуры, передавая функции как объекты.

Так же стоит отметить, что Kotlin является абсолютно совместимым с языком программирования Java, из этого вытекает, что в одной кодовой базе могут взаимодействовать файлы формата «.java» с файлами формата «.kt».

Исходя из описанных требований и перечисленных преимуществ языка программирования Kotlin, в качестве инструментов реализации серверной части системы, автором принято решения использовать следующей стек технологий:

- Kotlin – высокоуровневый ЯП;
- Spring Boot – серверный фреймворк;
- Spring Security – проект для обеспечения безопасности приложения;
- Spring Data Jpa – проект для обеспечения взаимодействия с реляционной СУБД;

- Elasticsearch – поисковая система, позволяющая составлять сложные агрегаты.

Spring – фреймворк, поддерживающий целый ряд языков семейства JVM, предоставляет инфраструктуру, для написания, как правило, серверных веб-приложений. Spring представляет из себя целую экосистему библиотек, проектов и расширений, которые дополняют уже существующую инфраструктуру совершенно новым функционалом:

- Spring Security – позволяет настроить защиту веб-приложения от несанкционированного доступа [18];
- Spring Web – расширение, предоставляющее инфраструктуру для создания сетевого взаимодействия между серверами;
- Spring Data \* – расширение, предоставляющее инфраструктуру для взаимодействия с различными СУБД;
- Spring Boot – проект, которому необходимо уделить отдельное внимание. Spring Boot не расширяет функционал, но представляет автоматическую конфигурацию инфраструктуры приложения, встроенный контейнер сервлетов для запуска приложения, и систему стартеров и транзитивных зависимостей, которая позволяет не заботиться о совместимости версий используемых библиотек [19]. Таким образом, загрузив проект из специализированного ресурса, можно фактически сразу запустить веб-приложение.

### **3.3. ВЫВОДЫ ПО ГЛАВЕ 3**

В текущей главе были описаны ключевые аспекты выбора инструментов реализации СОКПЗ.

В результате были сформированы требования к инструментам реализации, проведён сравнительный анализ и выбран целевой инструмент для пользовательского интерфейса, серверного приложения и СУБД.

## ГЛАВА 4. РЕАЛИЗАЦИЯ

После определения с выбором основных технических средств, автор приступил к разработке целевой системы. На рисунке 4.1 представлена схема архитектуры решения СОКПЗ.

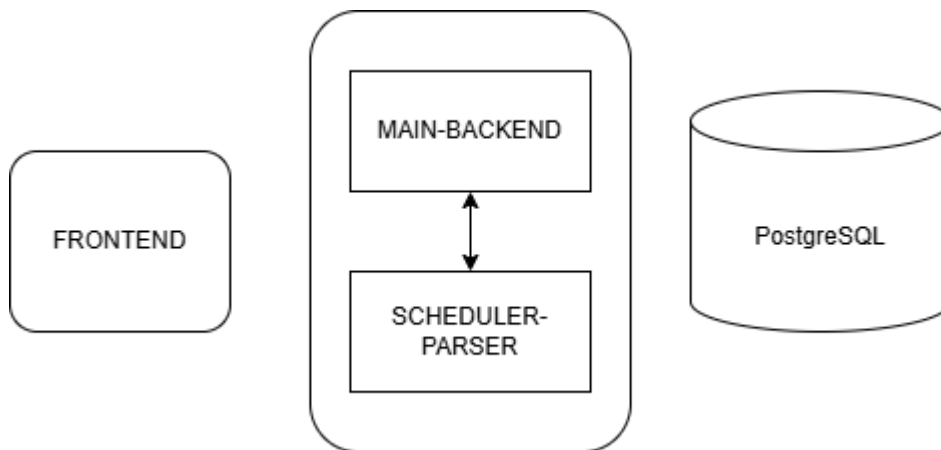


Рис. 4.1. Общая архитектурная схема решения

На данной схеме представлены 4 целевых элемента архитектуры решения.

Frontend - пользовательский интерфейс, реализованный с помощью фреймворка Next.js, языка JavaScript и его диалекта TypeScript.

Main-backend - основное серверное приложение. Отвечает за весь функциональный спектр - авторизация, расписание, тестирование и т.д.

Scheduler-Parser - дополнительное, но обязательно серверное приложение. Его ответственность - преобразовывать файлы формата pdf в целевую json-структуру для дальнейшего сохранения и отображения расписания.

PostgreSQL - целевая СУБД. В ней хранятся фактически все данные всех процессов системы.

Далее в настоящей главе будут подробно описаны нюансы и детали реализации каждого компонента

На рисунке 4.2 представлено древо страниц пользовательского интерфейса.

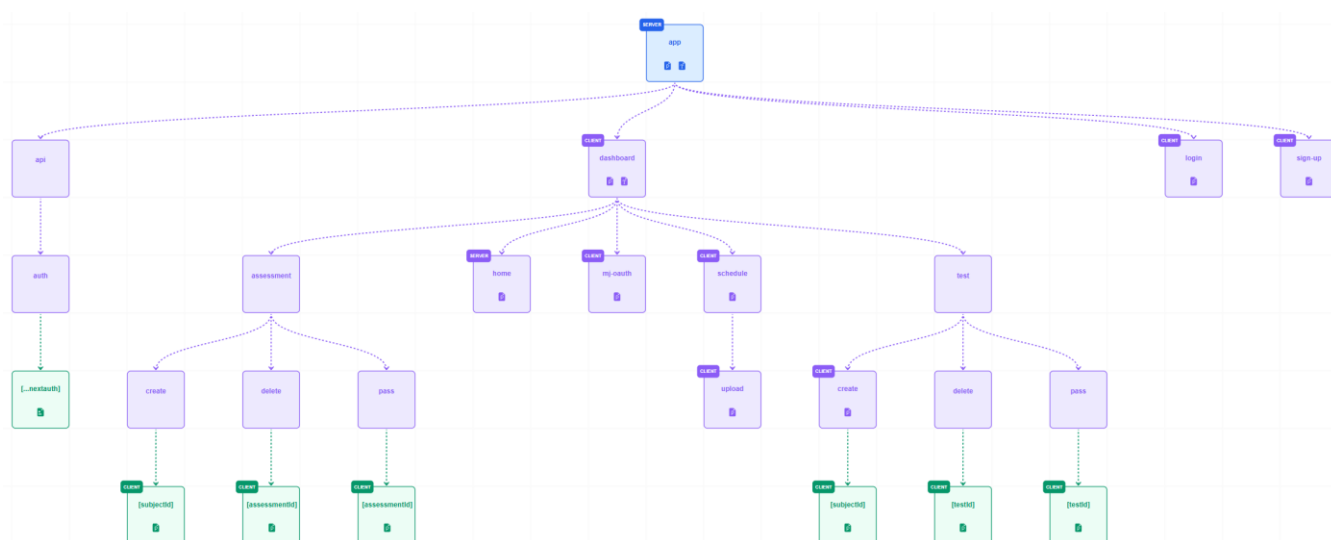


Рис. 4.2. Древо страниц пользовательского интерфейса СОКПЗ

Так же стоит отметить, что в фреймворке для языка JavaScript Next.js, директория “/app” является основной директорией для организации структуры маршрутов web-страниц. Каждая вложенная директория внутри директории “/app” представляет собой URL-путь. Файлы page.tsx внутри таких директорий - являются непосредственным компилируемым визуальным содержанием web-страницы по относительному директории «/app» адресу пути. Пример такой структуры приведён на рисунке 4.3

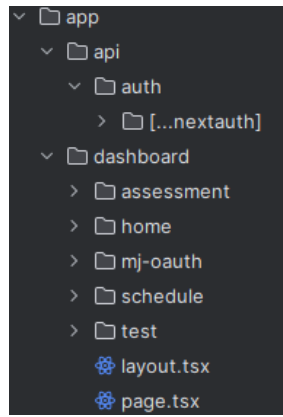


Рис. 4.3. Организационная структура проекта пользовательского интерфейса СОКПЗ

Разобравшись с организационной структурой маршрутов web-страниц, далее в этой главе будут приведены особенности реализации целевой функциональности СОКПЗ.

#### 4.1.1. Расписание занятий

Для того чтобы в пользовательском интерфейсе была возможность удобно создавать и проходить тестирование, проставлять оценки, и совершать удобную навигацию по спискам занятий, автором была разработана система отображения расписания (см. Рис 4.4).

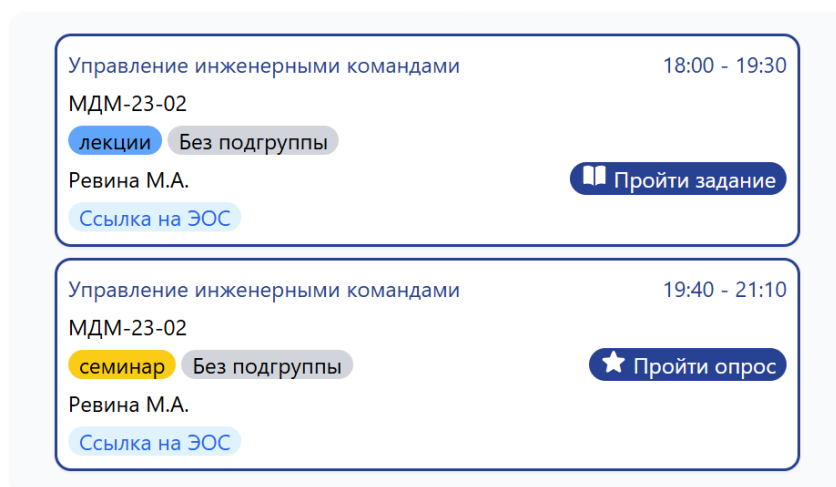


Рис. 4.4. Расписание в пользовательском интерфейсе СОКПЗ



Данный компонент представляет из себя универсальную абстракцию, благодаря которой ученики могут проходить задания преподавателей и их опросы, а преподаватели и внешний аудит управлять (создавать, редактировать, удалять) такие задания и опросы.

В текущем виде предусмотрен один вид создания расписаний - загрузка специального .pdf файла, сгенерированного программой «Ректор-ВУЗ» и поставляемого всем студентам вначале учебного периода. СОКПЗ самостоятельно исключит дубликаты, и уведомит пользователя, что такое расписание уже существует. Форма для загрузки расписания представлена на рисунке 4.5.

В результате успешного заполнения формы, на вспомогательное серверное приложение «scheduler-parser» отправляется целевой pdf-файл с расписанием в бинарном формате для дальнейшего преобразования в целевой формат хранения.

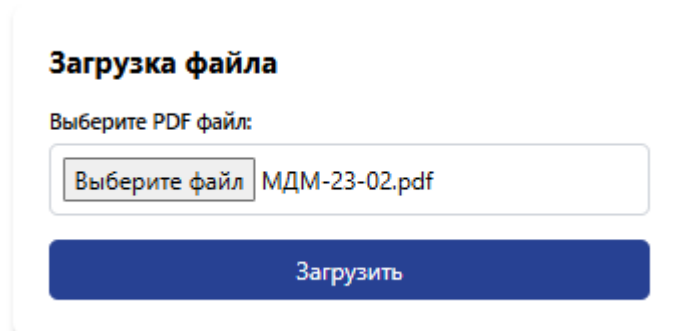


Рис. 4.5. Форма для загрузки расписания в пользовательском интерфейсе СОКПЗ

Также в будущем планируется разработка API для синхронизации расписаний с общим репозиторием по определённому cron-таймеру.

Для того, чтобы данные о занятиях в .pdf файле могли быть сохранены в общеупотребимом формате, система СОКПЗ сначала создает структуру в формате JSON, разворачивает данный JSON в список занятий, сортирует

занятия и сохраняет в реляционную базу данных. Подробнее данный процесс будет рассмотрен в главе, посвящённой средствам реализации.

#### 4.1.2. Тестирование и опросы

На рисунке 4.6 представлен компонент расписания, доступный для роли «Преподаватель» и «Аудитор» для создания элементов заданий и опросов.

The image shows a user interface for managing a schedule. It contains two distinct time slots, each with a header and a list of controls.

**Top Slot:**

- Header: Управление инженерными командами (18:00 - 19:30)
- Course: МДМ-23-02
- Activity: лекции (highlighted in blue)
- Group: Без подгруппы
- Teacher: Ревина М.А.
- Controls: [Ссылка на ЭОС](#), [Удалить задание](#), [Создать опрос](#)

**Bottom Slot:**

- Header: Управление инженерными командами (19:40 - 21:10)
- Course: МДМ-23-02
- Activity: семинар (highlighted in yellow)
- Group: Без подгруппы
- Teacher: Ревина М.А.
- Controls: [Ссылка на ЭОС](#), [Создать задание](#), [Удалить опрос](#)

Рис. 4.6. Управление тестами и опросами

На рисунке 4.7 представлена форма для создания элементов тестирования. Данная форма позволяет задать необходимое количество вопросов с различным количеством вариантов ответов. В результате успешного заполнения формы на сервер отправляются данные о тестировании, которые в дальнейшем отобразятся у группы, которой принадлежит данный элемент ежедневного расписания в соответствии с сформированной схемой данных.

## Создание теста

Вопрос 1

Вопрос 2

+

Вопрос 1

Удалить вопрос

Что является ключевым фактором успешного управления инженерной командой

Варианты ответов:

☐

Водопадная модель (Waterfall)

×

☒

Канбан-доска (Kanban)

×

☐

Диаграмма Ганта (Gantt Chart)

×

☐

Полная свобода без фиксированных процессов

×

+ Добавить вариант ответа

Сохранить тест

Рис. 4.7. Форма создания элемента тестирования

На рисунке 4.8 представлена форма для создания элементов опроса об оценке качества проведённого занятия. Данная форма позволяет задать необходимое количество вопросов, ответом на которые должна быть целочисленная оценка по 10-бальной шкале. В результате успешного заполнения формы на сервер отправляются данные об опросе, которые в дальнейшем отобразятся у группы, которой принадлежит данный элемент ежедневного расписания в соответствии с сформированной схемой данных.

Рис. 4.8. Форма создания элемента опроса оценки качества  
проведённого занятия

На рисунке 4.9 представлена форма для прохождения задания, созданного ролью «Преподаватель». Данная форма позволяет численно оценить качество и уровень вовлеченности у роли «Студент» и сохранить полученные данные для дальнейшей агрегации. В результате успешного заполнения формы на сервер отправляются данные об ответах студента, в частности ссылка на опрос, ссылка на студента, выполнявшего задание и численный результат.

## Управление инженерными командами

вторник, 1 апреля 2025 г.

семинар

Группа: МДМ-23-02	Аудитория: Не указана
Подгруппа: Без подгруппы	Время: 19:40 - 21:10
Преподаватель: Ревина М.А.	Тип занятия: семинар

Прогресс: 2 из 2 100%

---

**Вопросы теста (2)**

**1** Какой метод чаще всего используется для управления задачами в agile-командах?

- ☐ Водопадная модель (Waterfall)
- ☒ Канбан-доска (Kanban) ✓ Верно
- ☐ Диаграмма Ганта (Gantt Chart)
- ☐ Полная свобода без фиксированных процессов

**2** Что является ключевым фактором успешного управления инженерной командой?

- ☐ Жёсткий контроль каждого этапа работы
- ☒ Микроуправление и постоянные проверки ✗ Неверно
- ☐ Чёткое распределение ролей и зон ответственности
- ☐ Полное отсутствие дедлайнов

**Результат теста**

**1 из 2 (50%)**

Рис. 4.9. Форма прохождения элемента тестирования ролью «Студент»

На рисунке 4.10 представлена форма для сбора оценок качества проведённого занятия. Данная форма позволяет численно оценить субъективную оценку роли «Студент» качества проведённого занятия. В

результате успешного заполнения формы на сервер отправляются данные оценок в 10-бальной шкале, ссылка на опрос и ссылка на студента, выполнявшего задание.

**Оценка занятия**

Пожалуйста, оцените каждый вопрос по 10-бальной шкале

1. Насколько содержимое лекции соответствует заявленной теме? ★★★★★★★★ 9/10

2. Насколько содержимое лекции соответствует вашим ожиданиям? ★★★★★★★ 7/10

Отправить оценки

Рис. 4.10. Форма прохождения элемента опроса об оценке качества проведённого занятия

Для удаления элементов тестирования и опросов так же предусмотрены формы, в результате успешного заполнения которых отправляется запрос на сервер с целью удаления из базы данных таких сведений. Примеры форм приведены на рисунках 4.11 и 4.12.

**Опрос #166a3e65-3651-4f4d-806c-b00ca0f1d05f**

Создан: 01.05.2025

Удалить опрос

Вопросы (2)

1. Насколько содержимое лекции соответствует заявленной теме?

2. Насколько содержимое лекции соответствует вашим ожиданиям?

Рис. 4.11. Форма удаления элемента опроса из базы данных

## Управление инженерными командами

вторник, 1 апреля 2025 г.

Удалить задание

Группа:	МДМ-23-02	Аудитория:	Не указана
Подгруппа:	Без подгруппы	Время:	19:40 - 21:10
Преподаватель:	Ревина М.А.	Тип занятия:	семинар

### Вопросы теста (2)

1 Какой метод чаще всего используется для управления задачами в agile-командах?

- Водопадная модель (Waterfall)
- ✓ Канбан-доска (Kanban)
- Диаграмма Ганта (Gantt Chart)
- Полная свобода без фиксированных процессов

2 Что является ключевым фактором успешного управления инженерной командой?

- Жёсткий контроль каждого этапа работы
- Микроуправление и постоянные проверки
- ✓ Чёткое распределение ролей и зон ответственности
- Полное отсутствие дедлайнов

Рис. 4.12. Форма удаления элемента тестирования из базы данных

### 4.1.3. Модульный журнал. OAuth 2.0 провайдер

Для того чтобы однозначно идентифицировать сущность студента и привязать данные студента к сущности пользователя системы, автором системы была разработана интеграция с модульным журналом «МГТУ СТАНКИН».

Сервис «Модульный журнал» может быть использован для как средство аутентификации студентов МГТУ Станкин и получения их персональных данных, таких как имя, фамилия, номер студенческого билета, группы и т.д. посредством протокола OAuth 2.0.

На данный момент автоматическая регистрация сторонних сервисов в системе «Модульный журнал» не реализована. По этой причине автор связался с главным разработчик для регистрации СОКПЗ.

Процедура аутентификации выглядит следующим образом [6]:

1. На сайте необходимо разместить ссылку «[https://lk.stankin.ru/webapi/oauth/authorize?response\\_type=code&client\\_id=\[client\\_id\]&redirect\\_uri=\[redirect\\_uri\]](https://lk.stankin.ru/webapi/oauth/authorize?response_type=code&client_id=[client_id]&redirect_uri=[redirect_uri])», где переменные `client_id` – идентификатор интегрируемой системы, а `redirect_url` – адрес сервера пользовательского интерфейса, на который необходимо вернуть управление.
2. При клике на ссылку из п.1 пользователь будет перенаправлен на страницу запроса разрешения передачи данных интегрируемому сервису.
3. В случае введения корректных логина и пароля студента, пользователь будет перенаправлен на `[redirect_url]`, при этом к строке запроса будет добавлен параметр `[code]`, при помощи которого появится возможность получить информацию о пользователе. `/task/{taskId}` - получение перечня вопросов по уникальному идентификатору конкретного занятия (`taskId`).
4. Пользовательский интерфейс отправляет запроса на «main» серверное приложение, которое в свою очередь отправляет запрос в систему «Модульный журнал». Пример запроса приведён на рисунке 4.13.
5. В ответ пользовательский интерфейс получает данные студента.



```
POST /webapi/oauth/token HTTP/1.1
Host: lk.stankin.ru
Content-type: application/x-www-form-urlencoded
Content-Length: [длина тела запроса]

code=[полученный code]&client_id=[ваш client_id]&client_secret=[ваш client_secret]
```

Рис 4.13. Пример запроса в систему «Модульный журнал»

Пример данных студента, полученных от системы «Модульный журнал» приведён на рисунке 4.14.

```
{
  "access_token": "647801fa-bd52-4d85-87d1-01e80e00b380",
  "token_type": "bearer",
  "userInfo": {
    "name": "Иван",
    "surname": "Иванов",
    "patronym": "Иванлыич",
    "stgroup": "ИДБ-00-00",
    "cardid": "1234567"
  }
}
```

Рис. 4.14. Пример данных студента, полученных из системы «Модульный журнал»

В результате прохождения процедуры аутентификации, в пользовательском интерфейсе отобразятся данные студента (рисунок 4.15). Так же в данной форме доступна кнопка привязывания данных студента с сущностью пользователя системы СОКПЗ.

**Данные авторизации** bearer

2f421118-418a-4dae-aedf-372fa926e5b2

**Романов Илья Олегович**  
Студент

**Учебная информация**

Группа  
**ИДБ-19-10**

ID карты  
**119161**

**Персональные данные**

Фамилия  
**Романов**

Имя  
**Илья**

Отчество  
**Олегович**

Данные обновлены: 01.05.2025

Закрепить данные студента с текущей учётной записью? **Подтвердить**

Рис. 4.15. Пример данных студента, полученных из сервиса  
«Модульный журнал»

Без выполнения данной процедура, пользователю не удастся выполнить тестирование и пройти опрос контроля качества проведённых занятий, что обеспечивает прозрачность процесса.

Также в данном компоненте доступна кнопка изменения данных о группе студента, т.к. существует ситуации, в которых такие сведения могут быть неактуальны, а расписание отображается только для группы, указанной в данной форме.

## 4.2. РЕАЛИЗАЦИЯ СЕРВЕРНОЙ ЧАСТИ

Серверная часть СОКПЗ имеет два серверных приложения:

1. Основное серверное приложение - представляет из себя монолит, имеющий REST API для интеграции с пользовательским

интерфейсом. В нём реализованы HTTP - конечные точки, отвечающие за авторизацию и аутентификацию, управление расписанием, тестированием и опросами, а также генерацией целевых отчётов.

2. Дополнительное серверное приложение - является утилитарным и предназначено для преобразования целевого файла формата pdf в структуру данных формата json с общими сведениями о расписании. После преобразования, сервис отправляет полученные данные в основное серверное приложение, где затем происходит финальное преобразование и сохранение в базу данных.

Разобравшись с общей функциональной принадлежностью, далее будут описаны особенности реализации серверной части СОКПЗ

#### 4.2.1. Авторизация и аутентификация

Для того чтобы изолировать систему от несанкционированного доступа, в ней была реализована система авторизации и аутентификации. Пользователю предоставляется форма для ввода логина и пароля, а также отдельная форма для регистрации с последующей модерацией. Форма представлена на рисунке 4.16.

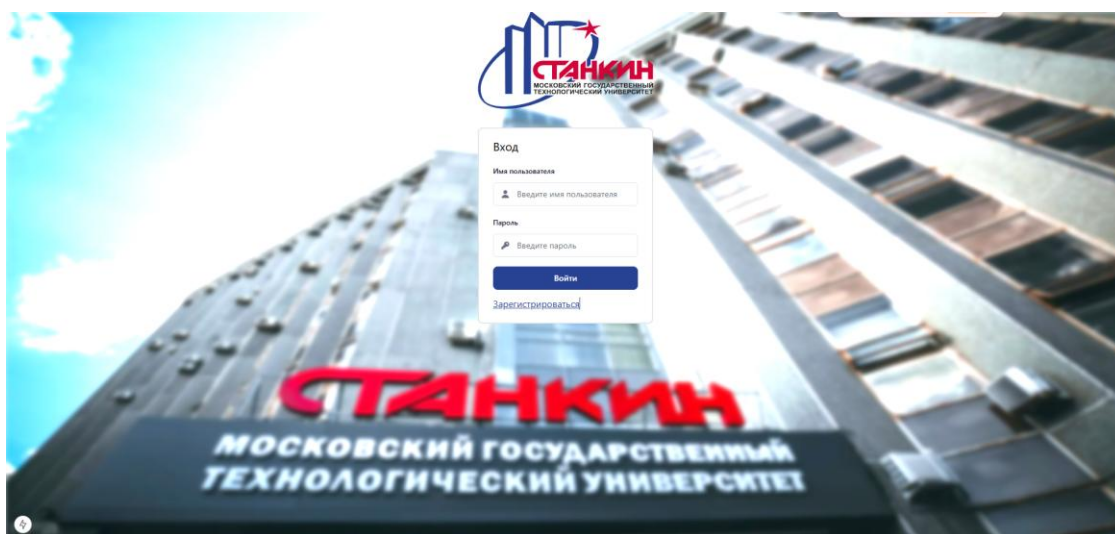



Рис. 4.16. Форма для входа в пользовательский интерфейс СОКПЗ

При регистрации новому пользователю необходимо в течение суток привязать свой аккаунт к данным студента. Это реализовано благодаря интеграции с системой «Модульный журнал». Если этого не сделать, реализован асинхронный модуль, который удаляет из БД пользователей с ролью «Студент», без привязки к данным конкретного студента. В результате взаимодействия с системой «Модульный журнал», система СОКПЗ получает необходимые данные о пользователе, которые позволяют определить фамилию, имя, отчество, группу и номер студенческого билета. Компонент пользовательского интерфейса, отображающий описанные данные приведён на рисунке 4.17.

После регистрации пользователь переадресуется на страницу входа. Если пользователь с введёнными учебными данными найден в базе данных системы, браузеру вернётся JWT (JavaScript Web Token) - в котором, в зашифрованном виде возвращаются сведения о роли пользователя и других технических сведений. Данный токен сохраняется в контексте безопасности приложения, и отправляется с каждым запросом, связанным с другими целевыми бизнес - процессами.

**Данные авторизации** bearer  
2f421118-418a-4dae-aedf-372fa926e5b2

 **Романов Илья Олегович**  
Студент

Учебная информация	Персональные данные
Группа <b>ИДБ-19-10</b>	Фамилия <b>Романов</b>
ID карты <b>119161</b>	Имя <b>Илья</b>
	Отчество <b>Олегович</b>

Данные обновлены: 16.05.2025

Закрепить данные студента с текущей учётной записью? **Подтвердить**

Рис. 4.17. Компонент с данным студента, полученными в результате интеграции с системой «Модульный журнал»

Таким образом реализована система авторизации и аутентификации, позволяющая однозначно идентифицировать пользователя в пользовательском интерфейсе приложения и персонифицировать, и изолировать его для отдельных ролей.

#### **4.2.2. Сетевое взаимодействие серверного приложения с пользовательским интерфейсом**

Программная архитектура целевой системы оценки качества проведённых занятий имплементирует классическую трёхуровневую модель разделения приложения на три логических и физических независимых уровня, каждый из которых несёт строго определённую ответственность [14]. Такая структура обеспечивает модульность, масштабируемость и лёгкость поддержки системы.

Так как пользовательский интерфейс был рассмотрен ранее, автором предложено осветить аспект сетевого взаимодействия серверного приложения с пользовательским интерфейсом.

Для реализации сетевого взаимодействия, за основу был взят REST API (Representational State Transfer Application Programming Interface) - архитектурный стиль для создания сетевого взаимодействия между компонентами сети, которые используются стандартные HTTP-методы. Данный архитектурный стиль основан на ресурсной модели, где каждый объект представлен уникальными URL, а операции с ним выполняются через HTTP - запросы.

Рассмотрим в качестве примера реализации взаимодействия между пользовательским интерфейсом и серверным приложением, программный интерфейс для CRUD (Create, Read, Update, Delete) операций над результатами выполнения тестовых заданий, исходный код которого приведён на рисунке 4.18.

```

@RestController  ± manysleeplover1
@RequestMapping(±"/task")
class TestTaskController(
    private val taskForClassService: TaskForClassService
) {

    @PostMapping(±"/task")  ± manysleeplover1
    fun saveTestTask(@RequestBody request: SaveTaskForClassRequest) =
        taskForClassService.saveTaskForClass(request)

    @GetMapping(±"/{taskId}")  ± manysleeplover1
    fun getTaskForClassById(@PathVariable("taskId") taskId: String) =
        taskForClassService.getTaskForClassById(taskId)

    @DeleteMapping(±"/{taskId}")  ± manysleeplover1
    @ResponseStatus(HttpStatus.OK)
    fun deleteTaskForClassById(@PathVariable("taskId") taskId: String) =
        taskForClassService.deleteTaskForClassById(taskId)

    @PostMapping(±"/passed")  ± manysleeplover1
    @ResponseStatus(HttpStatus.OK)
    fun savePassedTestResult(@RequestBody passedTestResult: PassedTestResult){
        taskForClassService.savePassedTestResult(passedTestResult)
    }
}

```

Рис. 4.18. Исходный код API для управления результатами прохождения тестов

Данный код представляет собой REST-контроллер на языке Kotlin, реализующий API управления результатами выполнения тестовыми заданиями. Данный контроллер реализован благодаря фреймворку Spring и его поддержке аннотаций, благодаря чему, такие типовые задачи, как написание REST API - контроллеров вырождается, по сути, в декларативное программирование.

В общей структуре исходного кода можно выделить аннотации `@RestController`, `.*Mapping` и `@ResponseStatus`.

Аннотация `@RestController` автоматически создаёт объект в контексте приложения Spring и встраивает его в иерархию обработки HTTP-запросов. Благодаря данной аннотации нет необходимости вручную создавать и описывать подробные конфигурации конечной точки приёма HTTP-запросов.

Аннотации `@GetMapping`, `@PostMapping`, `@DeleteMapping` и т.д. определяют сигнатуру принимаемого HTTP-запроса, в частности его метод и URI для доступа к конечному ресурсу [10].

Данный контроллер определяет четыре конечные точки:

1. POST /task - сохранение перечня вопросов для определённого занятия в расписании.
2. GET /task/{taskId} - получение перечня вопросов по уникальному идентификатору конкретного занятия (taskId).
3. DELETE /task/{taskId} - удаление перечня вопросов по уникальному идентификатору конкретного занятия.
4. POST /task/passed - сохранение результата прохождения ролью «Студент» перечня вопросов конкретного занятия в расписании.

Представленный контроллер является исчерпывающе репрезентативным. Подходы, применённые в данном контроллере, применяются во всей схеме сетевого взаимодействия серверного приложения.

Таким образом остаётся лишь рассмотреть аспект сетевого взаимодействия разработанного серверного приложения СУБД PostgreSQL.

#### **4.2.3. Сетевое взаимодействие серверного приложения с СУБД PostgreSQL**

Автор ранее упоминал, что СОКПЗ имплементирует трёхуровневую архитектуру программного обеспечения. То есть инкапсулируются в отдельные физические и логические уровни пользовательский интерфейс, серверное приложение и СУБД. Исходя из этого необходимо так же описать взаимодействие с СУБД PostgreSQL.

В серверном приложении СОКПЗ взаимодействие с СУБД PostgreSQL происходит на достаточном высоком уровне абстракции. Дело в том, что целевым фреймворком выбран Spring Framework. В экосистеме данного инструмента существует широчайший спектр проектов, одним из которых является Spring Data JPA и его расширение spring-boot-starter-data-jpa.

JPA (Java Persistence API) - это ORM (Object-Relational Mapping)-спецификация, которая предоставляет интерфейс для Java, позволяющий работать с реляционными базами данных в объектно-ориентированном стиле. Она

является частью спецификации JakartaEE и представляет набор интерфейсов и аннотация для отображения Java-объектов на таблицы БД и наоборот.

JPA по сути инкапсулирует другой низкоуровневый инструмент JDBC (Java Database Connectivity). JDBC состоит из двух главных компонентов: API - поддерживает связь между Java-приложением и менеджером JDBC, и Драйвера JDBC, которые поддерживают связь между менеджером JDBC и драйвером базы данных. Для установки соединения необходим URL сервера базы данных, учётные данные для авторизации и название целевой базы данных.

Таким образом, JDBC представляет из себя ничто иное, как набор TCP/IP соединений между приложением и СУБД, ведущие взаимодействие по протоколу PostgreSQL Protocol, который является бинарным.

Погрузившись и разобравшись в подходах сетевого взаимодействия серверных приложений Java и СУБД PostgreSQL, рассмотрим в качестве примера прикладной реализации взаимодействия API для забора данных о семестровом расписании. На рисунке 4.19 приведён исходный код компонента взаимодействия с СУБД PostgreSQL.

```
@Repository
interface SemesterScheduleRepository : JpaRepository<SemesterScheduleEntity, String> {

    fun findByFirstClassDateAndLastClassDateAndStgroupAndVersionDate(
        firstClassDate: LocalDate,
        lastClassDate: LocalDate,
        stgroup: String,
        versionDate: String?
    ) : List<SemesterScheduleEntity>

    @Query(nativeQuery = true, value = """SELECT s.id, s.version_date
        FROM public.semester_schedule s
        where s.stgroup = :stgroup
        and :date between s.first_class_date and s.last_class_date""")
    fun findByStgroupAndDate(
        @Param("stgroup") stgroup: String,
        @Param("date") date: LocalDate) : List<SemesterScheduleIdAndVersionDateProjection>
}
```

Рис. 4.19. Исходный код взаимодействия с СУБД PostgreSQL



Интерфейс «SemesterScheduleRepository» представляет собой репозиторий для работы с сущностями семестровых расписаний «SemesterScheduleEntity» в СОКПЗ. Он расширяет стандартный интерфейс JPA JpaRepository из проекта Spring Data Jpa. Данная имплементация позволяет неявно создать целый ряд методов доступа к данным из таблицы «semster\_schedule» несмотря на то, что в явном виде в репозитории объявлено всего два метода. Это происходит за счёт того, что интерфейс JpaRepository сам в свою очередь наследует ряд других интерфейсов, имплементация которых генерируется для сущности, указанной в сигнатуре наследования, в нашем случае «SemesterScheduleEntity». Неполная иерархия наследования интерфейса JpaRepository, включая методы, представлена на рисунке 4.20.

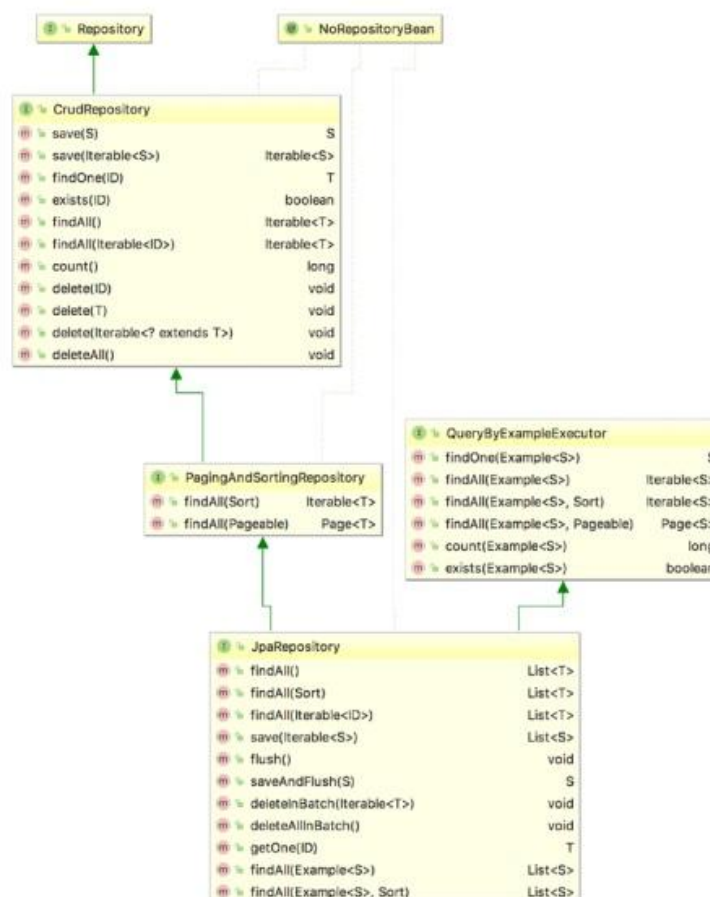


Рис. 4.20. Иерархия наследования интерфейса JpaRepository в Spring Data JPA

Также на рисунке 4.19 можно обратить внимание, что для одного из методов указана аннотация `@Query` с включённым в её сигнатуру явным SQL-запросом, тогда как метод с очень длинным названием «`findByFirstClassDateAndLastClassDateAndStgroupAndVersionDate`» не имеет приведённой аннотации в явном виде, программа при этом функционирует в абсолютно штатном режиме. Это возможно, благодаря механизму рефлексии и библиотеке генерации кода, активно используемой в экосистеме Spring. Особенность заключается в том, что SQL-запрос генерируется из полного названия метода и сущности, для которой создан репозиторий [9]. Таким образом метод «`findByFirstClassDateAndLastClassDateAndStgroupAndVersionDate`» ведёт поиск в таблице по полям `FirstClassDate`, `LastClassDate`, `Stgroup` и `VersionDate`, таблицы `SemestewrSchedule`.

Полный перечень поддерживаемых ключевых слов внутри имён методов приведён на рисунке 4.21

Ключевое слово	Образец	Фрагмент JPQL
Distinct	<code>findDistinctByLastNameAndFirstname</code>	<code>select distinct _ where x.lastname = ?1 and x.firstname = ?2</code>
And	<code>findByLastNameAndFirstname</code>	<code>... where x.lastname = ?1 and x.firstname = ?2</code>
Or	<code>findByLastNameOrFirstname</code>	<code>... where x.lastname = ?1 or x.firstname = ?2</code>
Is, Equals	<code>findByFirstname,</code> <code>findByFirstnameIs, findByFirstnameEquals</code>	<code>... where x.firstname = ?1 (или ... where x.firstname IS NULL если аргумент null)</code>
Between	<code>findByStartDateBetween</code>	<code>... where x.startDate between ?1 and ?2</code>
LessThan	<code>findByAgeLessThan</code>	<code>... where x.age &lt; ?1</code>
LessThanEqual	<code>findByAgeLessThanEqual</code>	<code>... where x.age &lt;= ?1</code>
GreaterThan	<code>findByAgeGreaterThan</code>	<code>... where x.age &gt; ?1</code>
GreaterThanEqual	<code>findByAgeGreaterThanEqual</code>	<code>... where x.age &gt;= ?1</code>
After	<code>findByStartDateAfter</code>	<code>... where x.startDate &gt; ?1</code>
Before	<code>findByStartDateBefore</code>	<code>... where x.startDate &lt; ?1</code>
IsNull, Null	<code>findByAge(Is)Null</code>	<code>... where x.age is null</code>
IsNotNull, NotNull	<code>findByAge(Is)NotNull</code>	<code>... where x.age is not null</code>
Like	<code>findByFirstnameLike</code>	<code>... where x.firstname like ?1</code>
NotLike	<code>findByFirstnameNotLike</code>	<code>... where x.firstname not like ?1</code>
StartingWith	<code>findByFirstnameStartingWith</code>	<code>... where x.firstname like ?1 (параметр связан с добавленным %)</code>
EndingWith	<code>findByFirstnameEndingWith</code>	<code>... where x.firstname like ?1 (параметр связан с добавленным префиксом %)</code>
Containing	<code>findByFirstnameContaining</code>	<code>... where x.firstname like ?1 (привязка параметра заключена в %)</code>
OrderBy	<code>findByAgeOrderByLastNameDesc</code>	<code>... where x.age = ?1 order by x.lastname desc</code>
Not	<code>findByLastNameNot</code>	<code>... where x.lastname &lt;&gt; ?1 (или ... where x.lastname IS NOT NULL если аргумент null)</code>
In	<code>findByAgeIn(Collection&lt;Age&gt; ages)</code>	<code>... where x.age in ?1</code>
NotIn	<code>findByAgeNotIn(Collection&lt;Age&gt; ages)</code>	<code>... where x.age not in ?1</code>
True	<code>findByActiveTrue()</code>	<code>... where x.active = true</code>
False	<code>findByActiveFalse()</code>	<code>... where x.active = false</code>
IgnoreCase	<code>findByFirstnameIgnoreCase</code>	<code>... where UPPER(x.firstname) = UPPER(?1)</code>

Рис. 4.21. Поддерживаемые ключевые слова внутри имён методов JPA

Таким образом, представленный репозиторий является исчерпывающе репрезентативным. Подходы, применённые в данном репозитории, применяются во всей схеме взаимодействия серверного приложения и СУБД.

#### 4.2.4. Преобразования файлов расписания в целевой формат хранения

СОКПЗ представляет API для преобразования файла, содержащего расписание, поставляемого студентам «МГТУ «СТАНКИН» в формате PDF в формат JSON.

Данный файл генерируется программой «Ректор-ВУЗ». «Ректор-ВУЗ» - ПО, предназначенное для составления расписаний в высших учебных заведениях [1]. Готовое расписание можно сгенерировать в формате Microsoft Word, Excel или HTML. Пример готового расписания представлен на рисунке 4.22.

ИДМ-23-08

	8:30 - 10:10	10:20 - 12:00	12:20 - 14:00	14:10 - 15:50	16:00 - 17:40	18:00 - 19:30	19:40 - 21:10	21:20 - 22:50
Понедельник					Системы менеджмента информационной безопасности. Тихомирова В.Д. семинар. 308. [24.03, 31.03]	Системы менеджмента информационной безопасности. Симонов М.Ф. лекции. 0209. [10.02, 17.02] Системы менеджмента информационной безопасности. Тихомирова В.Д. семинар. 308. [24.03, 31.03]	Системы менеджмента информационной безопасности. Симонов М.Ф. лекции. 0209. [10.02-31.03 к.н.]	
Вторник					Стоимостной инжиниринг. Крючкова Е.В. семинар. [18.02-01.04 к.н.]	Архитектура, программное обеспечение и безопасность автоматизированных систем. Кузовкин К.Н. лекции. [11.02-01.04 к.н.]	Архитектура, программное обеспечение и безопасность автоматизированных систем. Колеснин В.А. лабораторные занятия. (А). [11.02-25.03 к.н.] Архитектура, программное обеспечение и безопасность автоматизированных систем. Колеснин В.А. лабораторные занятия. (Б). [18.02-01.04 к.н.] Системы менеджмента информационной безопасности. Тихомирова В.Д. лабораторные занятия. (Б). [25.02-25.03 к.н.] Системы менеджмента информационной безопасности. Тихомирова В.Д. лабораторные занятия. (А). [04.03-01.04 к.н.]	
Среда								
Четверг					Стоимостной инжиниринг. Крючкова Е.В. семинар. [20.03-03.04 к.н.]			
Пятница					Стоимостной инжиниринг. Крючкова Е.В. лекции. [14.02, 04.04]	Стоимостной инжиниринг. Крючкова Е.В. лекции. [14.02-04.04 к.н.]		
Суббота								

Рис. 4.22. Результат работы программы «Ректор-ВУЗ»

Scheduler-parser является самостоятельным серверным приложением, представляющим HTTP - конечную точку, которая принимает на вход файл, содержащий расписание.

Данное серверное приложение реализовано на языке TypeScript, серверном фреймворке Node.js и инкапсулирует в себе библиотеку с открытым исходным кодом «rector-scheduler-parser». Библиотека «rector-scheduler-parser» представляет API для преобразования файлов в бинарном и PDF-формате.

Результатом работы данного серверного компонента является структура данных, содержащая массив объектов JSON, которые в свою очередь содержат сведения о предмете. В эти сведения входят:

1. Номер группы.
2. Название предмета.
3. Аудитория проведения занятия.
4. Временной период проведения занятия.
5. Чётность недель проведения занятия.
6. Время начала проведения занятия.
7. Время окончания проведения занятия.
8. Подгруппа.
9. Имя преподавателя.
10. Тип занятия.

Пример результата работы серверного компонента представлен на рисунке 4.23

```

{
  "stgroup": "МДМ-23-02",
  "subject": "Инструментальное обеспечение машиностроительных производств",
  "audience": "0622",
  "periods": [
    {
      "start_date": "10.02",
      "end_date": "31.03",
      "repeat": "к.н."
    },
    {
      "start_date": "14.04",
      "end_date": "05.05",
      "repeat": "к.н."
    }
  ],
  "dates": [
  ],
  "start_time": "18:00",
  "end_time": "19:30",
  "group": "Без подгруппы",
  "teacher": "Сотова Е.С.",
  "type": "лекции"
},
{
  "stgroup": "МДМ-23-02",
  "subject": "Инструментальное обеспечение машиностроительных производств",
  "audience": "235(в) - ТехП 6",
  "periods": [
  ],
  "dates": [
    "10.03",
    "24.03",
    "14.04",
    "28.04"
  ],
  "start_time": "19:40",
  "end_time": "21:10",
  "group": "(А)",
  "teacher": "Сотова Е.С.",
  "type": "лабораторные занятия"
},

```

Рис. 4.23. Результат работы серверного приложения «scheduler-parser»

Приведённая структура данных хоть и является исчерпывающей, однако не позволяет сохранить каждое занятие в реляционной СУБД и вести быстрый оптимизированный поиск, например по календарному числу даты проведения занятия. Для достижения этой цели, автором разработан серверный компонент, который дополнительно разворачивает каждый элемент массива, сгенерированный из файла PDF - расписания в сортированный по дате и

времени проведения список занятий, благодаря чему данные после цепочки преобразований могут храниться в виде, приведённом на рисунке 4.24.

id	date	group	subject	id	date	group	subject	id	date	group	subject
4087468-4088-4035...	2025-02-10	МДР-23-02	Инструментальное обеспечение машиностроительных производств	0022	10:00	19:30	без подгруппы	Сотова Е.С.	лекция	d022d05a-fc21-41f1-80ed-b3ce1e1f49e	null
5082988-4053-405f...	2025-02-10	МДР-23-02	Инструментальное обеспечение машиностроительных производств	0022	19:40	21:10	без подгруппы	Сотова Е.С.	семинар	d022d05a-fc21-41f1-80ed-b3ce1e1f49e	null
4070087-4096-4171...	2025-02-11	МДР-23-02	Управление инженерными командами	0209	10:00	19:30	без подгруппы	Резина Н.А.	лекция	d022d05a-fc21-41f1-80ed-b3ce1e1f49e	null
9409013-0846-440f...	2025-02-17	МДР-23-02	Инструментальное обеспечение машиностроительных производств	0022	10:00	19:30	без подгруппы	Сотова Е.С.	лекция	d022d05a-fc21-41f1-80ed-b3ce1e1f49e	null
cefd3063-2f0a-41b1...	2025-02-17	МДР-23-02	Инструментальное обеспечение машиностроительных производств	0022	19:40	21:10	без подгруппы	Сотова Е.С.	семинар	d022d05a-fc21-41f1-80ed-b3ce1e1f49e	null
19d2d902-0f81-4c05...	2025-02-18	МДР-23-02	Управление инженерными командами	0209	10:00	19:30	без подгруппы	Резина Н.А.	лекция	d022d05a-fc21-41f1-80ed-b3ce1e1f49e	null
37d0d94b-cf09-47f3...	2025-02-18	МДР-23-02	Управление инженерными командами	0209	19:40	21:10	без подгруппы	Резина Н.А.	семинар	d022d05a-fc21-41f1-80ed-b3ce1e1f49e	null
67ee8770-8691-408a...	2025-02-24	МДР-23-02	Инструментальное обеспечение машиностроительных производств	0022	10:00	19:30	без подгруппы	Сотова Е.С.	лекция	d022d05a-fc21-41f1-80ed-b3ce1e1f49e	null
721f0e49-4a34-49d1...	2025-02-24	МДР-23-02	Инструментальное обеспечение машиностроительных производств	0022	19:40	21:10	без подгруппы	Сотова Е.С.	семинар	d022d05a-fc21-41f1-80ed-b3ce1e1f49e	null
8c0a6a35-0c50-4c5e...	2025-02-25	МДР-23-02	Управление инженерными командами	0209	10:00	19:30	без подгруппы	Резина Н.А.	лекция	d022d05a-fc21-41f1-80ed-b3ce1e1f49e	null

Рис. 4.24. Фрагмент таблицы `daily_schedule`, содержащий расписание занятий

#### 4.2.5. Сбор метрик тестирования студентов и оценивания занятий

В главе 2 «Проектирование» автор описал схемы данных для процессов «Тестирование» и «Оценка качества проведённых занятий».

Результатом этих процессов является сохранение целевых сведений в базе данных. Сами по себе эти сведения имеют мало смысла, однако в СОКПЗ реализован асинхронный модуль, который перемещает данные в тиражируемую программную поисковую систему Elasticsearch, а точнее в её версию с открытым исходным кодом Opensearch. Opensearch представляет широкий спектр инструментов для полнотекстового поиска, и агрегации данных различных типов. Благодаря этому инструменту реализована возможность собирать данные в различные структурированные агрегаты и использовать их для любых целей. В случае СОКПЗ - необходимо анализировать динамику тестирования и оценок качества проведённых занятий, для последующего проведения аудита.

Данные в Opensearch перемещаются благодаря стеку технологий Debezium/Kafka — это распределённые высоконагруженные системы, подробное описание которых выходит за рамки данной работы [20]. Вкратце, данный стек позволяет указать источник данных - в нашем случае СУБД PostgreSQL, целевое назначение – Opensearch, и агрегатный SQL-запрос, в результате которого формируется JSON (JavaScript Object Node) - структура,

которая и сохраняется в Opensearch. Примером агрегатного запроса, может выступать SQL-запрос, приведённый на рисунке 4.25.

```
select ds.date, ds.teacher, ds.stgroup, ps.estimate, ds.subgroup, ds.semester_schedule_id, ds.type
from passed_assessment as ps
join assessment as a 1..n<->1: on ps.assessment_id = a.id
join daily_schedule as ds 1..n<->1: on a.daily_schedule_id = ds.id;
```

Рис. 4.25. Агрегатный запрос для сбора сведений о качестве проведённых занятий

#### 4.2.6. Результат агрегации и генерации метрик оценки качества проведённых занятий

Поскольку определение методологии оценки качества проведённых занятий, в частности перечень вопросов, которые предоставляются роли «Студент» для оценивания, делегируется роли «Преподаватель» или «Аудитор», т.е. конкретный преподаватель или кафедра определяют данный перечень, далее будут представлены данные агрегации целевых метрик, на абстрактном обезличенном перечне вопросов. На рисунке 4.26 представлен пример результата агрегатной функции, которая принимает на вход уникальный идентификатор конкретного занятия и возвращает распределение оценок по каждому вопросу из перечня вопросов оценки качества проведённых занятий, где столбец «question» содержит вопрос, «rate» - оценку по десятибалльной шкале, «count» - количество одинаковых оценок одной ценности, и «percentage» - процентное отношение оценок одной ценности относительно всего количество оценок.



	question	rate	count	percentage
1	Насколько содержимое лекции соответствует вашим ожиданиям?	1	19	9.41
2	Насколько содержимое лекции соответствует вашим ожиданиям?	2	24	11.88
3	Насколько содержимое лекции соответствует вашим ожиданиям?	3	23	11.39
4	Насколько содержимое лекции соответствует вашим ожиданиям?	4	14	6.93
5	Насколько содержимое лекции соответствует вашим ожиданиям?	5	23	11.39
6	Насколько содержимое лекции соответствует вашим ожиданиям?	6	14	6.93
7	Насколько содержимое лекции соответствует вашим ожиданиям?	7	18	8.91
8	Насколько содержимое лекции соответствует вашим ожиданиям?	8	22	10.89
9	Насколько содержимое лекции соответствует вашим ожиданиям?	9	22	10.89
10	Насколько содержимое лекции соответствует вашим ожиданиям?	10	23	11.39
11	Насколько содержимое лекции соответствует заявленной теме?	1	19	9.41
12	Насколько содержимое лекции соответствует заявленной теме?	2	16	7.92
13	Насколько содержимое лекции соответствует заявленной теме?	3	19	9.41
14	Насколько содержимое лекции соответствует заявленной теме?	4	24	11.88
15	Насколько содержимое лекции соответствует заявленной теме?	5	23	11.39
16	Насколько содержимое лекции соответствует заявленной теме?	6	12	5.94
17	Насколько содержимое лекции соответствует заявленной теме?	7	23	11.39
18	Насколько содержимое лекции соответствует заявленной теме?	8	22	10.89
19	Насколько содержимое лекции соответствует заявленной теме?	9	21	10.4
20	Насколько содержимое лекции соответствует заявленной теме?	10	23	11.39

Рис. 4.26 Распределение оценок качества проведённого занятия по целевому перечню вопросов

### 4.3. ВЫВОДЫ ПО ГЛАВЕ 4

Таким образом, в результате разработки системы, автором был разработан пользовательский интерфейс, главное серверное приложение серверное приложение, отвечающее за обработку основных бизнес-процессов, дополнительное серверное приложение, отвечающее за преобразование файлов расписания в целевые форматы, развернута СУБД PostgreSQL и инфраструктурные компоненты для миграции данных в Opensearch.

Результатом работы системы являются целочисленные агрегаты, которые в дальнейшем можно применить в анализе оценки качества проведённых занятий.

## **ГЛАВА 5. ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ СИСТЕМЫ ОЦЕНКИ КАЧЕСТВА ПРОВЕДЁННЫХ ЗАНЯТИЙ «СОКПЗ»**

### **5.1. ОЦЕНКА ЗАТРАТ ПРОЕКТА**

Для оценки затрат на разработку и внедрение системы оценки качества проведённых занятий необходимо определить следующие ключевые ресурсы:

1. Специалист по внедрению и поддержки ПО: системный администратор/DevOps, осуществляющий внедрение системы оценки качества проведённых занятий.
2. Full-stack разработчик: специалист, ведущий разработку и сопровождение целевой системы.
3. Инфраструктура и оборудование: серверное и клиентское оборудование (ПК системного администратора).

Для расчёта затрат нужно определить, какое количество вышеописанных ресурсов необходимо для создания минимально готового к внедрению решения, а также оценить их стоимость.

Для внедрения продукта необходимо минимум 3 специалиста – системный администратор с заработной платой 500 руб/ч на неполный рабочий день (4ч/день), full-stack разработчик с заработной платой 1200 руб/ч на неполную рабочую неделю (2 дня в неделю 4ч/день), dev-ops инженер, необходимый для доставки на целевые сервера, разворачивания и поддержки целевых дистрибутивов с заработной платой 1000 руб/ч (1 день в неделю 4ч/день) (см. Таблица 5.2).

Таблица 5.1.  
Расчёт прямых трудовых затрат

Наименование работ	Трудоемкость, часы	Необходимый специалист	Затраты, руб.
Разработка системы	360	Full-stack разработчик	432 000
Установка программы и развертывание ее на сервере	40	Dev-ops инженер	40 000
Сопровождение	150	Full-stack разработчик	180 000
Поддержка программы	150	Системный администратор	75 000

Итого общая сумма расходов на трудовые затраты составит 727 000 рублей. Общее количество трудовых затрат составит 700 чел/часов.

Таблица 5.2.  
Расчёт затрат на ПО

Наименование	Стоимость, руб.	Примечание
CentOS	0	Бесплатно, исходный открытый код
PostgreSQL	0	Бесплатно, исходный открытый код
Лицензия на антивирус	5 000	Стоимость лицензии на год

Для реализации отказоустойчивой архитектуры системы требуется развертывание основного сервера приложения и резервного сервера на случай аварийных ситуаций. Для рабочего места администратора необходим ПК средней производительности, способный эффективно работать с серверными ресурсами (см. Таблица 5.3). В стоимость проекта необходимо заложить пусконаладочные мероприятия, доставку комплектующих и монтажные услуги (см. Таблица 5.4).

Таблица 5.3.  
Расчёт затрат на инфраструктуру

Часть инфраструктуры	Наименование	Количество, ед.	Стоимость, руб.
Серверная часть	Сервер (Lenovo ThinkSystem SR250, Xeon Silver 4208 (8 ядер), 32 Гб DDR4, 2×1TB HDD)	1	110 250
Серверная часть	Резервный сервер (Intel NUC 11 Pro (NUC11TNKi5, i5-1135G7 (4 ядра), 16 Гб DDR4, 512GB SSD)	1	51 450
Серверная часть	Сетевое оборудование (Gigabit Ethernet, 2.5 Гбит/с)	1	1 500
Клиентская часть	ПК (Intel Core i7, RAM 16 Гб)	1	88 200

Для расчета амортизации оборудования необходимо установить срок полезного использования оборудования – предположительно 7 лет. В таком случае сумму годовой амортизации необходимо рассчитать по формуле  $\text{Балансовая стоимость, руб.} / \text{Срок полезного использования, лет}$  (табл. 5.5).

Таблица 5.4.  
Расчёт затрат на амортизацию

Наименование амортизируемого объекта	Стоимость, руб.	Сумма для амортизации, руб.
ПК	88 200	15 750
Основной сервер	110 250	15 000
Резервный сервер	51 450	7 350

Общая сумма амортизации составит 35 700 рублей.

Далее необходимо учесть затраты на содержание и использование оборудования – для расчёта затрат на ремонтные работы серверов и ПК стоит

заложить 10% от стоимости ПК и серверов (24 990 рублей) ежегодно, а также затраты на электроэнергию (примерно 6000 рублей) ежегодно.

Так же необходимо учесть не только стартовые затраты на разработку, но и ежегодные расходы на поддержку и администрирование системы (загрузка списка студентов, преподавателей и расписания). Получаем:  $10 \text{ ч/мес} \times 500 \text{ руб./ч} = 50\,000 \text{ рублей}$  ежегодно.

Исходя из проведённых вычислений, можно сформировать смету затрат на внедрение электронного мобильного журнала (табл. 5.6).

Таблица 5.5.  
Общая смета

Статья затрат	Сумма, руб.
Трудовзатраты (заработная плата системного администратора)	727 000
Программное обеспечение	105 000
Инфраструктура	251 475
Амортизация оборудования	35 700
Электроснабжение и ремонт	30 990
Поддержка и администрирование системы	50 000
ИТОГО	1 200 165

## 5.2. ОБОСНОВАНИЕ ЭКОНОМИЧЕСКОЙ ВЫГОДЫ ПРОЕКТА

Внедрение системы оценки качества проведённых занятий обеспечит следующие экономические преимущества:

Улучшение качества проведённых занятий: СОКПЗ обеспечит технологическую базу для анализа и агрегации численных метрик, обеспечивающих возможность проведения аудита.

Мультиплатформенный менеджмент расписания: СОКПЗ обеспечивает доступ к структурированному, соответствующему принципам UI/UX, отображению расписания, доступному на всех актуальных платформах (PC, Android, IOS) что улучшает пользовательский опыт.

При выполнении комплекса работ по внедрению системы в образовательный процесс университета возможно добиться следующих показателей:

Улучшение качества проведённых занятий: аналитически отчёты численных метрик по административным единицам (институты, кафедры, предметы, направления, преподаватели, группы и т.д.) учреждений высшего образования позволяют проводить аудит и точно выявлять недочёты в образовательном процессе.

Необходимо произвести расчёт для абстрактного вуза, который использует, например, программу «3KL Русский Moodle».

Цена «3KL Русский Moodle» составляет 333 000 рублей/год за ежегодную лицензию для тарифа Optima – минимальная поддержка + ежегодное продление 193 000 [8].

Для вуза с 3000+ студентами и 100+ преподавателями потребуется:

1. Доработка под требования вуза (кастомизация интерфейса).
2. Техническая поддержка.

При реализации проекта силами одного специалиста (например, администратора) общие затраты на внедрение Среды электронного обучения 3KL Русский Moodle может составлять значительно большую сумму, по сравнению с внедрением СОКПЗ.

Для работы Среды электронного обучения 3KL Русский Moodle необходимо будет закупить то же оборудование, что и для системы СОКПЗ, поскольку система так же является сетевой. Таким образом, затраты на инфраструктуру составят так же около 200 000 рублей.

Итак, для абстрактного вуза, использующего программу Среды электронного обучения 3KL Русский Moodle, расходы на внедрение и использование электронного журнала составляют около 726 000 рублей.

Общие затраты на проект «Электронный журнал кафедры ИТиВС» составляют 1 200 165 рублей. Ежегодно на амортизацию оборудования будет расходоваться 35 700 рублей, а на администратора системы 50 000 рублей.

Итого, разработка и внедрение СОКПЗ обойдётся дороже, чем внедрение 3KL русский Moodle, однако экономической эффективности можно добиться за счёт ежегодной амортизации.

Разница между стоимостью финансовых затрат систем составляет 474 000 рублей. Однако ежегодная амортизация СОКПЗ составляет 85 700 рублей, а 3KL Русский Moodle - 278 700. Таким образом окупаемость можно вычислить следующим образом.

$$\text{Время окупаемости} = \frac{|\text{Разница между стоимостью внедрения систем}|}{|\text{Разница между стоимостью амортизации систем}|} \approx$$

2,464 лет.

### 5.3. ВЫВОДЫ ПО ГЛАВЕ 5

Исходя из проведённого расчёта, можно сделать вывод о том, что без учёта позитивного эффекта улучшения качества проведённых занятий, время окупаемости проекта составит примерно 2 года.

Для образовательных учреждений, которые стремятся добросовестно улучшить свои позиции в рейтингах качества образования, разрабатываемая система является экономически целесообразным решением. Системы обеспечит единую точку сбора, агрегации и анализа метрик и позволит точно выявлять недостатки в образовательном процессе, и как следствие, повышать качество проведённых занятий.

При этом для небольших учебных подразделений (например, с численностью менее 20 студентов) внедрение системы может быть менее выгодным с экономической точки зрения, так как затраты на внедрение не окупятся за счет небольшого объема данных, а ручной учет в таких условиях может оставаться более целесообразным.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы по направлению 09.04.04 «Программная инженерия» на тему «Исследование и разработка системы оценки качества проведённых занятий», был проведён анализ существующих решений на рынке, разработана диаграмма

прецедентов и схема данных решения, выявлены и обоснованы требования к средствам реализации и агрегации данных.

Также были разработаны серверное приложение и пользовательский интерфейс целевой система.



**СПИСОК ЛИТЕРАТУРЫ**

1. Ректор СПб: Расписание [Электронный ресурс]. URL: <https://rector.spb.ru/raspisanie-vuz-4u.php> (Дата обращения: 22.05.2025).
2. ACID [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/ACID> (Дата обращения: 22.05.2025).
3. Elastic [Электронный ресурс]. URL: <https://www.elastic.co/docs/get-started/> (Дата обращения: 22.05.2025).
4. Kotlin Null Safety [Электронный ресурс]. URL: <https://kotlinlang.org/docs/null-safety.html> (Дата обращения: 22.05.2025).
5. Next.js [Электронный ресурс]. URL: <https://nextjs.org/docs> (Дата обращения: 22.05.2025).
6. OAuth Provider [Электронный ресурс]. URL: <https://github.com/stankin/mj/blob/master/oauthProvider.md> (Дата обращения: 22.05.2025).
7. RussianMoodle [Электронный ресурс]. URL: <https://opentechnology.ru/products/russianmoodle> (Дата обращения: 22.05.2025).
8. RussianMoodle: Покупка [Электронный ресурс]. URL: <https://opentechnology.ru/products/russianmoodle/buy> (Дата обращения: 22.05.2025).
9. Spring Data JPA Query Methods [Электронный ресурс]. URL: <https://docs.spring.io/spring-data/jpa/reference/jpa/query-methods.html> (Дата обращения: 22.05.2025).
10. Spring MVC @RequestMapping [Электронный ресурс]. URL: <https://docs.spring.io/spring-framework/reference/web/webmvc/mvc-controller/ann-requestmapping.html> (Дата обращения: 22.05.2025).
11. Stepik [Электронный ресурс]. URL: <https://welcome.stepik.org/ru/about> (Дата обращения: 22.05.2025).
12. TypeScript [Электронный ресурс]. URL: <https://www.typescriptlang.org/> (Дата обращения: 22.05.2025).

13. Webask [Электронный ресурс]. URL: <https://webask.io/help/chto-takoe-webask> (Дата обращения: 22.05.2025).
14. Skyprow: Трёхуровневая клиент-серверная архитектура [Электронный ресурс] URL: <https://sky.pro/wiki/sql/trehurovnevaya-klient-servernaya-arhitektura/> (Дата обращения: 22.05.2025).
15. OAUTH 2.0 [Электронный ресурс] URL: <https://oauth.net/2/> (Дата обращения 22.05.2025).
16. PostgreSQL: What is PostgreSQL [Электронный ресурс] URL: <https://www.postgresql.org/about/> (Дата обращения 22.05.2025).
17. The Java Virtual Machine [Электронный ресурс] URL: <https://docs.oracle.com/javase/specs/jvms/se8/html/jvms-1.html> (Дата обращения 22.05.2025).
18. Spring Security [Электронный ресурс] URL: <https://docs.spring.io/spring-security/reference/index.html> (Дата обращения 22.05.2025).
19. Spring Boot [Электронный ресурс] URL: <https://docs.spring.io/spring-boot/documentation.html> (Дата обращения 22.05.2025).
20. Debezium Architecture [Электронный ресурс] URL: <https://debezium.io/documentation/reference/stable/architecture.html> (Дата обращения 22.05.2025).