

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

ДЛЯ ПРОХОЖДЕНИЯ практической подготовки, проводимой в виде

|  |
| --- |
| **Преддипломной практики** |
| *(вид практической подготовки)* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ОБУЧАЮЩЕГОСЯ | **2** | КУРСА | магистратуры | ГРУППЫ | ИДМ-23-08 |
|  |  |  | *(уровень профессионального образования)* |  |  |

|  |
| --- |
| **романова ильи олеговича** |
| *(ФИО полностью)* |

|  |  |
| --- | --- |
| Кафедра: | Информационных технологий и вычислительных систем |

|  |  |
| --- | --- |
| Направление подготовки  (Специальность): | 09.04.04 Программная инженерия |

|  |  |
| --- | --- |
| Место прохождения  практической подготовки: | ФГБОУ ВО «МГТУ «СТАНКИН», кафедра  ИТиВС |

|  |  |
| --- | --- |
| сРОКИ прохождения  практической подготовки: | 01.10.2024 г. – 28.12.2024 г. |

Москва

2025

|  |
| --- |
|  |
| 1. Цель: разработать и описать программное решение серверного приложения и пользовательского интерфейса системы оценки качества проведённых занятий. 2. Задание: завершить разработку серверного приложения и пользовательского интерфейса, а также сформировать описание решения. 3. План-график выполнения работ:  |  |  |  | | --- | --- | --- | | № | Этапы | Сроки выполнения | | 1 | Разработать пользовательский интерфейс | 11.04.2025-20.04.2025 | | 2 | Разработать серверное приложение | 21.04.2025-25.04.2025 | | 3 | Описать интерфейс | 26.04.2025-30.04.2025 | | 4 | Описать серверное приложение | 01.05.2025-04.05.2025 | | 5 | Сформировать отчет | 05.05.2025-11.05.2025 | |

|  |  |  |  |
| --- | --- | --- | --- |
| Практикант | Романов Илья Олегович |  |  |
|  | *(ФИО полностью)* |  | *(подпись)* |

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель практической подготовки от Университета | Бердюгин Антон Валерьевич |  |  |
|  | *(ФИО полностью, должность)* |  | *(подпись)* |



**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

Отчет

о прохождении практической подготовки, проводимой в виде

|  |
| --- |
| **Преддипломной практики** |
| *(вид практической подготовки)* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ОБУЧАЮЩЕГОСЯ | **2** | КУРСА | магистратуры | ГРУППЫ | ИДМ-23-08 |
|  |  |  | *(уровень профессионального образования)* |  |  |

|  |
| --- |
| **романова ильи олеговича** |
| *(ФИО полностью)* |

|  |  |
| --- | --- |
| Кафедра: | Информационных технологий и вычислительных систем |

|  |  |
| --- | --- |
| Направление подготовки (Специальность): | 09.04.04 Программная инженерия |

|  |  |
| --- | --- |
| Место прохождения  практической подготовки: | ФГБОУ ВО «МГТУ «СТАНКИН», кафедра  ИТиВС |

|  |  |
| --- | --- |
| сРОКИ прохождения  практической подготовки: | 01.10.2024 г. – 28.12.2024 г. |

Москва

2025

Оглавление

[ОПИСАНИЕ ТРЕБОВАНИЙ К СРЕДСТВАМ РЕАЛИЗАЦИИ РАЗРАБАТЫВАЕМОЙ СИСТЕМЫ 3](#_Toc197696793)

[ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРЙФЕЙС 3](#_Toc197696794)

[ХРАНЕНИЕ ДАННЫХ 4](#_Toc197696795)

[СЕРВЕРНОЕ ПРИЛОЖЕНИЕ 5](#_Toc197696796)

[РЕАЛИЗАЦИЯ 6](#_Toc197696797)

[РЕАЛИЗАЦИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА 7](#_Toc197696798)

[РАСПИСАНИЕ ЗАНЯТИЙ 8](#_Toc197696799)

[ТЕСТИРОВАНИЕ И ОПРОСЫ 10](#_Toc197696800)

[МОДУЛЬНЫЙ ЖУРНАЛ. OAUTH 2.0 ПРОВАЙДЕР 15](#_Toc197696801)

[РЕАЛИЗАЦИЯ СЕРВЕРНОЙ ЧАСТИ 18](#_Toc197696802)

[СПИСОК ЛИТЕРАТУРЫ 19](#_Toc197696803)

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 и их ролях
  + О занятиях и событиях, связанных с ними
* 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.

ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРЙФЕЙС

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

* Поддержка роутинга «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-интерфейса используя следующий стек: TypeScript/Next.Js.

TypeScript – расширение языка JavaScript, вводящее ограничения в систему типов языка JavaScript. 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 фреймоврк, предназначенный для разработки веб-приложений, обладающих функционалом, выходящим за рамки SPA (Single Page Application), т.е. так называемых одностраничных приложений. 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.

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 «Тестирвоание».
* 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). Это означает, что данные в рамках использования PostgreSQL буду согласованы и надёжно сохранены.
* Для хранения данных для рендеринга пользовательского интерфейса будет использован так же PostgreSQL, т.к. его 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. Этот механизм реализован благодаря встроенным механизмам токенизациии и обратной индексации. Данный инструмент так же позволяет легко составлять запросы на агрегацию данных различных форматов, что в рамках задачи системы, позволит быстро проводить аналитику.

СЕРВЕРНОЕ ПРИЛОЖЕНИЕ

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

Исходя из описанных требований, в качестве инструментов реализации серверной части системы, автором принято решения использовать следующей стек технологий: Kotlin/Spring Boot/Spring Web/Spring Security/Spring Data JPA/MongoDB/Elasticsearch.

Kotlin – кроссплатформенный, статически типизированный, мульти-парадигменный, работающий поверх JVM (Java Vrtual Machine) язык программирования.

Kotlin позволяет реализовывать серверные приложения, а благодаря поддержке фреймоврка Spring, проекта Spring Boot и его экосистемы, разработка простого веб-приложения может выродиться в написание пары строк кода.

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

* Spring Security – позволяет настроить защиту веб-приложения от несанкционированного доступа.
* 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. Таким образом, загрузив проект из <https://start.spring.io/> , можно фактически сразу запустить веб-приложение.

РЕАЛИЗАЦИЯ

После определения с выбором основных технических средств, автор приступил к разработке целевой системы. На рисунке 3.1 представлена схема архитектуры решения СОКПЗ.

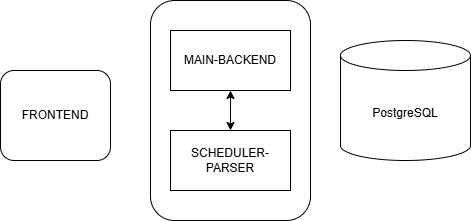


Рис. 3.1 Общая архитектурная схема решения

На данной схеме представлены 4 целевых элемента архитектуры решения:

1. Frontend - пользовательских интерфейс, реализованный с помощью фреймоврка Next.js, языка JavaScript и его диалекта TypeScript.
2. Main-backend - основное серверное приложение. Отвечает за весь функциональный спектр - авторизация, расписание, тестирование и т.д.
3. Scheduler-Parser - дополнительное, но обязательно серверное приложение. Его ответственность - преобразовывать файлы формата pdf в целевую json-структуру для дальнейшего сохранения и отображения расписания.
4. PostgreSQL - целевая СУБД. В ней хранятся фактичски все данные всех процессов системы.

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

РЕАЛИЗАЦИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Для того чтобы взаимодействие с СОКПЗ стало удобным и интуитивным, автором был разработан пользовательский интерфейс.

Пользовательский интерфейс состоит из ряда различных web-страниц, которые отвечают за определённый функционал для конкретных ролей.

На рисунке 3.2 представлено древо страниц пользовательского интерфейса.

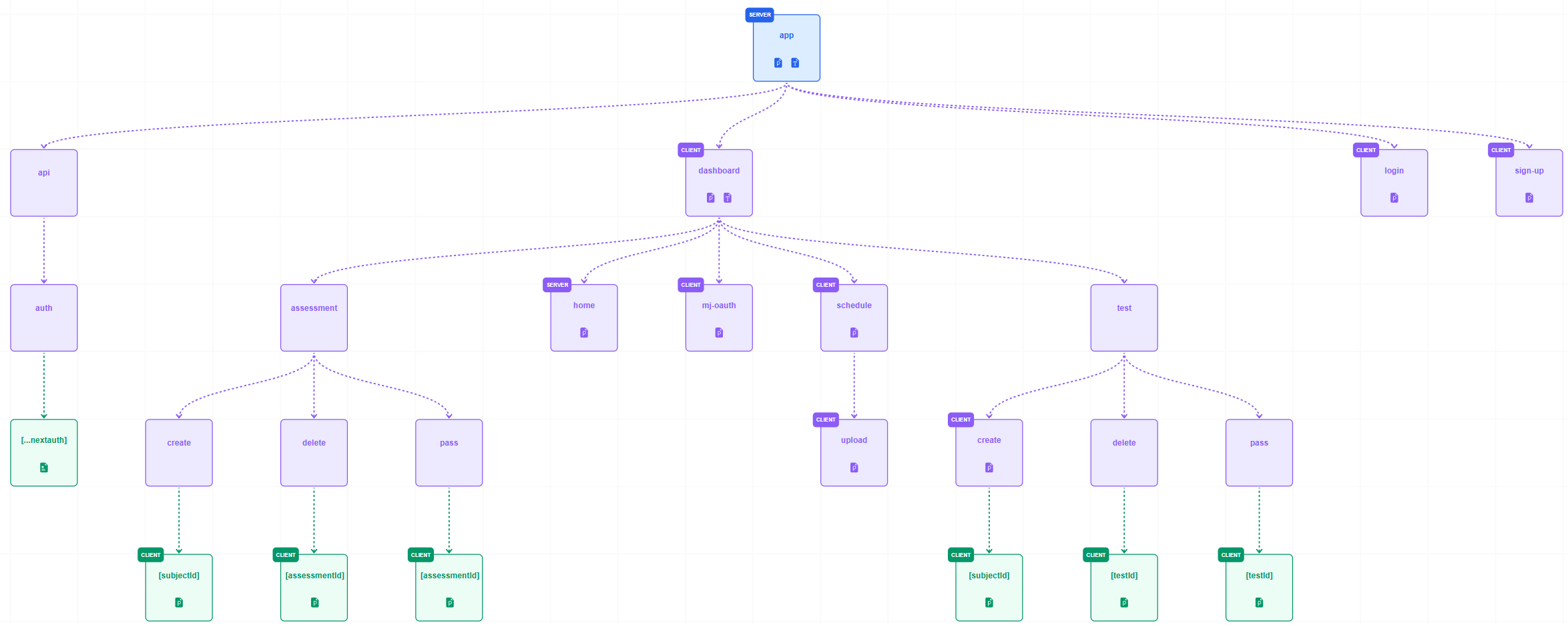


Рис. 3.2 Древо страниц пользовательского интерфейса СОКПЗ

Так же стоит отметить, что в фреймоврке 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 3.3

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 3.3 Организационная структура проекта пользовательского интерфейса СОКПЗ

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

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 (Рисунок 3.4).

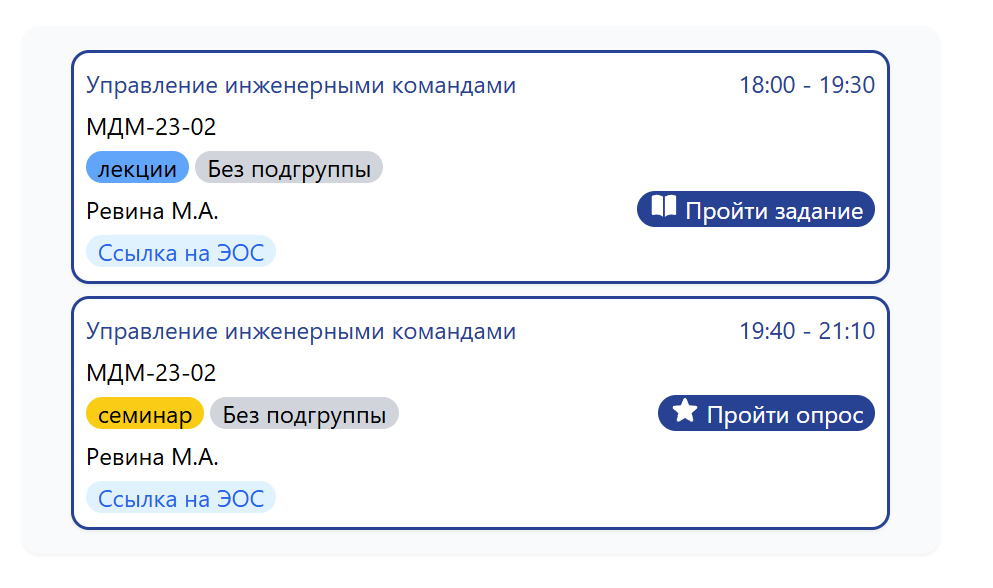


Рис. 3.4 Расписание в пользовательском интерфейсе СОКПЗ

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 3.5.

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.

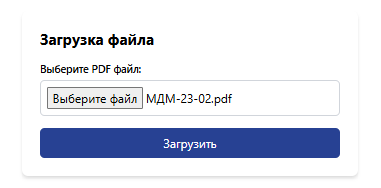


Рис. 3.5 Форма для загрузки расписания в пользовательском интерфейса СОКПЗ

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. Подробней данный процесс будет рассмотрен в главе, посвящённой средствам реализации.

ТЕСТИРОВАНИЕ И ОПРОСЫ

На рисунке 3.5 представлен компонент расписания, доступный для роли «Преподаватель» и «Аудитор» для создания элементов заданий и опросов.

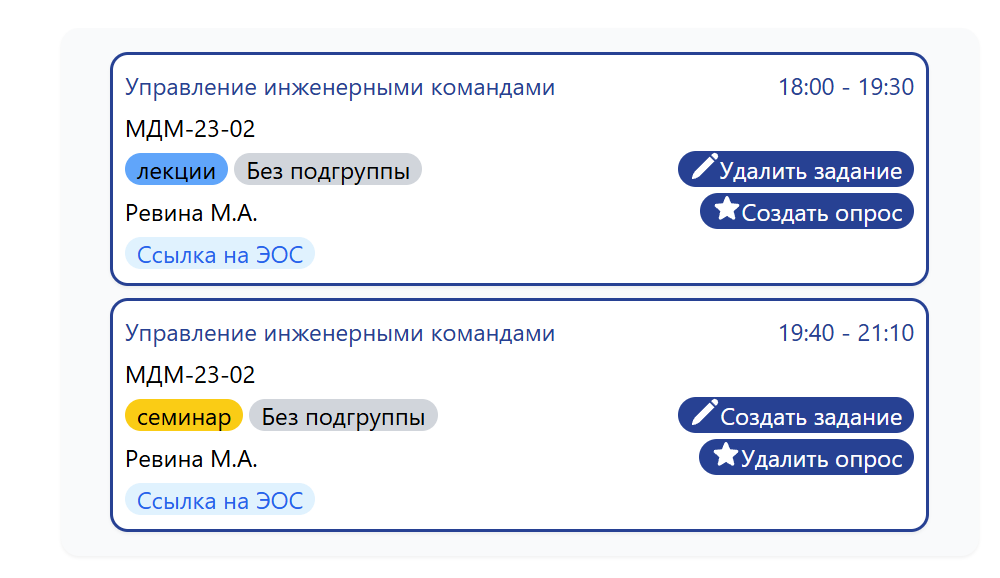


Рис. 3.6 Управление тестами и опросами

На рисунке 3.7 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.

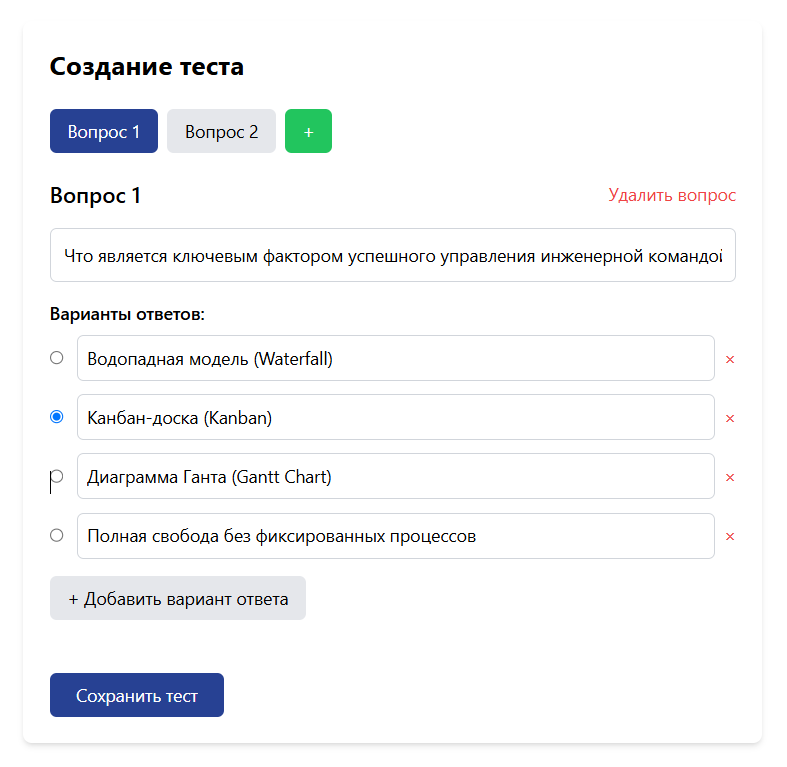


Рис. 3.7 Форма создания элемента тестирования

На рисунке 3.8 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.

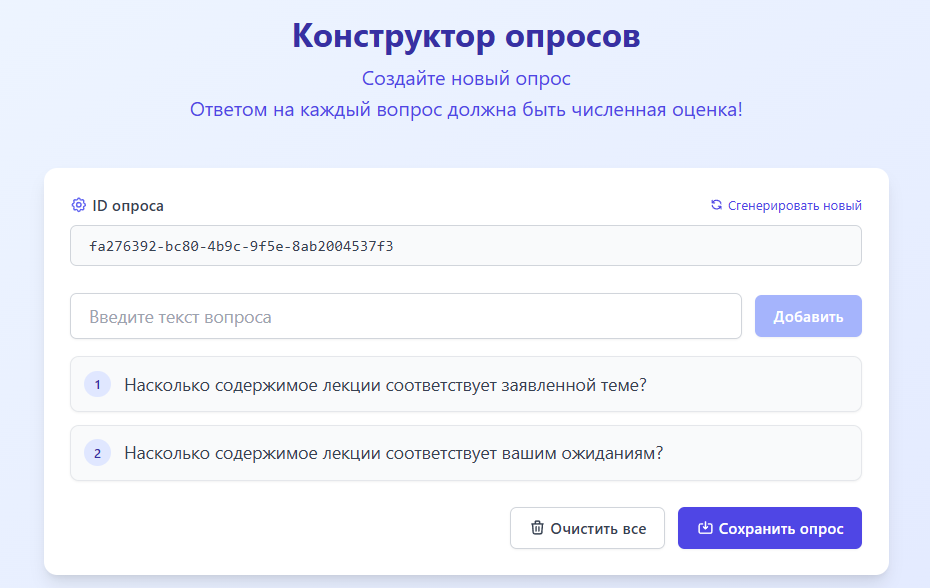


Рис. 3.8 Форма создания элемента опроса оценки качества проведённого занятия

На рисунке 3.9 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.

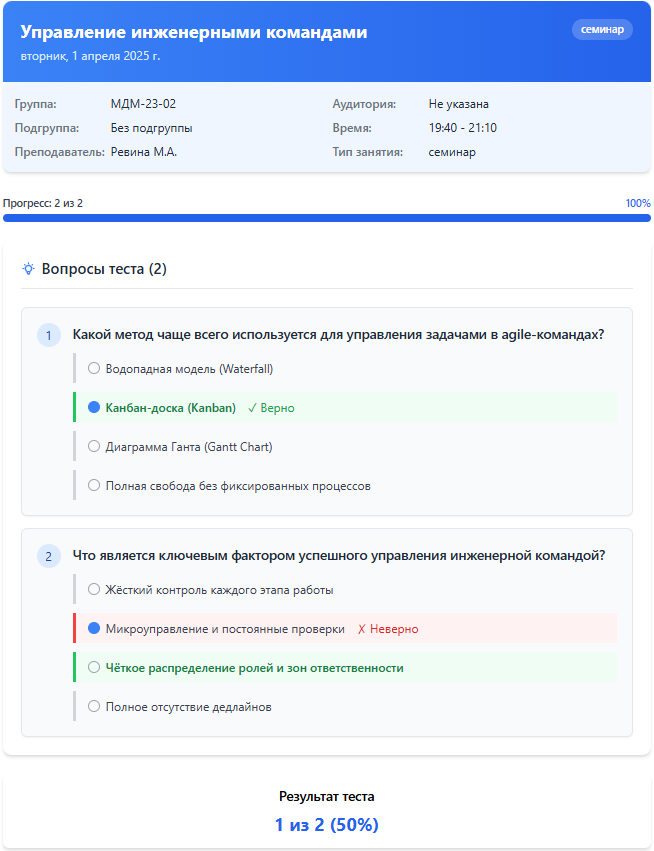


Рис. 3.9 Форма прохождения элемента тестирования ролью «Студент»

На рисунке 3.10 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.

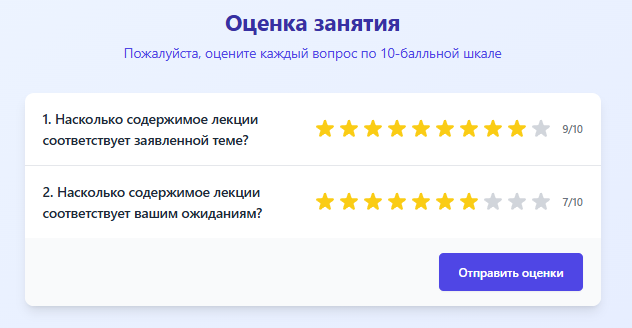


Рис. 3.10 Форма прохождения элемента опроса об оценке качества проведённого занятия

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 3.11 и 3.12.

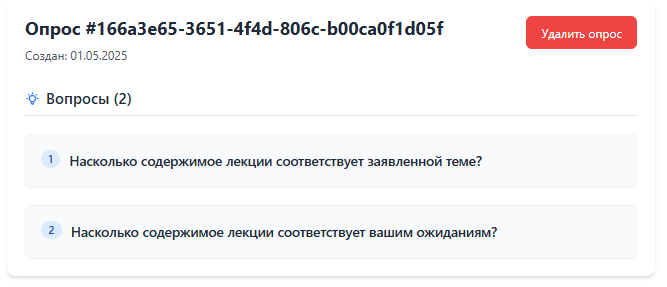


Рис. 3.11 Форма удаления элемента опроса из базы данных

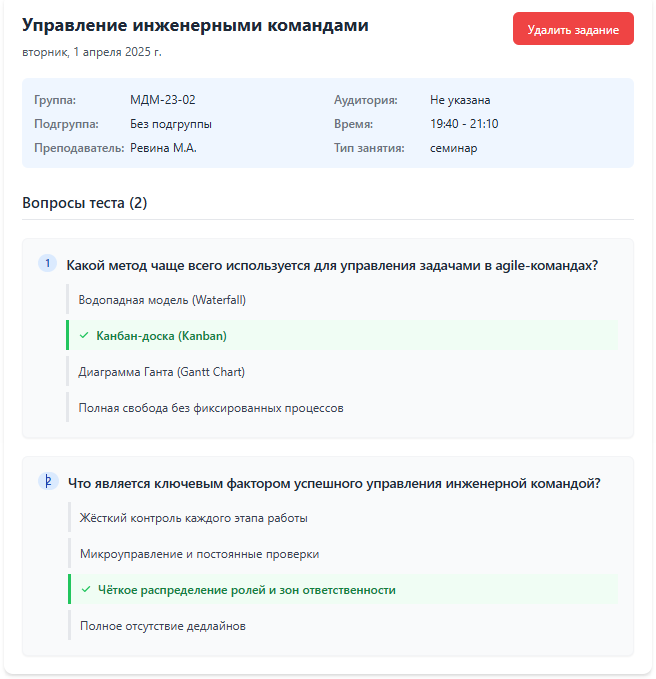


Рис. 3.12 Форма удаления элемента тестирования из базы данных

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:

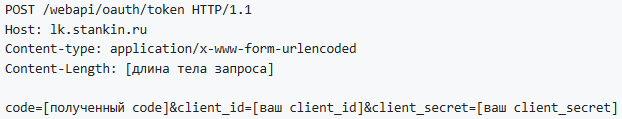
1. На сайте необходимо разместить ссылку «https://lk.stankin.ru/webapi/oauth/authorize?response\_type=code&client\_id=[client\_id]&redirect\_uri=[redirect\_uri]», где:
   1. [client\_id] - идентификатор интегрируемой системы, в случае СОКПЗ - rate-please-stankin
   2. [redirect\_uri] - адрес сервера 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.
4. Пользовательский интерфейс отправляет запроса на «main» серверное приложение, которое в свою очередь отправляет запрос в систему «Модульный журнал». Пример запроса приведён на рисунке 3.13
5. В ответ пользовательский интерфейс получает данные студента. Пример данных приведён на рисунке 3.14  
   

Рис 3.13 Форма удаления элемента тестирования из базы данных



Рис. 3.14 Пример данных студента, полученнёх из сервиса «Модульный журнал»

В результате прохождения процедуры аутентификации, в пользовательском интерфейсе отобразятся данные студента (рисунок 3.15). Так же в данной форме доступна кнопку привязывания данных студента с сущностью пользователя системы СОКПЗ.

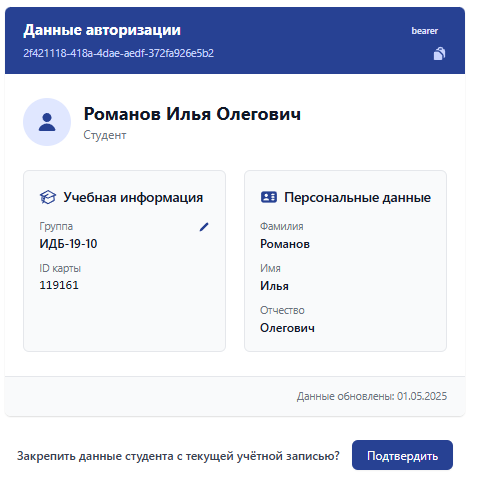


Рис. 3.15 Пример данных студента, полученных из сервиса «Модульный журнал»

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.

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 так же генерацией целевых отчетов.
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 данных в основное серверное приложение, где затем происходит финальное преобразование и сохранение в базу данных.

СПИСОК ЛИТЕРАТУРЫ

1. Spring Framework Documentation [Электронный ресурс]: офиц. сайт. // Режим доступа: <https://docs.spring.io/spring-framework/reference/index.html>

|  |  |  |  |
| --- | --- | --- | --- |
| Практикант | Романов Илья Олегович |  |  |
|  | *(ФИО полностью)* |  | *(подпись)* |

Отчет о прохождении практической подготовки сдан «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.

|  |  |
| --- | --- |
| Оценка |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель практической подготовки от Университета | Бердюгин А.В., ст. преподаватель |  |  |
|  | *(ФИО полностью, должность)* |  | *(подпись)* |