Министерство науки и высшего образования

Российской Федерации

ФГБОУ ВО

«Уральский государственный горный университет»

Кафедра информатики

Утверждаю

Зав. кафедрой информатики

\_\_\_\_\_\_\_\_\_\_\_\_ А. В. Дружинин

“ ” ” 201г.



*ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ОБРАЗОВАТЕЛЬНОЙ ОНЛАЙН-ПЛАТФОРМЫ*

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**Пояснительная записка**

Руководитель

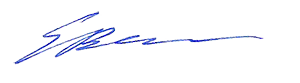
ВКР, доцент \_\_\_\_\_\_\_\_\_\_\_\_ *Е.А. Волкова*

(подпись, дата) (И. О. Ф.)

Разработал студент \_\_\_\_\_\_\_\_\_\_\_\_  *Е.А. Уфимцев*

(подпись, дата) (И. О. Ф.)

Группа   *АУБП-21-1*





Екатеринбург, 2025

ЛИСТ ЗАДАНИЯ

Министерство образования и науки

Российской Федерации

ФГБОУ ВО

«Уральский государственный горный университет»

Факультет *инженерно-экономический*

Кафедра *информатики*

Специальность *Информатика и вычислительная техника (направление подготовки «Автоматизация управления бизнес-процессами»)*

Утверждаю

Зав. кафедрой информ@тики

\_\_\_\_\_\_\_\_\_\_\_\_ А. В. Дружинин

“ ” ” 201\_ г.



**ЗАДАНИЕ**

**ПО ДИПЛОМНОМУ ПРОЕКТИРОВАНИЮ**

**Студенту** *Уфимцеву Егору Александровичу*

(фамилия, имя, отчество полностью)

**1 Тема ВКР** *Проектирование и разработка образовательной онлайн-платформы*

Утверждена приказом по университету № 941/3е от «18»\_\_\_\_апреля\_\_\_\_\_ 2025 г.

**2 Срок сдачи студентом ВКР:**

**Сдача пояснительной записки, видео и презентации для прохождения процедуры допуска до:**

«10\_» июня 202 г.

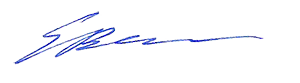
**Сдача итоговой версии пояснительной записки (с отсканированными подписями и рецензиями), видео и презентации для процедуры дипломирования до:**

«16 \_»   июня 2025г.

**Ф. И. О. руководителя дипломного проекта (работы)\_\_** *Волкова Е.А.*

ученая степень -\_, ученое звание \_-\_.

Дата выдачи задания «18 » *« апреля »* 2025 г.



**Руководитель ВКР** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.А. Волкова

**Задание на проектирование ВКР получил** \_\_\_\_\_\_\_\_\_\_\_\_\_ Е.А. Уфимцев

Содержание

[АННОТАЦИЯ……………………………………………………………………………](#_Toc104922466)5

[ВВЕДЕНИЕ………………………………………………………………………………](#_Toc104922468)8

[1 ХАРАКТЕРИСТИКА ПРОБЛЕМНОЙ СИТУАЦИИ И ПОСТАНОВКА ЗАДАЧ.10](#_Toc104922469)0

[1.1 Характеристика проблемной области 10](#_Toc104922470)0

[1.2 Аналитический обзор решений 10](#_Toc104922471)1

[1.3 Постановка задач, решаемых в проекте 13](#_Toc104922472)3

[ВЫВОДЫ ПО 1 ГЛАВЕ 14](#_Toc104922473)4

[2 СИСТЕМНЫЙ АНАЛИЗ ОБЪЕКТА УПРАВЛЕНИЯ/ИНФОРМАТИЗАЦИИ…..](#_Toc104922474)15

[2.1 Концептуальная модель управления объекта](#_Toc104922475) 16

[2.2 Функциональное моделирование](#_Toc104922476) 30

[2.3 Формирование требований к системе](#_Toc104922477) 60

[ВЫВОДЫ ПО 2 ГЛАВЕ](#_Toc104922478) 63

[3 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ……………………](#_Toc104922479)64

[3.1 Проектирование архитектуры информационной системы](#_Toc104922480) 65

[3.2 Инфологическое проектирование](#_Toc104922481) 73

[3.3 Проектирование интерфейса](#_Toc104922482) 80

[3.4 Программная реализация проекта](#_Toc104922483) 89

[ВЫВОДЫ ПО 3 ГЛАВЕ 169](#_Toc104922484)

[ЗАКЛЮЧЕНИЕ………………………………………………………………171](#_Toc104922485)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ……………………………...](#_Toc104922486)174

**АННОТАЦИЯ**

АННОТАЦИЯ

**Название ВКР:** «Проектирование и разработка образовательной онлайн-платформы».

**ФИО автора:** Уфимцев Егор Александрович.

**Научный руководитель:** Доцент Волкова Евгения Алексеевна.

**Ключевые слова:** электронный журнал, WEB-платформа, сдача заданий, голосовая связь, информационная система, навигация, HTML, CSS, JS, MySQL, Java, Spring boot, Thymeleaf, Nginx, Janus Gateway.

Выпускная квалификационная работа посвящена исследованию, проектированию и разработке онлайн-образовательной платформы.

Выпускная квалификационная работа состоит из аннотации, введения, трёх глав и заключения. Работа содержит 174 страницы, 91 рисунок, 0 таблиц, 24 листинга программного кода, 16 литературных источников.

Объекты разработки – система аудио/видео связи, система заданий, система журнала.

В работе выполнено исследование, предметом которого является аудио/видео онлайн-связь.

Спроектировано, разработано и представлено программное обеспечение, представляющее собой прототип информационной системы.

ABSTRACT

**Graduation work title**: *"Design and Development of an Educational Online Platform"*  
**Author’s name**: Egor Aleksandrovich Ufimtsev  
**Supervisor**: Evgeniya Alekseyevna Volkova  
**Keywords**: electronic journal, web platform, assignment submission, voice communication, information system, navigation, HTML, CSS, JS, MySQL, Java, Spring Boot, Thymeleaf, Nginx, Janus Gateway

This final qualification work is dedicated to the research, design, and development of an online educational platform.

The thesis consists of an abstract, an introduction, three chapters, and a conclusion. The work includes 174 pages, 91 figures, 0 tables, 24 code listings, and 16 references.

Development objects: audio/video communication system, assignment system, journal system.

The study conducted in this work focuses on the subject of audio/video online communication.

The software developed and presented in this thesis is a fully functional information system.

**ВВЕДЕНИЕ**

ВВЕДЕНИЕ

С каждым годом образовательные онлайн-платформы стабильно набирают популярность. Они используются в школах, ВУЗах, для домашнего обучения и дистанционных занятий. Любой человек, обладающий смартфоном или любым другим современным устройством с доступом к интернету, сможет без проблем получить доступ к образовательному процессу онлайн [2, 3].

Образовательные онлайн-платформы упрощают жизнь. Пользователи могут обмениваться заданиями, вести электронные журналы, использовать текстовые чаты для решения важных вопросов, создавать аудио/видео конференции, если текстовых чатов недостаточно, и всё это дистанционно.

В настоящее время всё больше и больше аспектов повседневной жизни получает цифровизацию и переходит в онлайн формат. Не стало исключением и образование. Карантин в 2020-2021 годах заставил учебные заведение искать альтернативу очным занятиям. Многие из них временно перешли на дистанционный формат обучения и проводили занятия на онлайн-образовательных платформах. После окончания карантина, потребность в онлайн-платформах только увеличилась и тенденции к получению образования онлайн сильно возросли [4 ,5].

Учебные заведения нередко прибегают к использованию образовательных онлайн-платформ. Они могут быть интегрированы в образовательный процесс целиком или же отдельные их функции (например, электронный журнал).

Онлайн-образование – популярная, особенно среди молодёжи, тенденция – это и есть актуальность проекта. Использование простого, понятного и удобного способа обмена информацией существенно упростит жизнь как учащимся, так и преподавателям.

На основе данной информации, можно сформулировать следующую цель: создание системы, которая совмещала бы видеоконференции, подсистему заданий, чаты и электронный журнал, а также автоматизировала ведение электронного журнала и обмен заданиями.

Одно из главных составляющих системы – WEB-приложение, совмещающее подсистему заданий, подсистему чатов, подсистему видеоконференций и электронный журнал.

Данная система обладает большим потенциалом, раскрыть который помогут дальнейшие нововведения. Позднее, разработанную образовательную онлайн-платформу могут перенять и другие учебные заведения, ведь она удобна и проста в использовании.

Использование образовательной онлайн-платформы намного упростит процесс обучения: пользователь легко и без проблем сможет сдать/разместить задание, ознакомиться с электронным журналом, принять участие в видеоконференции, написать сообщение в чате.

1. **ХАРАКТЕРИСТИКА ПРОБЛЕМНОЙ СИТУАЦИИ И ПОСТАНОВКА ЗАДАЧ**

1 ХАРАКТЕРИСТИКА ПРОБЛЕМНОЙ СИТУАЦИИ И ПОСТАНОВКА ЗАДАЧ

* 1. **Характеристика проблемной области**

Цифровая трансформация образования, ускоренная пандемией COVID-19, стала необратимым процессом. Сегодня онлайн-форматы обучения перешли из категории «временного решения» в устойчивый компонент образовательной среды. Согласно исследованию Grand View Research, глобальный рынок онлайн-образования демонстрирует стабильный рост и к 2030 году, по прогнозам, вырастет на 31,5% [15]. Такой масштабный рост подтверждает актуальность создания эффективных, удобных и комплексных онлайн-решений.

Тем не менее, быстрое развитие онлайн-платформ для обучения выявило ряд существенных проблем, которые затрудняют их использование и снижают эффективность как инструмента образования.

Наиболее явные проблемы:

1. Студенты и преподаватели вынуждены использовать сразу несколько несвязанных между собой инструментов. Это приводит к потере информации, снижению мотивации к обучению и росту когнитивной нагрузки. Преподаватели, в свою очередь, тратят значительное количество времени на дублирование информации и переключение между системами.
2. Многие из наиболее распространённых решений не обладают полным набором необходимых функций. Например, видеосвязь в Zoom или Teams не сопровождается встроенными средствами оценки и обратной связи. В то же время классические LMS-платформы (например, Moodle) требуют сложной настройки и предоставляют избыточный функционал, неиспользуемый пользователями [16].

Всё это говорит о высокой востребованности единой, интегрированной онлайн-платформы, которая:

* объединяет видеоконференции, систему обмена заданиями, чат и электронный журнал в одном интерфейсе;
* обеспечивает удобный доступ с любых устройств;
* облегчает работу преподавателя и делает взаимодействие с учащимися более структурированным.

Создание такого решения позволит устранить фрагментацию цифровой инфраструктуры, повысить эффективность взаимодействия между участниками образовательного процесса и сформировать устойчивую цифровую образовательную среду.

* 1. **Аналитический обзор решений**

При разработке системы был проведён анализ других, уже существующих, образовательных онлайн-платформ.

Одним из решений для обзора был взят Teams от Microsoft – один из крупнейших в мире разработчиков в сфере проприетарного программного обеспечения для различного рода вычислительной техники. Microsoft Teams — это корпоративная платформа, которая объединяет в себе функции чата, видеоконференций, хранения файлов и интеграции с различными приложениями. Предоставляет инструменты для дистанционного обучения, совместной работы преподавателей и студентов, а также управления образовательным процессом.

Это мощный инструмент для дистанционного и гибридного обучения, который объединяет в себе видеоконференции, управление заданиями и взаимодействие студентов и преподавателей. Несмотря на некоторые сложности в освоении, он предлагает обширный функционал для образовательных учреждений всех уровней.

Благодаря проведённому анализу были выделены преимущества этого продукта:

– Наличие личных групповых чатов;

– Возможность записывать и сохранять видеоконференции;

– Возможность совместной работы с документами;

– Поддержка большого количества участников видеоконференции;

– Возможность составления тестов в заданиях;

– Интеграция сервисов от Microsoft;

– Фильтрация нежелательного контента.

Разумеется, у данного продукта есть и свои недостатки:

– Запрещён на территории РФ;

– Нестабильная работа видеоконференций;

– Плохая работа синхронизации.

Следующим для сравнения был выбран Zoom. Это популярная платформа для видеоконференций, которая широко используется в бизнесе, образовании и личном общении.

Его преимуществами являются:

– Возможность назначить соорганизаторов конференции

– Возможность установить водяной знак для конфиденциальности

– Возможность вести несколько чатов в одной конференции

– Гибкая настройка видеоконференций

– Высокое качество аудио и видео

– Поддержка большого количества (до 1000 человек) в одной видеоконференции

Недостатками служат:

– Слабая система безопасности

– Функции бесплатной версии сильно ограничены

– Отсутствие важного функционала (например, системы заданий) для обеспечения удобного управления образовательным процессом

Для аналитического обзора решений также был рассмотрен Google classroom. Это облачная платформа для управления учебным процессом, которая предназначена для учителей, студентов и образовательных учреждений. Она позволяет организовывать онлайн-обучение, выдавать и проверять задания, а также вести коммуникацию между преподавателями и учащимися.

Несомненными преимуществами данной платформы являются:

– Интеграция с сервисами google

– Возможность совместной работы преподавателей над заданиями

– Возможность комментировать задания/записи в электронном журнале

– Возможность для преподавателей настроить права доступа к материалам

Недостатками продукта являются:

– Зависимость от google-аккаунта

– Нет возможности управлять участниками конференции

– Отсутствует электронный журнал

– Отсутствует возможность создавать тесты в заданиях

Был также рассмотрен Slack. Это корпоративный мессенджер, разработанный для удобного общения и совместной работы в командах. Он позволяет обмениваться сообщениями, создавать каналы по интересам или проектам, делиться файлами, проводить видеозвонки.

Преимущества платформы:

– Высокая степень защиты данных благодаря надёжным алгоритмам шифрования

– Присутствие систем аналитики активности пользователей

– Возможность автоматизации (например, создание чат-ботов)

– Возможность создавать приватные каналы внутри общего канала

– Быстрые алгоритмы поиска по сообщениям и файлам.

Недостатки платформы:

– Проблемы со стабильностью

– Отсутствие важного функционала (например, системы заданий) для обеспечения удобного управления образовательным процессом

– Большая стоимость платных тарифов

Исходя из результатов анализа, проведённого с различными системами, можно сделать вывод, что на рынке не так много популярных платформ, имеющих всё необходимое для удобного управления образовательным процессом онлайн, а те, что есть, имеют немало своих недостатков. Также можно выделить наиболее распространённые недостатки: плохая стабильность, отсутствие одного или более необходимых модулей. Плохо реализованные подсистемы обратной связи приложения с пользователем также может затруднить работу.

* 1. **Постановка задач, решаемых в проекте**

В результате анализа уже существующих на рынке решений, были выделены пункты, которые должны быть учтены и внедрены в разрабатываемый продукт:

– гибкость в управлении пользователями;

– упрощённый интерфейс;

– расширенный инструментарий для управления заданиями;

– поддержка большинства популярных браузеров;

– приемлемое качество звука и видео;

– необходимы набор инструментов для управления образовательным процессом (электронный журнал, система заданий, видеоконференции).

Исходя из данных пунктов, мной были выделены задачи, требующие реализации в рамках данной выпускной квалификационной работы:

1. Разработка концепта системы;
2. Выбор архитектуры разрабатываемой системы;
3. Проектирование и разработка WEB-платформы;
4. Проектирование базы данных;
5. Проектирование и реализация системы видео и аудиосвязи;
6. Проектирование и разработка системы заданий;
7. Проектирование и разработка электронного журнала;
8. Проектирование и реализация интерфейса системы.

**ВЫВОДЫ ПО 1 ГЛАВЕ**

Перед этапом разработки был проведён анализ уже существующих образовательных онлайн-платформ, были определены их соответствующие плюсы и минусы и выделены пункты, которые стоит учесть в разрабатываемом продукте.

Проведённый аналитический обзор показал, что существует ряд недостатков, основные из которых ­­– плохая стабильность, отсутствие необходимых инструментов, зависимость от сторонних платформ.

Исходя ряда недостатков, было решено сделать систему менее трудозатратной, простой и представляющей из себя минимальный набор необходимых функций для дистанционного управления образовательным процессом.

Система должна быть надёжной и стабильной. Также предоставляемые ей инструменты должны быть полностью настраиваемы под конкретные задачи.

Система должна предоставлять необходимый набор инструментов, таких, как электронный журнал, видеоконференции, система заданий. Интерфейс программы не должен вызывать трудностей у пользователей, не имеющих навыков работы с ПК

1. **СИСТЕМНЫЙ АНАЛИЗ ОБЪЕКТА УПРАВЛЕНИЯ/ИНФОРМАТИЗАЦИИ**

2 СИСТЕМНЫЙ АНАЛИЗ ОБЪЕКТА УПРАВЛЕНИЯ/ИНФОРМАТИЗАЦИИ

* 1. **Концептуальная модель управления объекта**

Объект информатизации – образовательный процесс в системе дистанционного обучения, включающий проведение видеоконференций, управление электронным журналом и управление заданиями.

Разрабатываемый продукт включает в себя следующие компоненты:

1. регистрация – выполняет функцию занесения данных о новом пользователе в базу данных.
2. авторизация – предоставление человеку прав на использование функций приложения, доступное только после прохождении аутентификации – процедуры проверки подлинности, которая сравнивает введённые пользователем логин и пароль с теми данными, которые хранятся в БД. Если сравнение удачно, то пользователь проходит аутентификацию и получает доступ к различным функциям системы, если же нет – сообщение об ошибке или предложение зарегистрироваться/восстановить пароль;
3. Подсистема заданий – раздел системы, содержащий в себе основные функции управления и обмена заданиями между пользователями. Пользователи с ролью «преподаватель» и выше получат доступ к полному управлению своими заданиями, а пользователи с ролью «студент» смогут просматривать их и сдавать выполненные задания;
4. Подсистема электронного журнала – раздел системы, содержащий в себе функции ведения журнала успеваемости. Пользователи с ролью «преподаватель» и выше могут заполнять успеваемость и просматривать успеваемость всех учеников, пользователи с ролью «студент» могут просматривать свою успеваемость.
5. Подсистема видеоконференций – раздел системы, содержащий в себе основные функции управления и проведения видеоконференций. Пользователи с ролью «преподаватель» и выше получат доступ к полному управлению своих видеоконференций. Также они получат расширенные возможности управления участниками во время проведения видеоконференциями. Пользователи с ролью «студент» смогут просматривать доступные видеоконференции, а также участвовать в них, с возможностью контроля своей периферии (микрофон, веб-камера). Все участники смогут пользоваться текстовым чатом конференции.
6. Подсистема текстовых чатов – раздел системы, позволяющий пользователям создавать чата и общаться между собой. Чат можно создать как без темы, так и посвящённый конкретному предмету или группе.
7. Меню администратора доступно только администраторам системы. Оно содержит в себе набор функций для управления пользователями, группами и контентом пользователей;

Сущности системы и их атрибуты видно на следующих рисунках 1 - 11:

Сущность «Активные видеоконференции» изображена на рисунке 1:



Рисунок 1 – Сущность «Активные видеоконференции»

Атрибуты:

* Идентификатор
* Id комнаты
* Количество участников
* Session id
* Handle id

Сущность «Группа» изображена на рисунке 2:

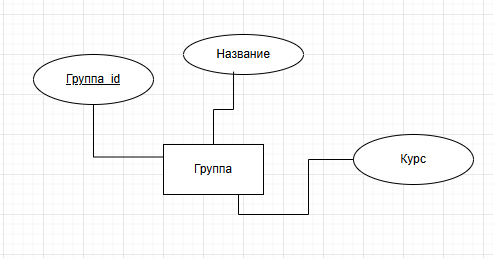


Рисунок 2 – Сущность «Группа»

Атрибуты:

* Идентификатор
* Название
* Курс

Сущность «Журнал» изображена на рисунке 3:

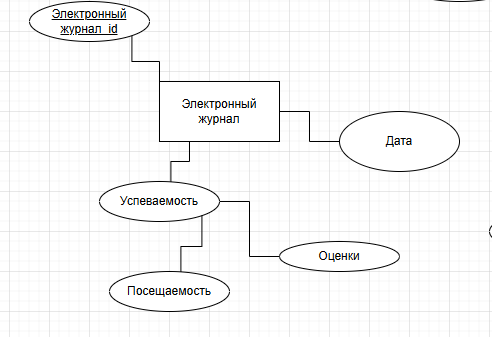


Рисунок 3 – Сущность «Журнал»

Атрибуты:

* Идентификатор
* Успеваемость (Оценки, Посещаемость)

Сущность «Задания» изображена на рисунке 4:



Рисунок 4 – Сущность «Задания»

Атрибуты:

* Идентификатор
* Текст
* Название
* Дата конца
* Дата начала

Сущность «Видеоконференции» изображена на рисунке 5:

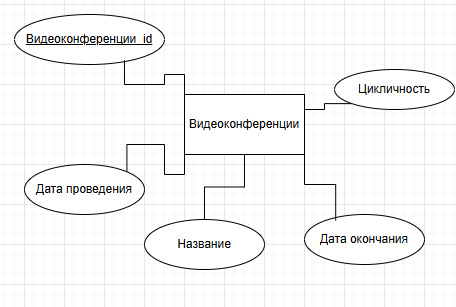


Рисунок 5 – Сущность «Видеоконференции»

Атрибуты:

* Идентификатор
* Цикличность
* Дата проведения
* Название
* Дата окончания

Сущность «Баны» изображена на рисунке 6:

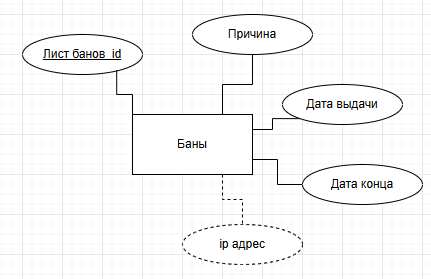


Рисунок 6 – Сущность «Баны»

Атрибуты:

* Идентификатор
* Дата выдачи
* Дата конца
* Причина
* Ip адрес

Сущность «Роль» изображена на рисунке 7:

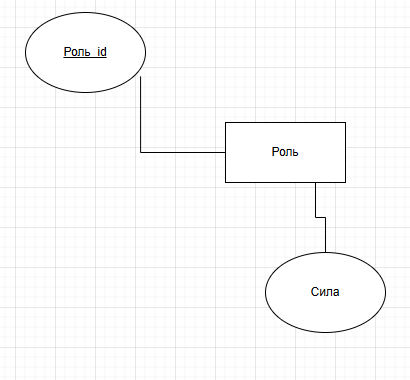


Рисунок 7 – Сущность «Роль»

Атрибуты:

* Идентификатор
* Сила

Сущность «Пользователь» изображена на рисунке 8:

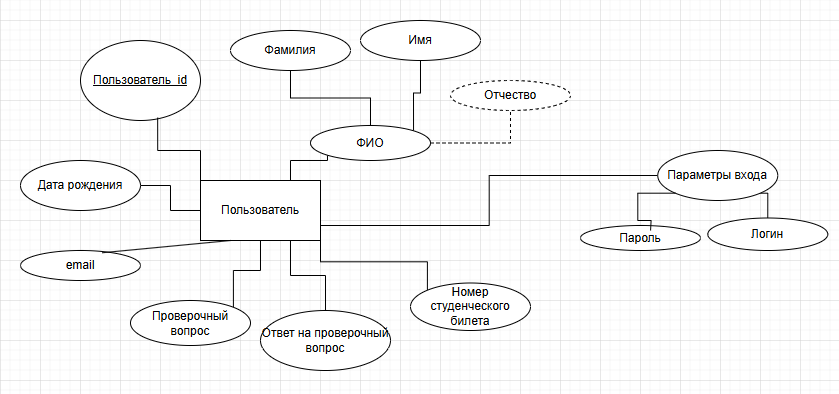


Рисунок 8 – Сущность «Пользователь»

Атрибуты:

* Идентификатор
* ФИО (Фамилия, Имя, Отчество (необязательный))
* Параметры входа (Пароль, Логин)
* Дата рождения
* Номер студенческого билета
* Ответ на проверочный вопрос
* Проверочный вопрос
* email

Сущность «Письма восстановления пароля» изображена на рисунке 9:

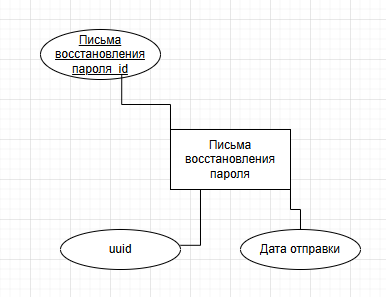


Рисунок 9 – Сущность «Администратор»

Атрибуты:

* Идентификатор
* UUID
* Дата отправки

Сущность «Журнал аудита» изображена на рисунке 10:

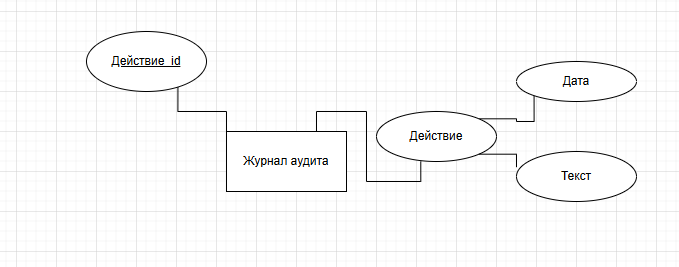


Рисунок 10 – Сущность «Журнал аудита»

Атрибуты:

* Идентификатор
* Действие (Дата, Текст)

Сущность «Чат» изображена на рисунке 11:

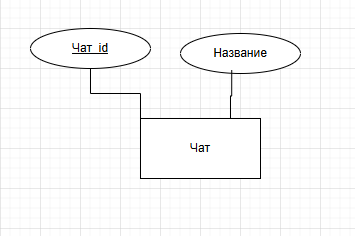


Рисунок 11 – Сущность «Чат»

Атрибуты:

* Идентификатор
* Название

Сущность «Сообщение чата» изображена на рисунке 12:

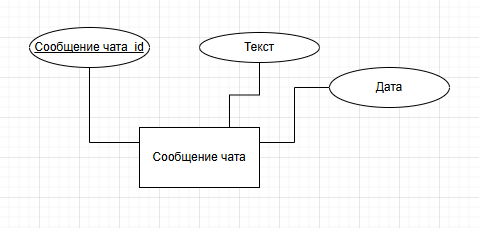


Рисунок 12 – Сущность «Сообщение чата»

Атрибуты:

* Идентификатор
* Текст
* Дата

Сущность «Файлы чата» изображена на рисунке 13:

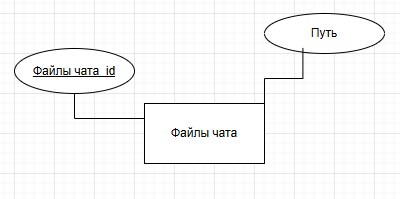


Рисунок 13 – Сущность «Файлы чата»

Атрибуты:

* Идентификатор
* Путь

Сущность «Видеоконференции» изображена на рисунке 14:



Рисунок 14 – Сущность «Видеоконференции»

Атрибуты:

* Идентификатор
* Цикличность
* Название
* Дата проведения
* Дата окончания

Сущность «Файлы выполненных заданий» изображена на рисунке 15:

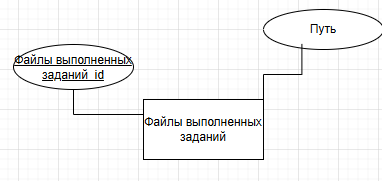


Рисунок 15 – Сущность «Файлы выполненных заданий»

Атрибуты:

* Идентификатор
* Путь

Сущность «Файлы пользователей» изображена на рисунке 16:

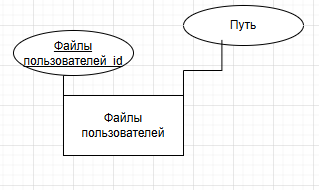


Рисунок 16 – Сущность «Файлы пользователей»

Атрибуты:

* Идентификатор
* Путь

Сущность «Файлы заданий» изображена на рисунке 17:

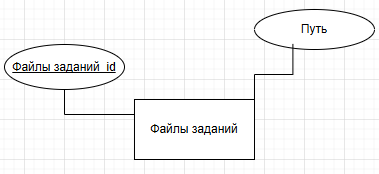


Рисунок 17 – Сущность «Файлы заданий»

Атрибуты:

* Идентификатор
* Путь

Сущность «Выполненные задания» изображена на рисунке 18:



Рисунок 18 – Сущность «Выполненные задания»

Атрибуты:

* Идентификатор
* Дата сдачи
* Дата проверки
* Оценка
* Комментарий студента
* Комментарий преподавателя

Сущность «Ограничения участника конференции» изображена на рисунке 19:

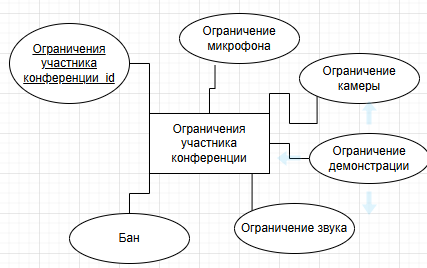


Рисунок 19 – Сущность «Ограничения участника конференции»

Атрибуты:

* Идентификатор
* Бан
* Ограничение звука
* Ограничение демонстрации
* Ограничение камеры
* Ограничение микрофона

Сущность «Чат видеоконференции» изображена на рисунке 20:

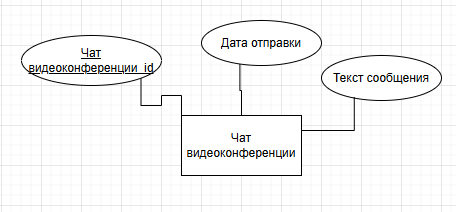


Рисунок 20 – Сущность «Чат видеоконференции»

Атрибуты:

* Идентификатор
* Текст сообщения
* Дата отправки

Сущность «Участники видеоконференции» изображена на рисунке 21:



Рисунок 21 – Сущность «Участники видеоконференции»

Атрибуты:

* Идентификатор
* Состояние демонстрации
* Подключён
* Состояние звука
* Состояние камеры
* Состояние микрофона

Сущность «Предмет» изображена на рисунке 22:

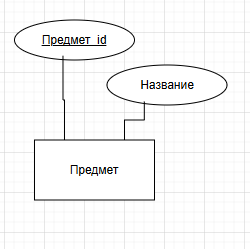


Рисунок 22 – Сущность «Предмет»

Атрибуты:

* Идентификатор
* Название

Связи между сущностями информационной системы показаны на рисунке 23:

«имеет роль» - связь между пользователем и ролью

"Состоит в" — связь между Студентом и Группой.

"Выполняет" — связь между Студентом и Заданием.

"Оцениваются в" — связь между Заданием и Электронным журналом.

"Участвует в" — связь между Пользователем и Видеоконференцией.

«Приглашается в» - связь между Группой и Видеоконференциями.

«Загружает» - связь между Пользователем и Файлами пользователя.

«Получает» - связь между Пользователем и Письмами для восстановления пароля.

«Участвует» - связь между Пользователем и Чатом.

«Принадлежат» -связь между Сообщениями в чате и Чатом.

«Имеют» - связь между Сообщениями в чате и Файлами сообщений.

«Создаёт» - связь между Преподавателем и Заданиями.

«Проверяет» - связь между Преподавателем и Выполненными заданиями.

«Загружает» - связь между Пользователем и Выполненными заданиями.

«Имеют» - связь между Выполненными заданиями и Файлами выполненных заданий.

«Относятся к» - связь между Выполненными заданиями и Заданиями.

«Имеют» - связь между Заданиями и Файлами заданий.

«Относятся к» - связь между Заданиями и Предметом.

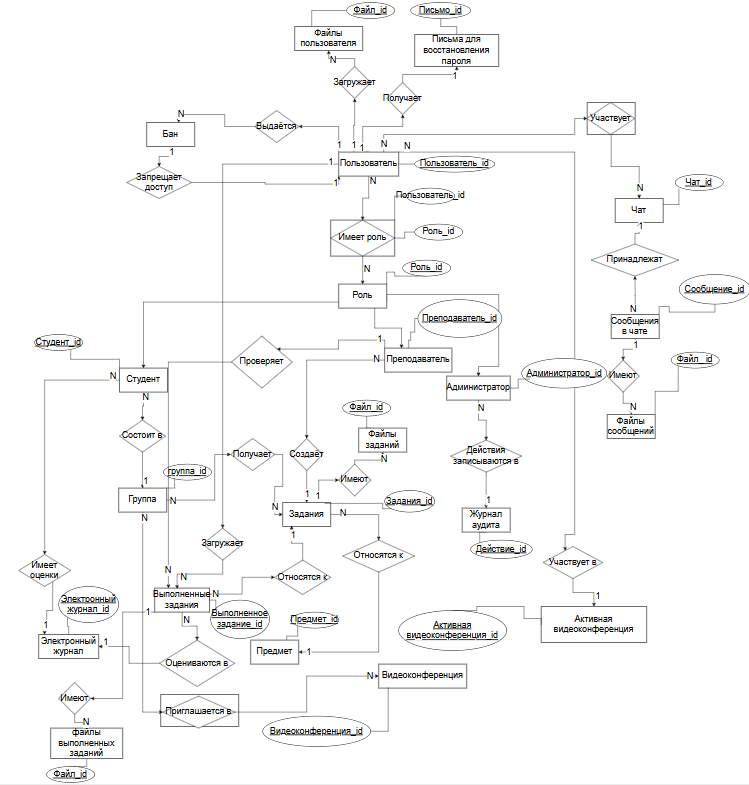


Рисунок 23 – Связи сущностей в системе

* 1. **Функциональное моделирование**

1. Диаграмма as-is

* Управление заданиями

Текущий процесс сопровождается рядом сложностей: из-за отсутствия единой системы преподавателям приходится вручную уведомлять студентов о новых заданиях и сроках их выполнения, а студентам может быть сложно систематизировать все полученные материалы.

Для того чтобы подробнее разобраться в текущем процессе управления заданиями и выявить его недостатки, будет использована методология функционального моделирования IDEF0. Эта диаграмма позволит наглядно отобразить этапы создания, выполнения, сдачи и проверки заданий в существующей системе.

Контекстная диаграмма IDEF0 as-is «Управление заданиями» изображена на рисунке 24:

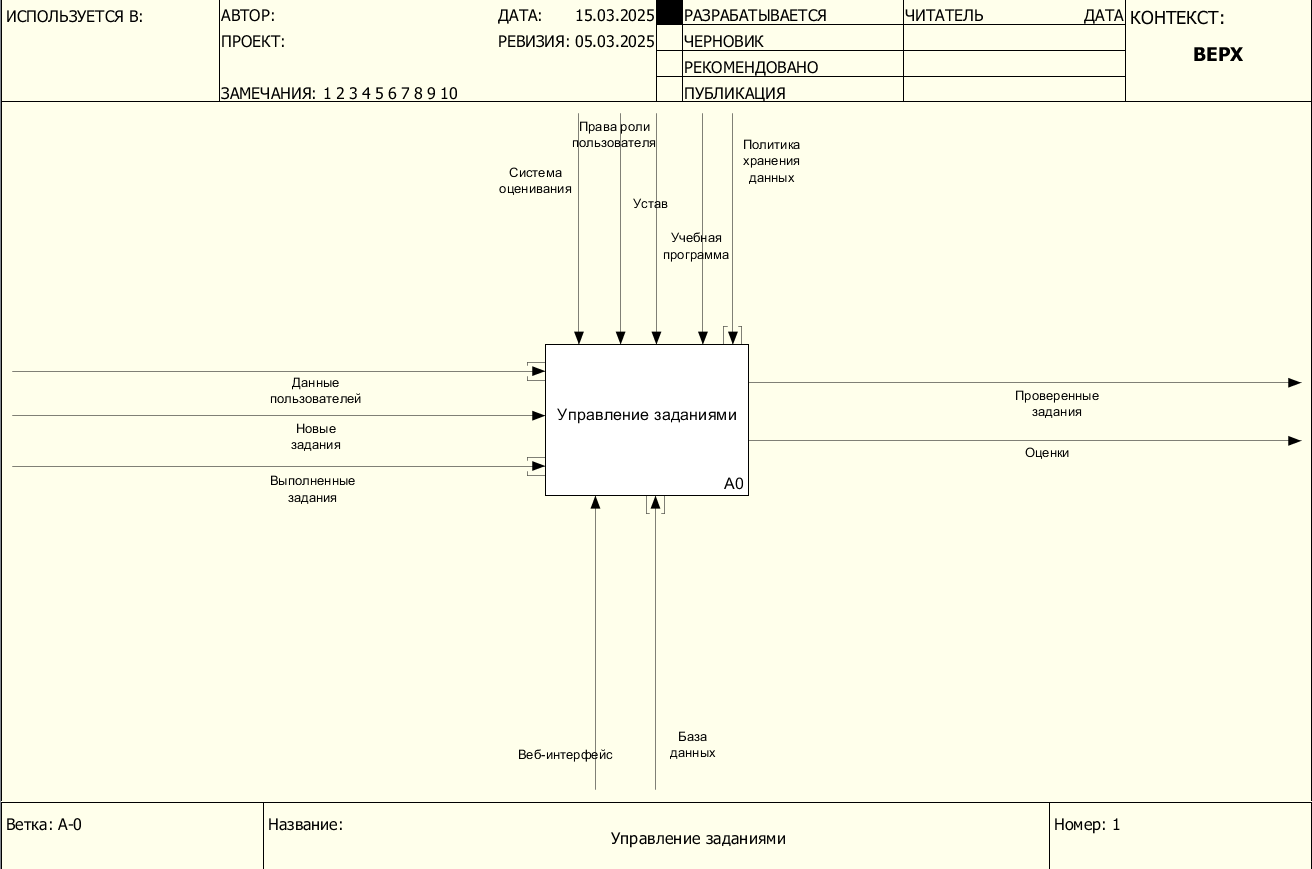


Рисунок 24 – Диаграмма as-is «Управление заданиями» в нотации IDEF0

Декомпозицию контекстной диаграммы IDEF0 as-is «Управление заданиями», имеющей идентификатор A0, можно увидеть на рисунке 25:

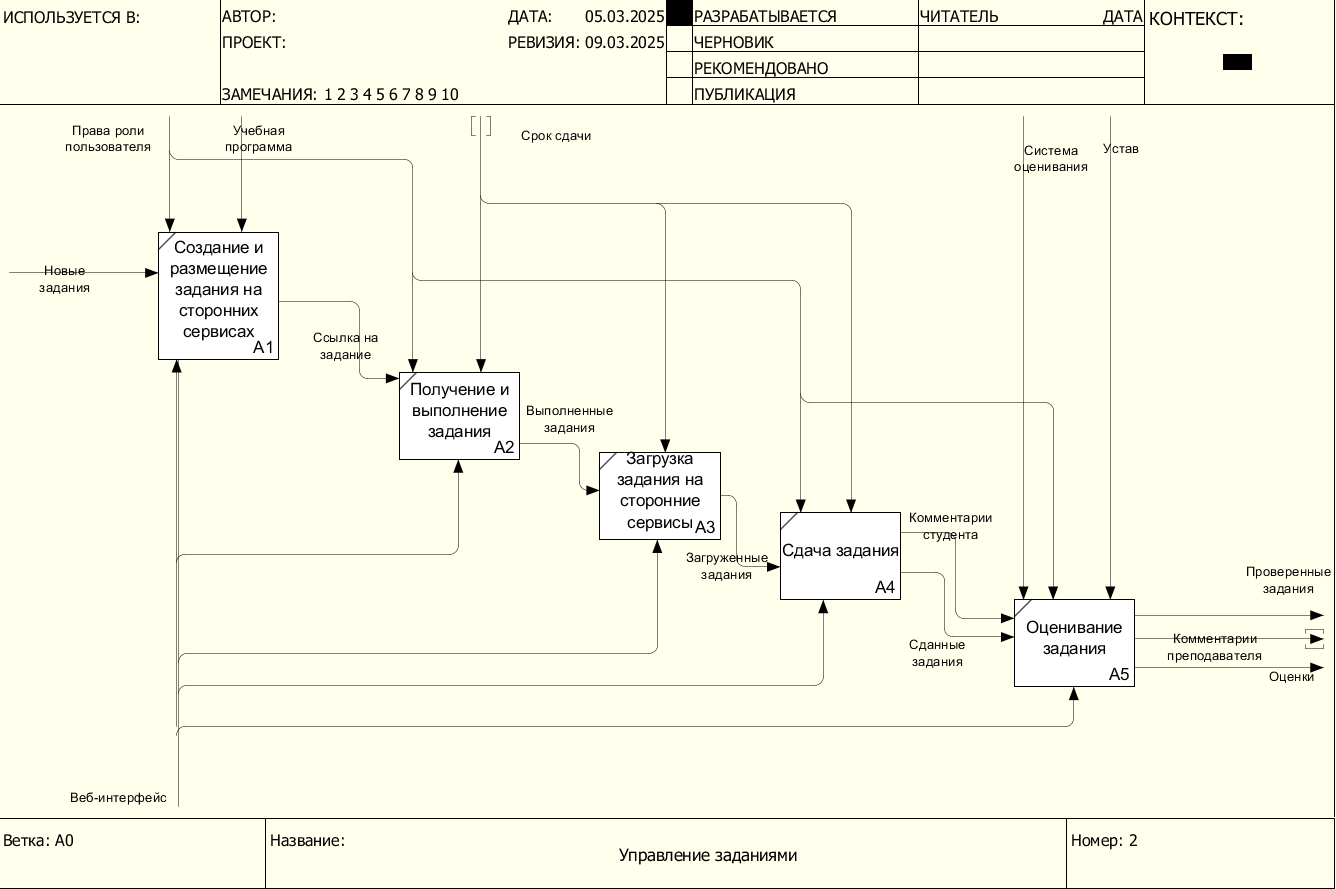


Рисунок 25 – Декомпозиция диаграммы «Управление заданиями»

Диаграмма показывает, как именно происходит обмен заданиями между преподавателем и студентами.

Преподаватели, размещая задания для студентов, вынуждены использовать различные сторонние сервисы, такие как Google Drive, электронная почта или мессенджеры. Студенты, в свою очередь, получают доступ к этим заданиям через рассылки или вручную ищут их в предоставленных источниках. После выполнения работы студенты также загружают свои решения на сторонние платформы или отправляют их преподавателю напрямую по почте.

* Управление видеоконференциями

Текущий процесс сопровождается рядом сложностей: из-за отсутствия единой системы преподавателям приходится использовать несколько сторонних сервисов для ведения электронного журнала, а студенты могут не получать уведомления своевременно. Кроме того, подобный вариант хранения электронного журнала не обеспечивает достаточный уровень надёжности и безопасности данных.

Для того чтобы подробнее разобраться в текущем процессе ведения электронного журнала и выявить его недостатки, будет использована методология функционального моделирования IDEF0. Эта диаграмма позволит наглядно представить этапы выставления оценок, учёта посещаемости, хранения и передачи данных об успеваемости в существующей системе.

Контекстная диаграмма IDEF0 as-is «Управление электронным журналом» изображена на рисунке 26:



Рисунок 26 – Диаграмма as-is «Управление электронным журналом» в нотации IDEF0

Декомпозицию контекстной диаграммы IDEF0 as-is «Управление электронным журналом», имеющей идентификатор A0, можно увидеть на рисунке 27:

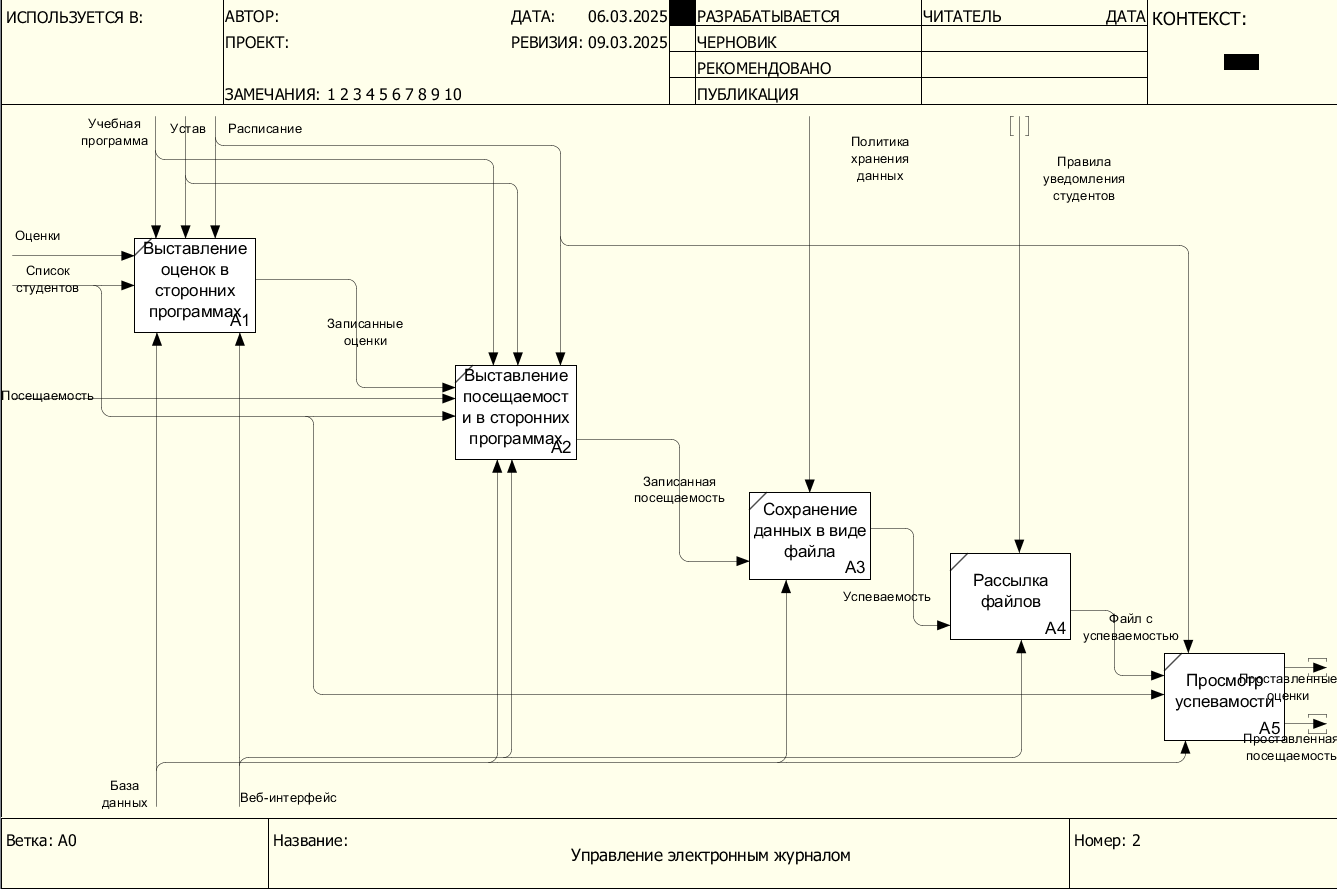


Рисунок 27 – Декомпозиция диаграммы «Управление электронным журналом»

Диаграмма показывает, как именно происходит уведомление студентов об их успеваемости.

Преподаватели, ведущие учёт успеваемости студентов, вынуждены использовать различные сторонние программы, такие как Excel, Google Sheets или текстовые редакторы, для заполнения электронного журнала. Оценки за выполненные задания и данные о посещаемости занятий вносятся вручную, что увеличивает вероятность ошибок и затрудняет оперативное обновление данных.

После заполнения журнала преподаватели сохраняют файлы с данными об успеваемости на своих устройствах или в облачных хранилищах. Для передачи информации студентам и администрации преподаватели отправляют файлы по электронной почте или через мессенджеры, что создаёт риск потери данных и усложняет доступ к информации. Студенты, в свою очередь, сталкиваются с трудностями при поиске своих оценок, так как информация может быть разбросана по нескольким источникам.

* Управление видеоконференциями

Для того чтобы подробнее разобраться в текущем процессе проведения видеоконференций и выявить его недостатки, будет использована методология функционального моделирования IDEF0. Эта диаграмма позволит наглядно отобразить этапы создания, подключения участников, проведения занятия и сохранения данных в существующей системе.

Контекстная диаграмма IDEF0 as-is «Управление видеоконференциями» изображена на рисунке 28:

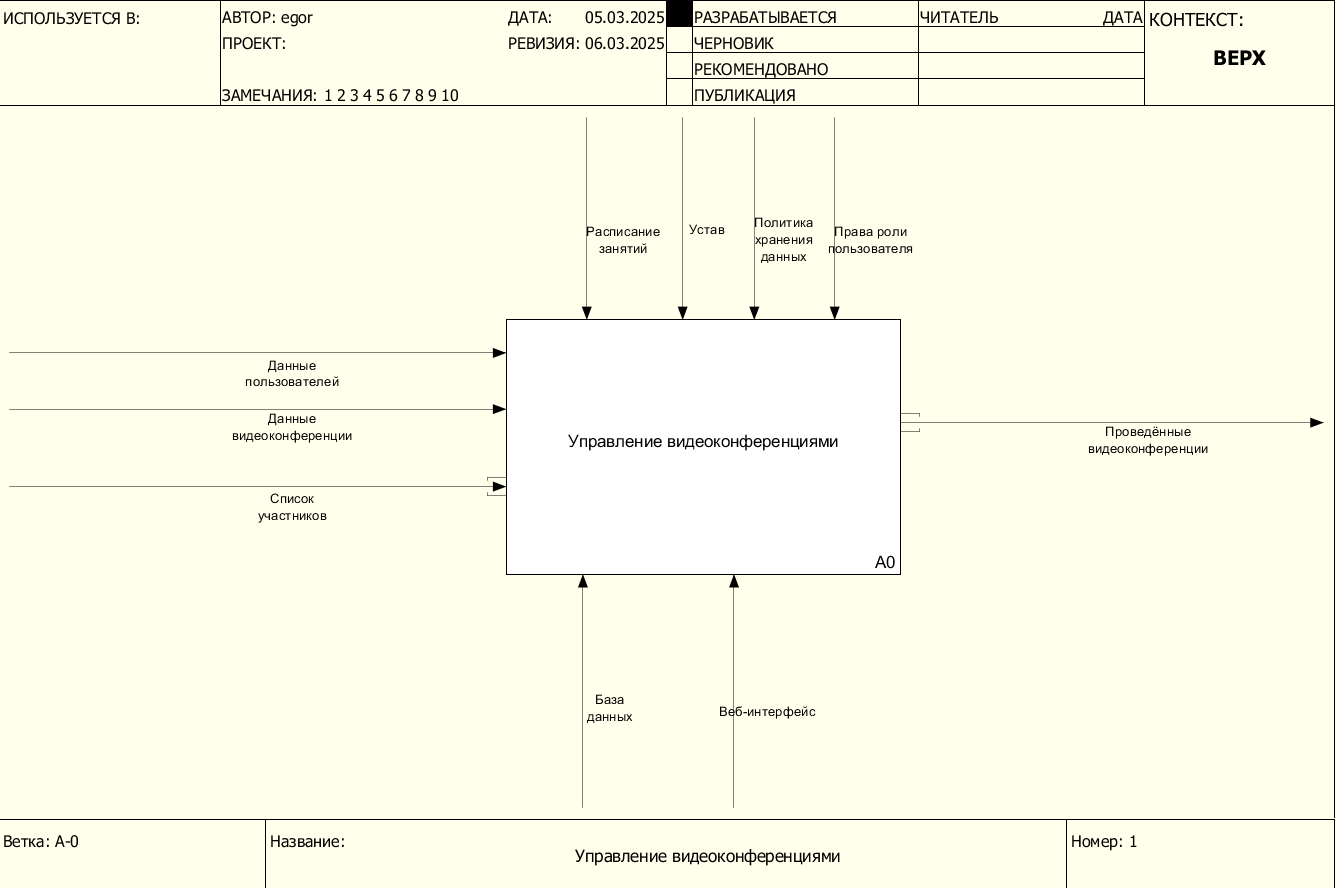


Рисунок 28 – Диаграмма as-is «Управление видеоконференциями» в нотации IDEF0

Декомпозицию контекстной диаграммы IDEF0 as-is «Управление видеоконференциями», имеющей идентификатор A0, можно увидеть на рисунке 29:

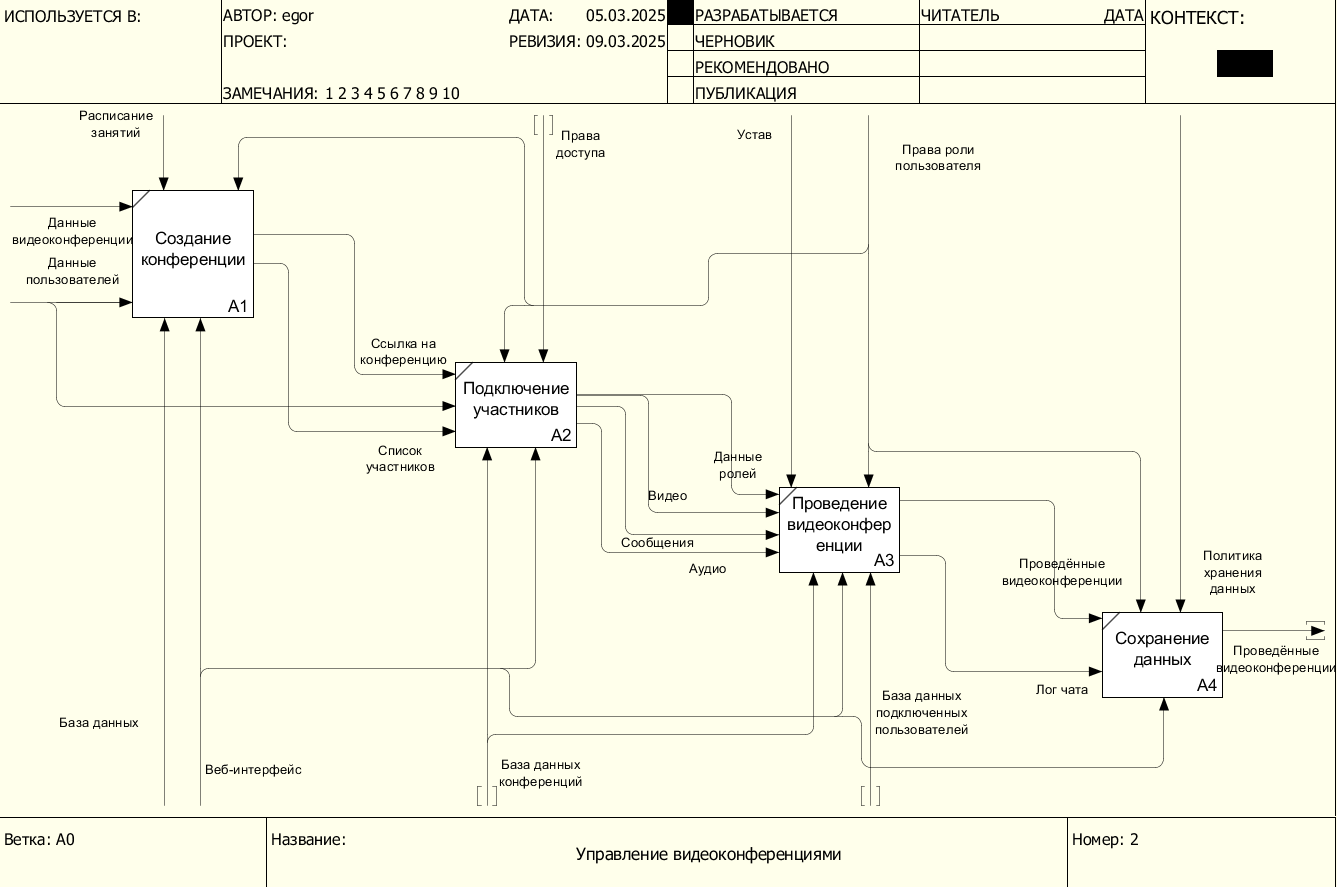


Рисунок 29 – Декомпозиция диаграммы «Управление видеоконференциями»

Преподаватели, организующие видеоконференции для проведения онлайн-занятий, вынуждены использовать сторонние сервисы, такие как Zoom, Microsoft Teams или Google Meet. Для создания конференции преподаватель вручную задаёт параметры встречи (дату, время, список участников) и отправляет студентам ссылку на подключение по электронной почте или в мессенджере.

Подключение студентов к конференции также сопровождается трудностями: ссылки могут потеряться, а доступ иногда ограничивается настройками сервиса.

1. Диаграмма to-be

* Управление заданиями

В новой автоматизированной системе процесс управления заданиями станет более удобным и эффективным за счёт интеграции всех этапов в единой платформе.

Для того чтобы подробно отобразить улучшенный процесс управления заданиями, будет использована методология функционального моделирования IDEF0. Диаграмма To-Be наглядно продемонстрирует все этапы процесса — от создания задания до его проверки и оценки — в условиях автоматизированной системы.

Контекстная диаграмма IDEF0 to-be «Управление заданиями» изображена на рисунке 30:

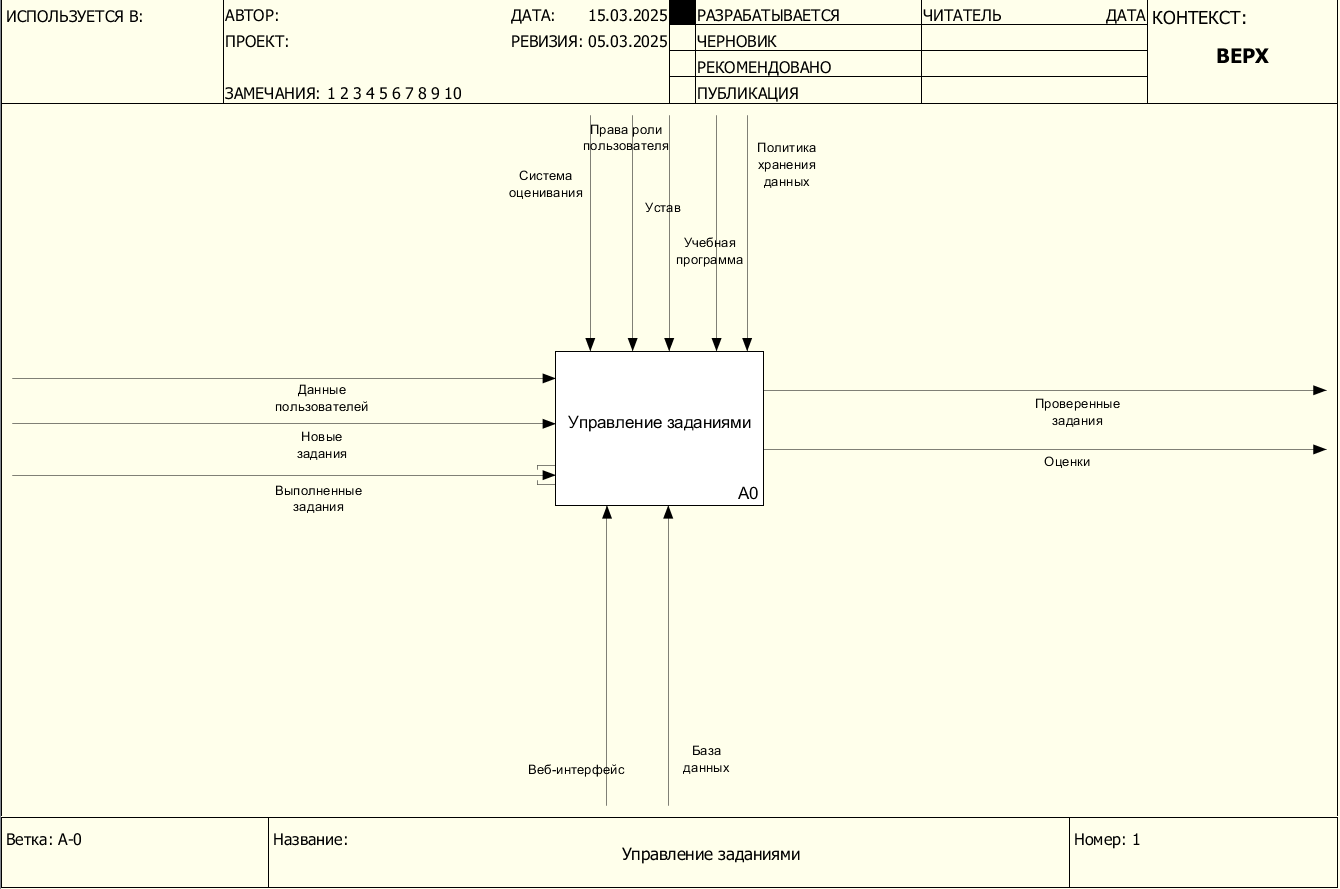


Рисунок 30 – Диаграмма to-be «Управление заданиями» в нотации IDEF0

Декомпозицию контекстной диаграммы IDEF0 to-be «Управление заданиями», имеющей идентификатор A0, можно увидеть на рисунке 31:

Преподаватель сможет создавать и размещать задания непосредственно в системе, задавая параметры выполнения, дедлайны и прикрепляя необходимые материалы. Студенты автоматически получат уведомления и доступ к этим заданиям в своём личном кабинете, что исключит необходимость поиска информации в сторонних сервисах или ожидания уведомлений по почте.

После выполнения задания студенты смогут загрузить свою работу прямо в систему, где она автоматически прикрепится к соответствующему заданию и станет доступной для проверки. Это упростит процесс сдачи работ и минимизирует риск потери данных.

Преподаватель сможет проверять сданные задания в системе, оставлять комментарии и выставлять оценки. Информация о результатах автоматически отобразится в электронном журнале, а студенты получат уведомление о выставленных баллах.



Рисунок 31 – Декомпозиция диаграммы «Управление заданиями»

* Управление видеоконференциями

В новой автоматизированной системе управление видеоконференциями станет более организованным и функциональным благодаря интеграции ключевых процессов в единой платформе и добавлению новых элементов управления и взаимодействия.

Для наглядного представления улучшенного процесса управления видеоконференциями будет использована методология функционального моделирования IDEF0. Диаграмма To-Be отразит новые функции и оптимизированные процессы, направленные на повышение удобства и качества проведения онлайн-занятий.

Контекстная диаграмма IDEF0 to-be «Управление видеоконференциями» изображена на рисунке 32:

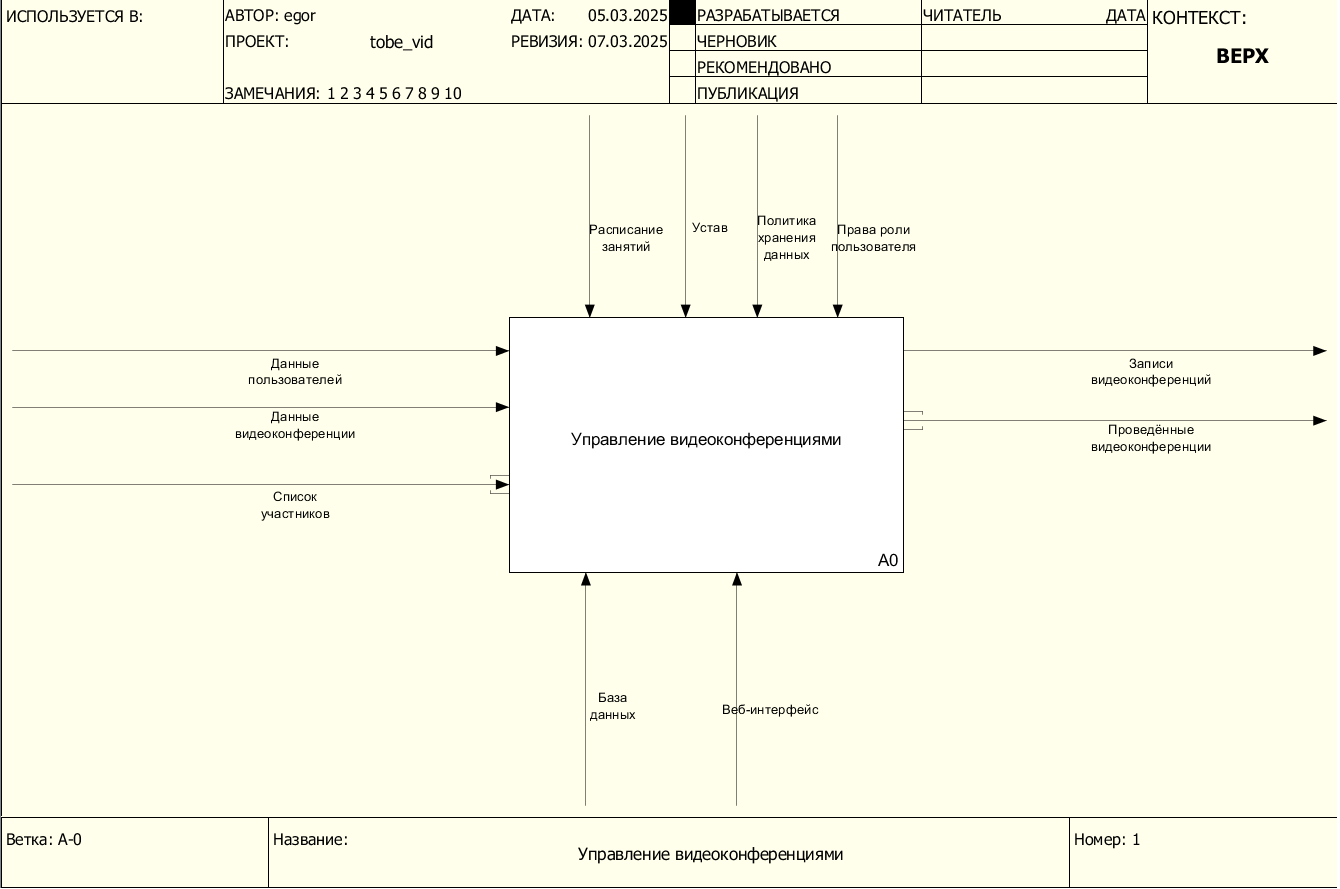


Рисунок 32 – Диаграмма to-be «Управление видеоконференциями» в нотации IDEF0

Декомпозицию контекстной диаграммы IDEF0 to-be «Управление видеоконференциями», имеющей идентификатор A0, можно увидеть на рисунке 33:

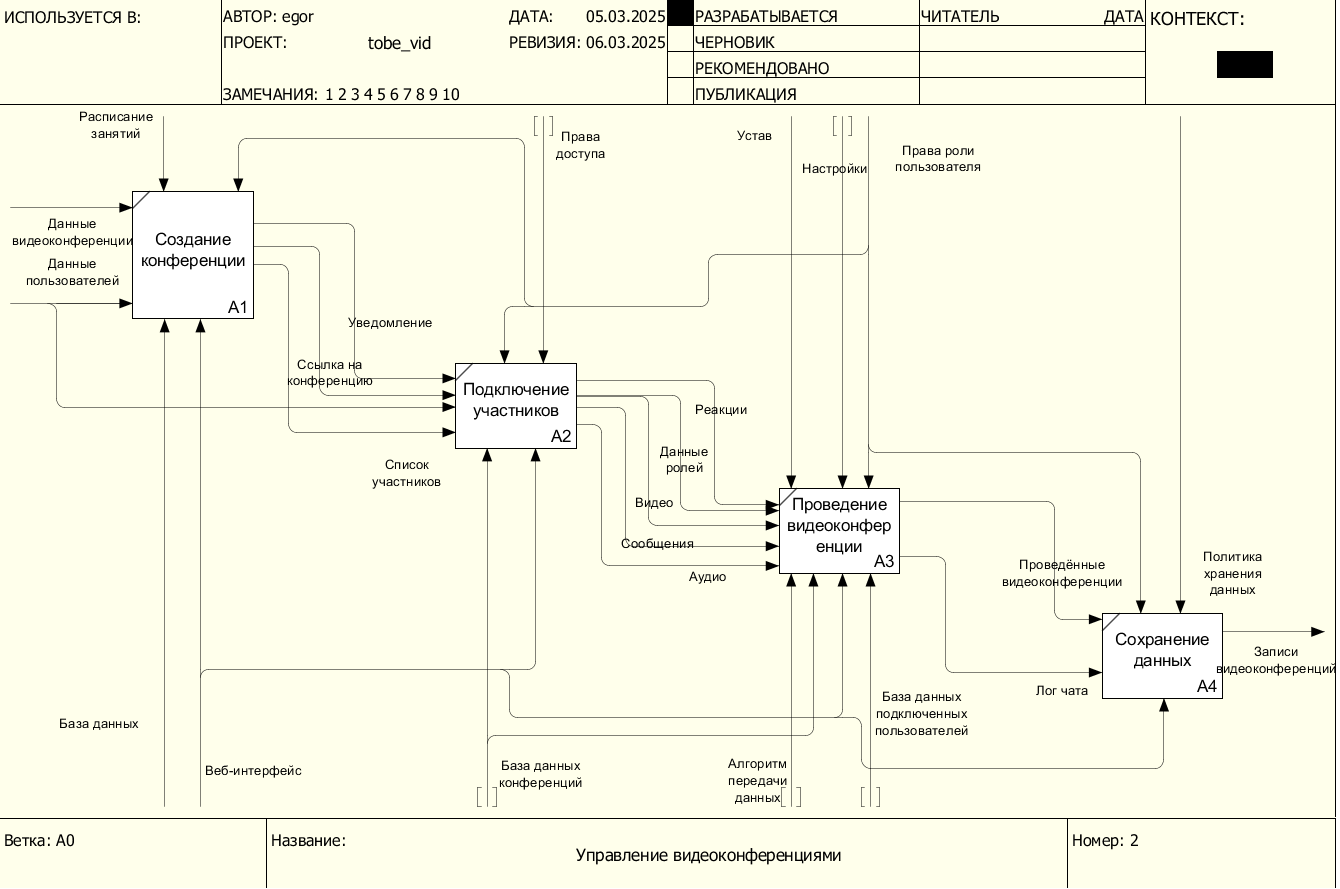


Рисунок 33 – Декомпозиция диаграммы «Управление видеоконференциями»

Преподаватель сможет не только создать видеоконференцию в системе, но и автоматически уведомить всех участников о её параметрах (дата, время, ссылка на подключение). Это устранит необходимость отправлять ссылки вручную и снизит риск пропуска занятия.

При подключении к конференции участники получат уведомление о начале встречи, а система предоставит возможность отправлять реакции (например, поднятие руки, жесты одобрения), что упростит взаимодействие без прерывания речи преподавателя.

Процесс проведения видеоконференции станет более гибким:

Добавление реакций обеспечит возможность учитывать сигналы участников в реальном времени.

Настройки позволят пользователям настраивать свои параметры окна видеоконференций с их последующим сохранением.

Улучшение алгоритма передачи данных повысит качество видеосвязи и обеспечит стабильную работу при большом числе участников.

По завершении конференции система автоматически сохранит её запись, что позволит студентам, пропустившим занятие, просмотреть материал в удобное время.

* Управление электронным журналом

В новой автоматизированной системе процесс управления электронным журналом будет значительно упрощён и улучшен за счёт интеграции всех этапов в единой платформе.

Для наглядного представления нового процесса управления электронным журналом будет использована методология функционального моделирования IDEF0. Диаграмма To-Be отразит последовательность всех этапов процесса, а также продемонстрирует улучшения, направленные на повышение точности учёта и удобства работы с журналом.

Контекстная диаграмма IDEF0 to-be «Управление электронным журналом» изображена на рисунке 34:

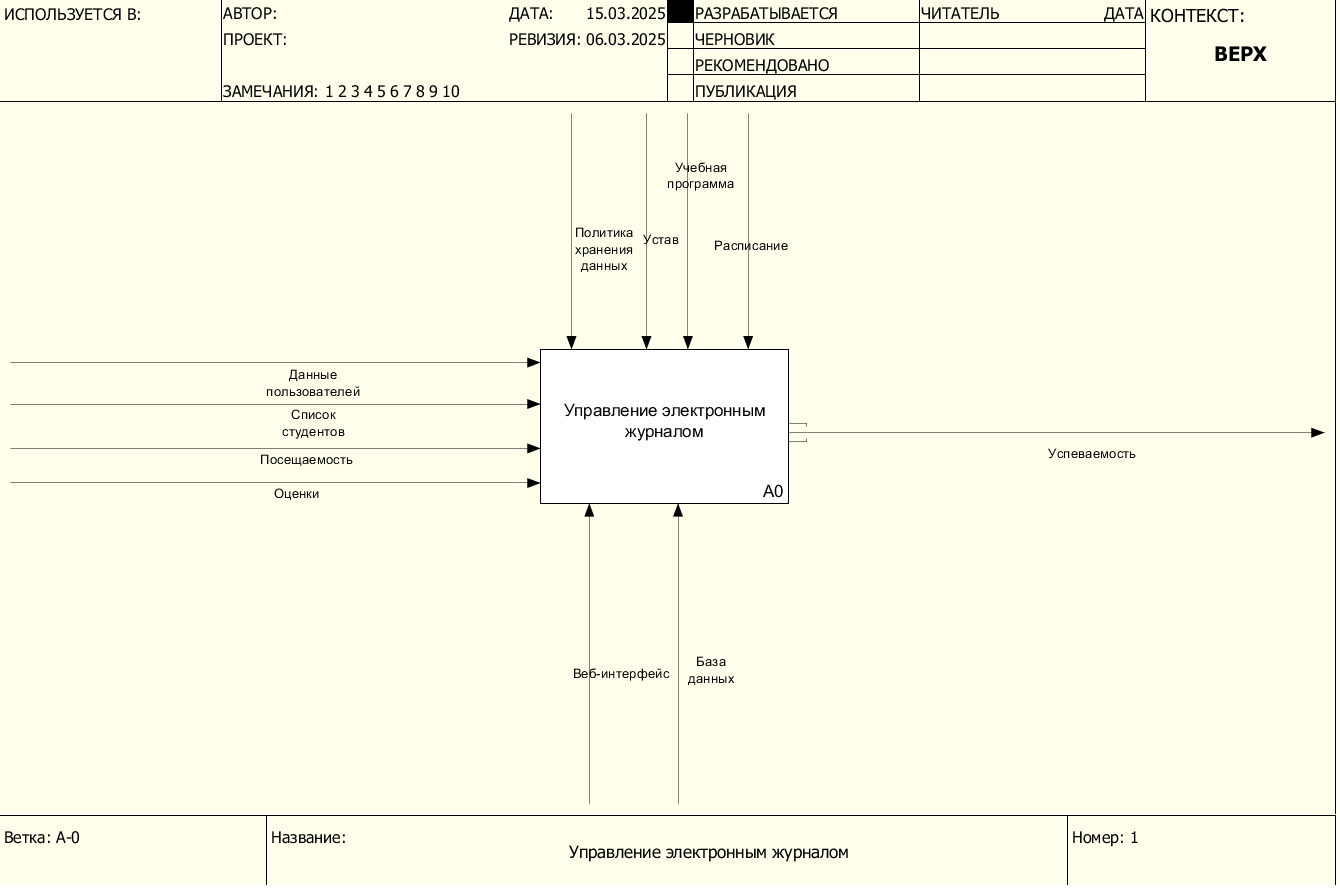


Рисунок 34 – Диаграмма to-be «Управление электронным журналом» в нотации IDEF0

Декомпозицию контекстной диаграммы IDEF0 to-be «Управление электронным журналом», имеющей идентификатор A0, можно увидеть на рисунке 35:

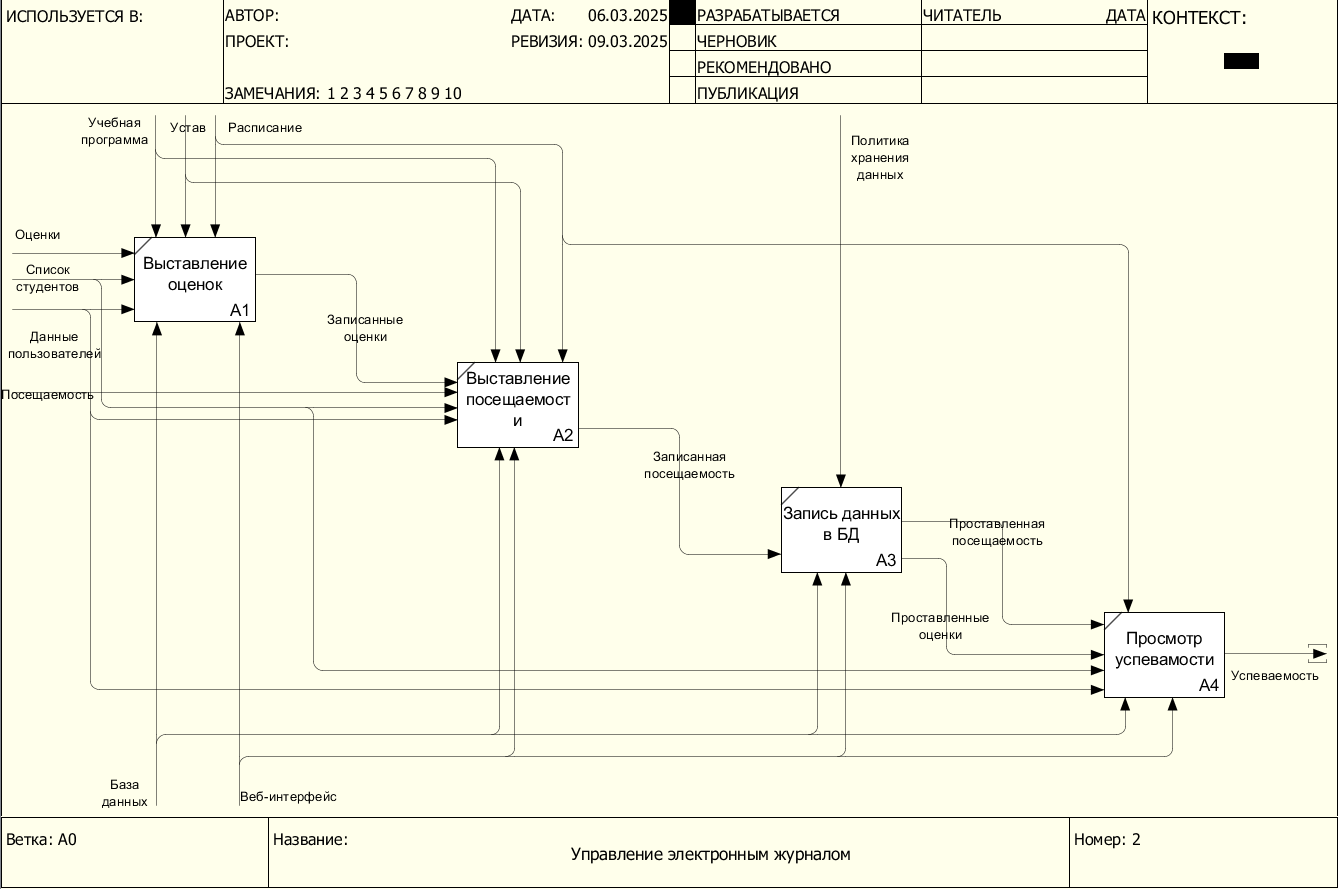


Рисунок 35 – Декомпозиция диаграммы «Управление электронным журналом»

Преподаватель сможет выставлять оценки и посещаемость непосредственно в системе, где они сразу сохранятся в базе данных, исключая необходимость ручного ведения таблиц или дублирования данных в сторонних программах. Это обеспечит более точный учёт результатов и снизит вероятность ошибок.

Все внесённые данные будут автоматически записываться в базу данных. Это обеспечит целостность данных и избавит от необходимости вручную формировать и пересылать файлы с успеваемостью.

Студенты и преподаватели получат возможность в любое время просматривать актуальную информацию об успеваемости через удобный интерфейс. Это повысит прозрачность образовательного процесса и позволит оперативно реагировать на изменения в оценках или данных о посещаемости.

* + 1. **Диаграмма вариантов использования UML**

В системе предусмотрено несколько экторов: студент, преподаватель и администратор. Пользователь «преподаватель» наследует варианты использования пользователя «студент». Пользователь «администратор» наследует варианты использования пользователя «преподаватель».

Сценарий для всех пользователей показан на рисунке 37:

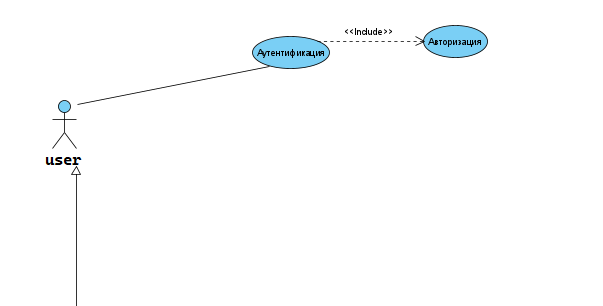


Рисунок 37 – Диаграмма use-case – пользователь

Согласно диаграмме вариантов использования для всех пользователей системы выявлены следующие преценденты: аутентификация.

1. Аутентификация
   1. Основной сценарий
2. Пользователь вводит данные учётной записи
3. Пользователь нажимает кнопку «войти»
4. Происходит переадресация на главную страницу
   1. Расширения
5. Пользователь вводит неверные данные учётной записи

a.1. Система отправляет сообщение об ошибке

Сценарий пользователя «Студент» показан на рисунке 38:

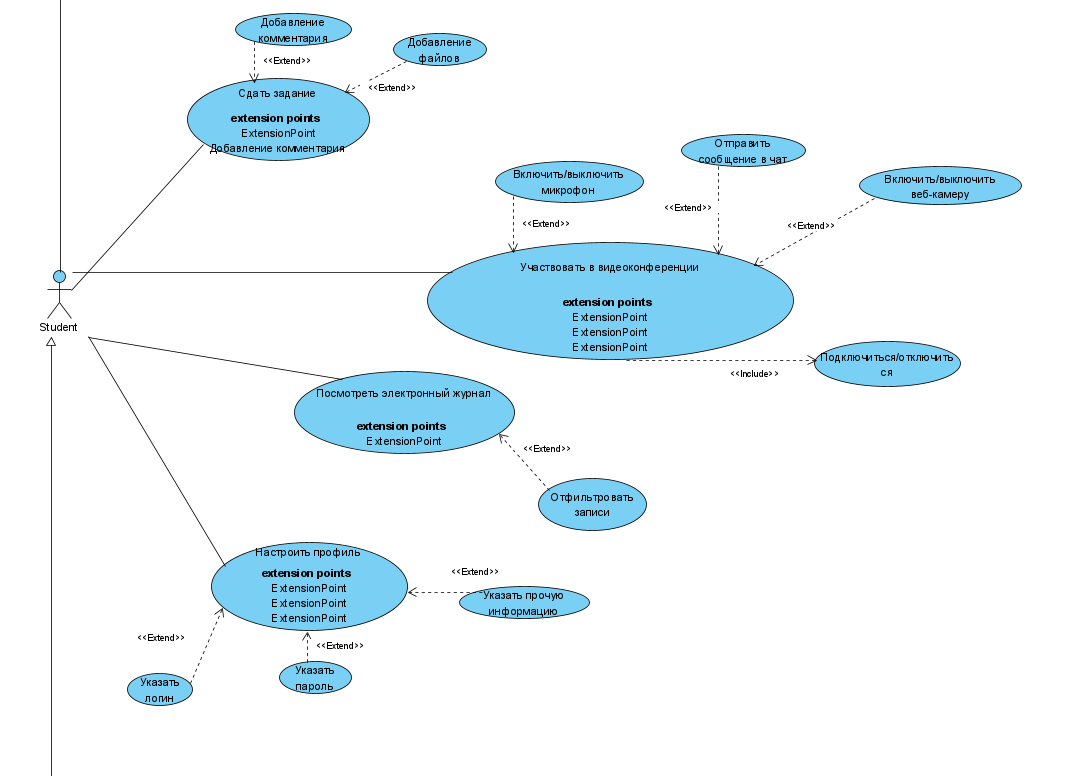


Рисунок 38 – Диаграмма use-case – студент

Согласно диаграмме вариантов использования для пользователя системы «студент» выявлены следующие преценденты: сдать задание, участвовать в видеоконференции, посмотреть электронный журнал, настроить профиль. Подробное описание вариантов использования:

1. Сдать задание
   1. Основной сценарий
2. Пользователь переход на страницу «Задания»
3. Выбирает задание

b.1. Заполняет форму сдачи

b.2. Нажимает кнопку «отправить»

1. Система отправляет сообщение об успехе отправки
2. Кнопка «сдать» меняется на кнопку «отменить»
   1. Расширения
3. Пользователь прикрепляет заражённые вирусом файлы

a.1. Система отправляет сообщение об ошибке

1. Участвовать в видеоконференции
   1. Основной сценарий
2. Пользователь переходит на страницу «Видеоконференции»
3. Выбирает видеоконференцию

b.1. Нажимает кнопку «подключиться»

1. Система предоставляет доступ пользователю

c.1. Система загружает настройки пользователя

3.2Пользователь переходит к окну видеоконференции

1. Пользователь отключает/включает микрофон
2. Пользователь отключает/включает веб-камеру
3. Пользователь отправляет сообщение в чат конференции
4. Пользователь участвует в конференции
5. Пользователь отключается
   1. Расширения
6. Пользователь отключает/включает микрофон

a.1. Система отключает/включает передачу голосовых данных пользователя

a.2. Система сохраняет настройки пользователя

1. Пользователь отключает/включает веб-камеру

b.1. Система отключает/включает передачу видео данных пользователя

b.2. Система сохраняет настройки пользователя

1. Пользователь отправляет реакцию

c.1. Система показывает участникам видеоконференции реакцию пользователя

1. Пользователь отправляет сообщение в чат

d.1. Пользователь не написал ничего

d.1.1. Пустое сообщение не будет отправлено

d.2. Пользователь стёр сообщение и ничего не написал

d.2.1. Пустое сообщение не будет отправлено

d.3. Пользователь вписал xss-код в сообщение

d.3.1. Сообщение не появится на странице

1. Посмотреть электронный журнал
   1. Основной сценарий
2. Пользователь переходит на страницу «Электронный журнал»

a.1. Пользователь смотрит записи в журнале

1. Настроить профиль
   1. Основной сценарий
2. Пользователь переходит на страницу «Профиль»
3. Пользователь нажимает кнопку «редактировать»

b.1. Пользователь изменяет картинку профиля

b.1.1 Пользователь выбирает новую картинку

b.2. Пользователь изменяет прочую информацию (ФИО, email и т.д.)

b.2.1 Пользователь указывает данные

b.3 Пользователь изменяет пароль

b.3.1 Пользователь указывает новый пароль

* 1. Расширения

1. Пользователь выбирает картинку для профиля слишком большого размера

a.1. Система отправляет сообщение об ошибке

1. Пользователь оставляет обязательное поле пустым

b.1. Система предупреждает «о незаполненности обязательного поля»

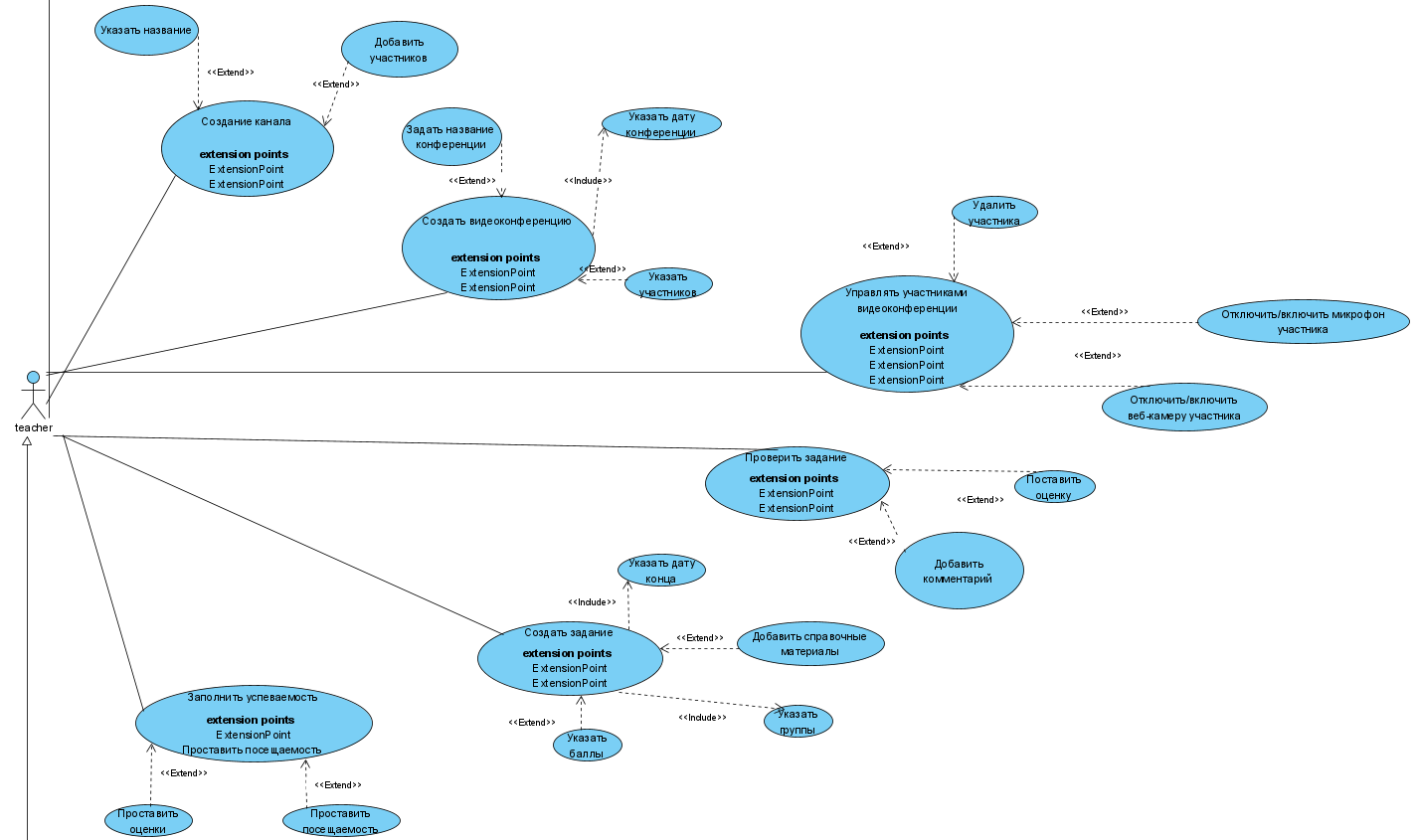
Далее на рисунке 39 будет приведена диаграмма вариантов использования UML для пользователя «преподаватель»:

Рисунок 39 – Диаграмма use-case – преподаватель

В соответствии с диаграммой вариантов использования для пользователя системы «преподаватель» определены следующие преценденты: создать задание, создать видеоконференцию, управлять участниками видеоконференции, написать сообщение в канал, проверить задание, создать задание, заполнить успеваемость. Подробное описание данных сценариев:

1. Создать задание
   1. Основной сценарий
      1. Пользователь переходит на страницу «Задания»
      2. Пользователь нажимает кнопку «Создать новое»
         1. Пользователь указывает название
         2. Пользователь указывает предмет
         3. Пользователь указывает срок сдачи
         4. Пользователь добавляет файлы
         5. Пользователь добавляет текст
         6. Пользователь выбирает группы, которым будет доступно задание
      3. Новое задание отображается на странице
   2. Расширения
      1. Пользователь не указал предмет/срок сдачи/не добавил файлы и текст
         1. Система предупреждает «о незаполненных полях»
2. Создать видеоконференцию
   1. Основной сценарий
      1. Пользователь переходит на страницу «Видеоконференции»
      2. Пользователь нажимает кнопку «Создать новую»
         1. Пользователь указывает название
         2. Пользователь указывает дату проведения
         3. Пользователь добавляет группы в список участников
         4. Пользователь указывает должна ли быть конференция цикличной
      3. Новая видеоконференция отображается на странице
   2. Расширения
      1. Пользователь не указал дату проведения
         1. Система предупреждает «о незаполненных полях»
3. Написать сообщение в чате
   1. Основной сценарий
      1. Пользователь переходит на страницу чатов
         1. Пользователь выбирает чат
         2. Пользователь пишет сообщение
         3. Пользователь нажимает кнопку «Отправить»
      2. Система отображает сообщение на странице канала
   2. Расширения
      1. Пользователь не написал ничего
         1. Пустое сообщение не будет отправлено
      2. Пользователь стёр сообщение и ничего не написал
         1. Пустое сообщение не будет отправлено
      3. Пользователь вписал xss-код в сообщение
         1. Сообщение не появится на странице
4. Управлять участниками видеоконференции
   1. Основной сценарий
      1. Пользователь присоединяется к видеоконференции
         1. Пользователь отключает/включает возможность управления микрофоном для участника
         2. Пользователь отключает/включает возможность управления веб-камерой для участника
         3. Пользователь отключает/включает звук для участника
         4. Пользователь выгоняет участника
      2. Система отправляет сообщение об успехе операции
5. Проверить задание
   1. Основной сценарий
      1. Пользователь переходит на страницу «Задания»
      2. Пользователь выбирает задание
         1. Пользователь выбирает студента
         2. Пользователь просматривает комментарий студента
         3. Пользователь скачивает файлы студента
         4. Пользователь ставит баллы
         5. Пользователь указывает комментарий к оценке
      3. Система создаёт запись с оценкой в журнале
   2. Расширения
      1. Пользователь не поставил баллы и не указал комментарий
         1. Система предупреждает «о незаполненных полях»
6. Заполнить успеваемость
   1. Основной сценарий
      1. Пользователь переходит на страницу «Электронный журнал»
         1. Пользователь выбирает дату
         2. Пользователь указывает предмет
         3. Пользователь указывает оценку
         4. Пользователь указывает посещаемость
         5. Пользователь выбирает применить к студенту или к группе
            1. Если пользователь выбрал применить к студенту, выбирает студента
            2. Если пользователь выбрал применить к группе, выбирает группу
      2. Новая запись журнала отображается на странице
   2. Расширения
      1. Пользователь не указал студента и группу/предмет/дату
         1. Система предупреждает «о незаполненных полях»
      2. Пользователь выбрал не существующего студента/группу/предмет
         1. Система возвращает сообщение об ошибке

Далее на рисунке 40 будет приведена диаграмма вариантов использования UML для пользователя «администратор»:

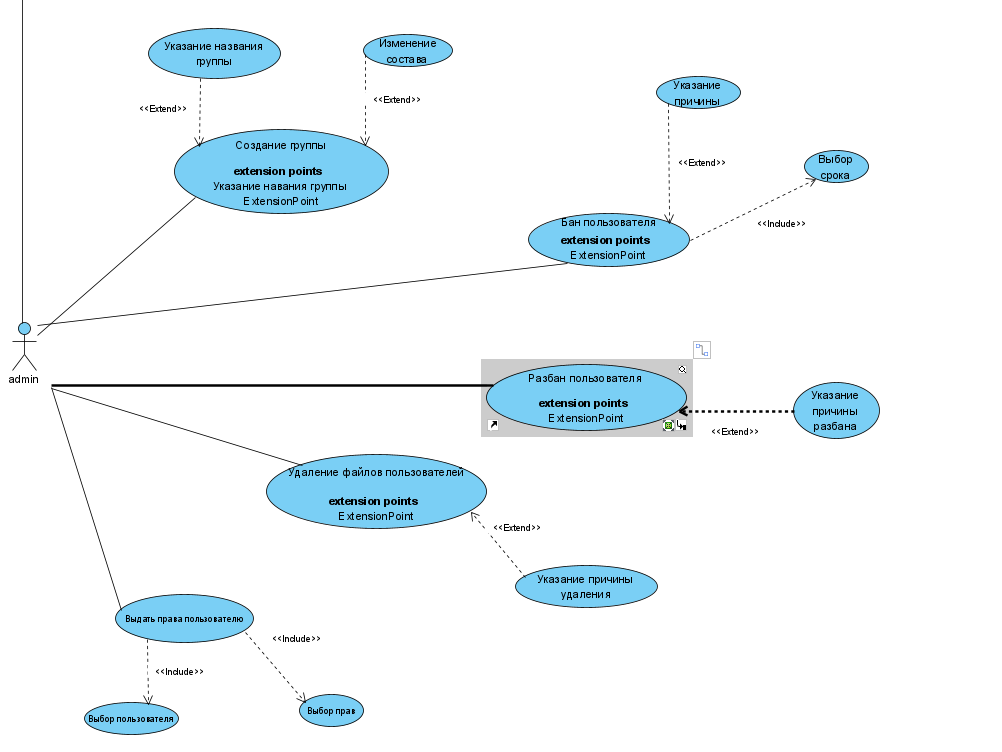


Рисунок 40 – Диаграмма use-case – администратор

Для пользователя «администратор», выполняющего функции модератора и менеджера системы, были выявлены следующие преценденты: создание группы, бан пользователя, разбан пользователя, удаление файлов пользователя, выдача прав пользователю. Подробное описание данных сценариев:

1. Создание группы
   1. Основной сценарий
      1. Пользователь заходит на страницу «Меню администратора»
      2. Пользователь нажимает кнопку «Создать группу»
         1. Пользователь указывает название
         2. Пользователь указывает дополнительную информацию
         3. Пользователь указывает участников
      3. Новая группа отображается в списке всех групп
2. Бан пользователя
   1. Основной сценарий
      1. Пользователь заходит на страницу «Меню администратора»
      2. Пользователь нажимает кнопку «Забанить пользователя»
         1. Пользователь выбирает пользователя для бана
         2. Пользователь указывает причину
         3. Пользователь указывает срок
      3. Забаненный пользователь добавляется в список забаненных пользователей
   2. Расширения
      1. Пользователь выбирает уже забаненного пользователя/другого администратора
         1. Система возвращает сообщение об ошибке
      2. Пользователь выбирает сам себя
         1. Система возвращает предупреждение «о бане себя»
3. Разбан пользователя
   1. Основной сценарий
      1. Пользователь заходит на страницу «Меню администратора»
      2. Пользователь нажимает кнопку «Разбанить пользователя»
         1. Пользователь выбирает пользователя для разбана
      3. Разбаненный пользователь теперь имеет доступ к приложению
   2. Расширения
      1. Пользователь выбирает незабаненного пользователя
         1. Система возвращает сообщение об ошибке
4. Удаление файлов пользователя
   1. Основной сценарий
      1. Пользователь заходит на страницу «Меню администратора»
      2. Пользователь нажимает кнопку «Удалить файлы»
         1. Пользователь выбирает файлы
         2. Пользователь указывает причину
5. Выдача прав пользователям
   1. Основной сценарий
      1. Пользователь заходит на страницу «Меню администратора»
      2. Пользователь нажимает кнопку «Выдать права»
         1. Пользователь выбирает пользователя для выдачи прав
         2. Пользователь выбирает права
      3. Выбранный пользователь имеет новые права
   2. Расширения
      1. Пользователь выдаёт уже имеющиеся права пользователю
         1. Система возвращает сообщение об ошибке
      2. Пользователь меняет права другого администратора
         1. Система возвращает сообщение об ошибке
      3. Пользователь уменьшает свои права
         1. Система предупреждает «об изменении своих прав»

2.2.2 DFD-диаграмма

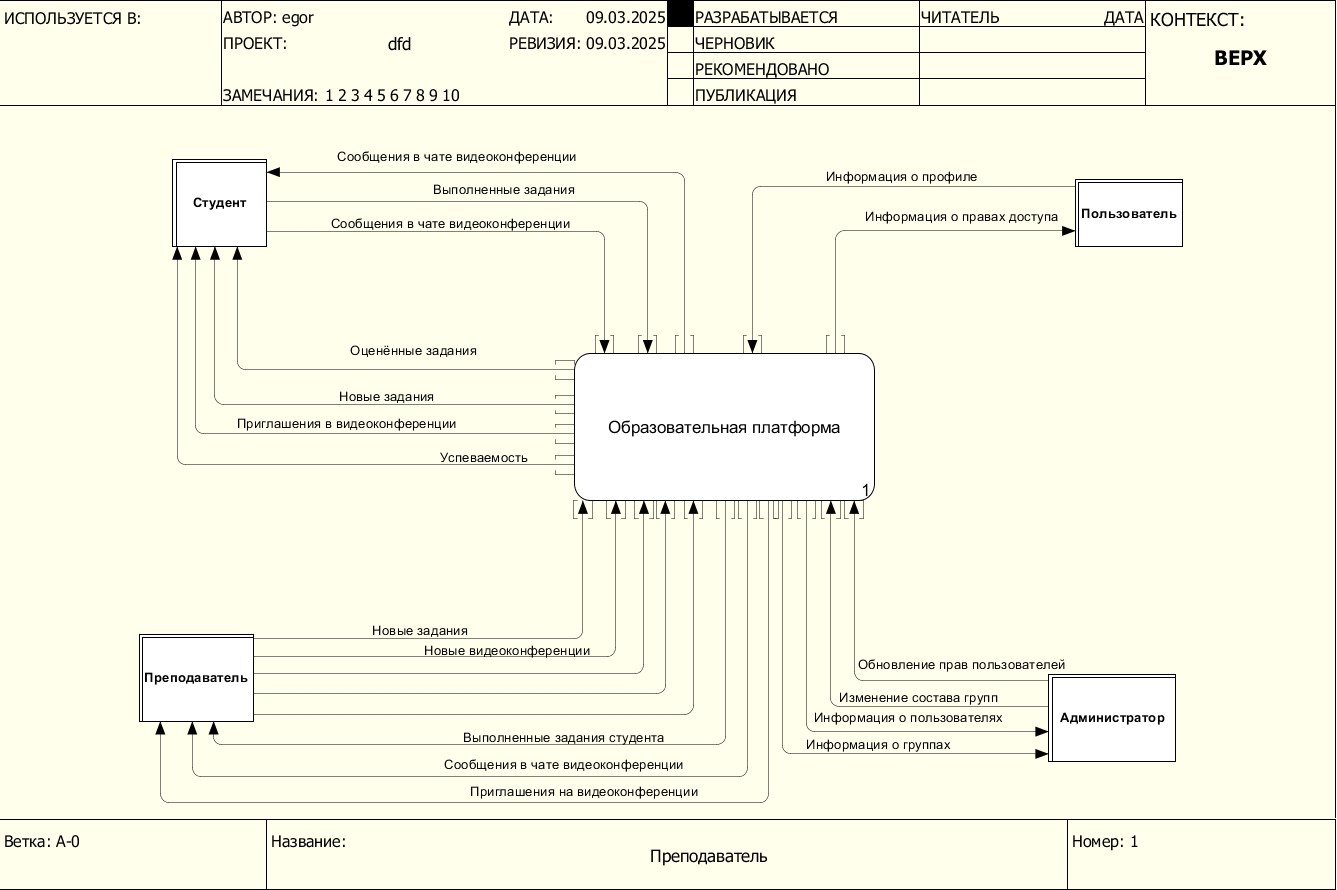
Контекстная dfd-диаграмма системы продемонстрирована на рисунке 41: 

Рисунок 41 – контекстная dfd-диаграмма

Контекстная диаграмма системы образовательной платформы отражает взаимодействие системы с следующими сущностями: пользователь, преподаватель, студент, администратор.

1. Студент

Потоки данных от студента в систему

* Выполненные задания – файлы и текстовые ответы, отправляемые студентом в систему для последующей проверки преподавателем. Содержат файлы распространённых форматов документов, изображений и видео, ответы в виде текста, дату сдачи
* Сообщения в чате видеоконференции – текстовые сообщения, которые студент отправляет в ходе видеоконференции. Содержат имя отправителя, время отправки, текст сообщения

Потоки данных от системы к студенту

* Оценённые задания — результаты проверки сданных студентом работ. Содержат оценку, комментарий преподавателя, дату оценивания
* Новые задания — информация об опубликованных заданиях, включая описание, сроки сдачи и требования. Содержат название, возможную максимальную оценку, текстовое описание, справочные материалы в виде файлов распространённых форматов документов, изображений и видео, дату сдачи
* Успеваемость — сведения о текущих оценках и посещаемости. Содержат оценки, посещаемость.
* Приглашения в конференции — уведомления о предстоящих и уже начавшихся видеоконференциях с указанием даты, времени и ссылки для подключения.
* Сообщения в чате конференции — сообщения от других участников видеоконференции, передаваемые студенту. Содержат имя отправителя, время отправки, текст сообщения

1. Преподаватель

Потоки данных от преподавателя в систему

* Новые задания — информация о созданных заданиях, включая их описание, сроки сдачи и прикреплённые файлы. Содержат название, текстовое описание, дату сдачи, справочные материалы в файлах
* Сообщения в чате конференции — текстовые сообщения, которые преподаватель отправляет в ходе видеоконференции. Содержат имя отправителя, время отправки, текст сообщения
* Оценённые задания — результаты проверки сданных студентом работ. Содержат оценку, комментарий преподавателя, дату оценивания
* Новые видеоконференции — информация о созданных конференциях. Содержит дату проведения, название предмета, название конференции, список участников
* Заполненная успеваемость — сведения об оценках и посещаемости, внесённые преподавателем в электронный журнал.

Потоки данных от системы к преподавателю

* Выполненные задания студента — файлы и текстовые ответы, которые студенты отправляют в систему для проверки. Содержат файлы распространённых форматов документов, изображений и видео, ответы в виде текста, дату сдачи
* Приглашения в конференции — уведомления о созданных или запланированных видеоконференциях, в которых преподаватель является участником.
* Сообщения в чате видеоконференции — текстовые сообщения от участников, поступающие в ходе конференции. Содержат имя отправителя, время отправки, текст сообщения

1. Пользователь

Поток данных от пользователя в систему

* Информация о профиле – данные, которые пользователь предоставляет при регистрации или редактирует в личном кабинете. Содержит ФИО, номер студенческого билета, email, пароль, логин, проверочный вопрос, ответ на проверочный вопрос, дату рождения, аватар

Поток данных от системы к пользователю

* Информация о правах доступа – сведения о доступных пользователю функциях и возможностях в системе на основе его роли

1. Администратор

Поток данных от администратора в систему

* Обновление прав пользователей — данные, содержащие информацию о присвоении или снятии определённых ролей
* Изменение состава групп — данные о добавлении, удалении или перераспределении студентов по группам

Поток данных от системы к администратору

* Информация о группах — список существующих групп и их состав. Содержит название группы, список участников, курс
* Информация о пользователях — сведения о зарегистрированных пользователях, полная информация об их профилях

Декомпозиция dfd-диаграммы представлена на рисунке 42:

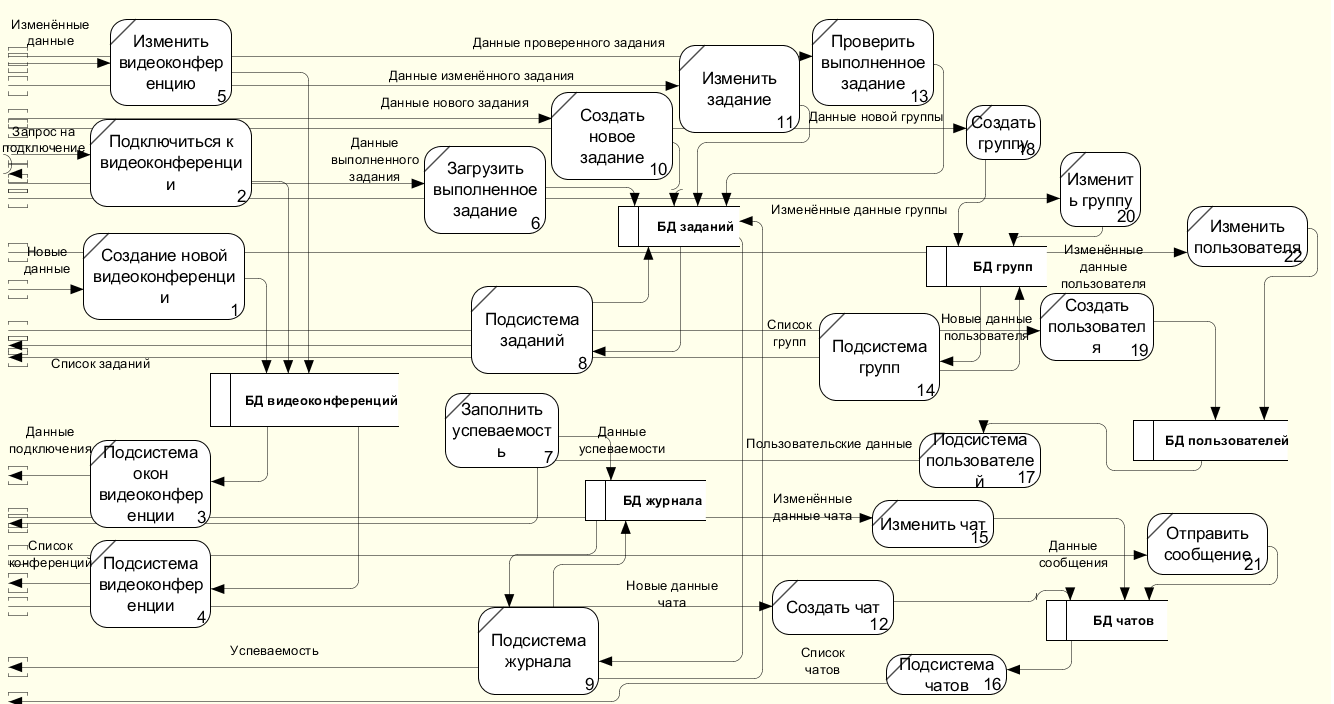


Рисунок 42 – декомпозиция dfd-диаграммы

Все элементы декомпозиции можно разбить на 6 подгрупп: задания, видеоконференции, журнал, пользователи, группы, каналы.

1. Задания

Входные процессы:

* Изменить задание

Процесс отвечает за внесение изменений в раннее созданное задание.

Входной поток – данные изменённого задания. Содержит название, тестовое описание, файлы, дату сдачи.

Действие – при поступлении данных система проверяет их корректность и передаёт информацию в базу данных заданий для обновления соответствующей записи.

* Загрузить выполненное задание

Процесс отвечает за добавление выполненных студентами заданий в систему.

Входной поток – данные выполненного задания. Содержит файлы, текст, данные студента

Действие – при поступлении данных система фиксирует и сохраняет их в БД.

* Создать новое задание

Процесс отвечает за создание и размещение новых заданий в системе.

Входной поток – данные нового задания. Содержит название, дату сдачи, текстовое описание, файлы, автора, список групп, получающих задание, название предмета.

Действие – система проверяет корректность введённых данных и записывает информацию в базу данных.

* Проверить выполненное задание

Процесс отвечает за фиксацию результатов проверки сданных работ.

Входной поток – данные проверенного задания. Содержит оценку, комментарий преподавателя, дату проверки.

Действие – система обновляет данные в базе данных и добавляет результат в список проверенных работ.

* Подсистема заданий

Отвечает за предоставление и обработку данных о всех заданиях, хранящихся в системе.

Входной поток – данные о заданиях, получаемые из БД заданий.

Хранилище данных:

* База данных заданий

Хранит данные всех заданий, размещённых в системе.

Исходящие процессы:

* Подсистема заданий

Отвечает за предоставление данных о всех заданиях, хранящихся в системе.

Выходной поток – список заданий. Содержит все задания, размещённые в системе.

* Подсистема журнала

Получает данные об оценённых заданиях.

Выходной поток – успеваемость. Содержит оценки, посещаемость.

1. Видеоконференции

Входные процессы:

* Изменить видеоконференцию

Отвечает за внесение изменений в раннее созданную видеоконференцию.

Входной поток – изменённые данные конференции. Содержит название, дату проведение, название предмета, список участников.

Действие – при поступлении данных система проверяет их корректность и передаёт информацию в базу данных заданий для обновления соответствующей записи.

* Подключиться к видеоконференции

Отвечает за идентификацию конференции, проверку доступа и предоставление пользователю данных для подключения.

Входной поток – запрос на подключение. Содержит данные конференции, к которой производится подключение, данные пользователя.

Действие – при поступлении данных система проверяет их корректность, права доступа пользователя и отправляет ответ пользователю.

* Создание новой видеоконференции

Отвечает за создание новых видеоконференций в системе

Входной поток – данные новой видеоконференции

Действие – система проверяет корректность введённых данных и записывает информацию в базу данных.

Хранилище данных:

* База данных видеоконференций

Хранит данные всех видеоконференций, размещённых в системе.

Исходящие процессы:

* Подсистема окон видеоконференции

Отвечает за подключение пользователя к видеоконференции и трансляцию видеоконференции.

Выходной поток – данные подключения. Содержит идентификатор пользователя, настройки пользователя, ФИО пользователя, роль пользователя, идентификатор конференции.

* Подсистема видеоконференции

Отвечает за предоставление и обработку данных о всех видеоконференциях, размещённых в системе.

Выходной поток – список видеоконференций.

1. Журнал

Входные процессы:

* Заполнить успеваемость

Отвечает за добавление данных успеваемости в систему.

Входной поток – данные успеваемости. Содержит оценки, посещаемость, ФИО студента, дату, идентификатор задания, название предмета.

Действие – система проверяет корректность введённых данных и записывает информацию в базу данных.

* Подсистема журнала

Отвечает за предоставление и обработку данных об успеваемости, хранящихся в системе.

Выходной поток – успеваемость. Содержит оценки, посещаемость, ФИО студента, дату, идентификатор задания, название предмета.

Хранилище данных:

* База данных журнала

Хранит данные об успеваемости всех студентов.

Исходящие процессы:

* Подсистема журнала

Отвечает за предоставление и обработку данных об успеваемости, хранящихся в системе.

Выходной поток – успеваемость. Содержит оценки, посещаемость, ФИО студента, дату, идентификатор задания, название предмета.

1. Пользователи

Входные процессы:

* Создать пользователя

Отвечает за добавление нового пользователя в систему.

Входной поток – данные нового пользователя. Содержат ФИО, номер студенческого, email, логин, пароль, проверочный вопрос, ответ на проверочный вопрос, роль, название группы, аватар.

Действие – система проверяет корректность введённых данных и записывает информацию в базу данных.

* Изменить пользователя

Отвечает за внесение изменений в данные раннее добавленного пользователя.

Входной поток – изменённые данные пользователя. Содержит ФИО, номер студенческого, email, логин, пароль, проверочный вопрос, ответ на проверочный вопрос, роль, название группы, аватар.

Действие – при поступлении данных система проверяет их корректность и передаёт информацию в базу данных пользователей для обновления соответствующего профиля.

Хранилище данных:

* База данных пользователей

Хранит данные о всех пользователях системы.

Исходящие процессы:

* Подсистема пользователей

Отвечает за предоставление и обработку данных о пользователях системы.

Выходной поток – пользовательские данные. Содержит ФИО, номер студенческого, email, логин, пароль, проверочный вопрос, ответ на проверочный вопрос, роль, название группы, аватар.

1. Группы

Входящие процессы:

* Создать группу

Отвечает за добавление новой группы в систему.

Входной поток – данные новой группы. Содержит название, список участников.

Действие – система проверяет корректность введённых данных и записывает информацию в базу данных.

* Изменить группу

Отвечает за внесение изменений в ранее добавленную группу.

Входной поток – изменённые данные группы. Содержит название, список участников.

Действие - при поступлении данных система проверяет их корректность и передаёт информацию в базу данных групп для обновления соответствующей группы.

* Подсистема групп

Отвечает за предоставление и обработку данных о группах в системе.

Выходной поток – список групп. Содержит название, список участников.

Хранилище данных:

* База данных групп

Хранит данные о всех группах в системе.

Исходящие процессы:

* Подсистема групп

Отвечает за предоставление и обработку данных о группах в системе.

Выходной поток – список групп. Содержит название, список участников.

1. Чаты

Входящие процессы:

* Создать чат

Отвечает за добавление нового канала в систему.

Входной поток – данные нового чата. Содержит название, список участников.

Действие – система проверяет корректность введённых данных и записывает информацию в базу данных.

* Изменить чат

Отвечает за внесение изменений в ранее добавленный чат.

Входной поток – изменённые данные канала. Содержит название, список участников, историю сообщений.

Действие – при поступлении данных система проверяет их корректность и передаёт информацию в базу данных каналов для обновления соответствующего чата.

* Отправить сообщение

Отвечает за добавление новых сообщений в чат.

Входной поток – данные сообщения. Содержит идентификатор отправителя, логин отправителя, текст сообщения, дату отправки, идентификатор сообщения, на которое отвечают.

Действие – система проверяет корректность введённых данных и записывает информацию в базу данных.

Хранилище данных:

* База данных чатов

Хранит данные о всех чатах в системе.

Исходящие процессы:

* Подсистема чатов

Отвечает за предоставление и обработку данных о чатах в системе.

Выходной поток – список чатов. Содержит название, список участников, историю сообщений.

2.2.3 **Формирование требований к системе**

Исходя из задач, поставленных в главе 1 можно сформулировать требования к разрабатываемой информационной системе. В рамках выпускной квалификационной работы было выбрано описание требований к системе на основе метода атрибутов качества (как подраздел FURPS+).

1. Функциональные

1.1. Электронная почта

* При восстановлении пароля используется email, на который высылается временный пароль.
* Если email при регистрации не указан — требуется указание нового email.

1.2. Безопасность

* Аутентификация пользователей обязательна.
* Пароли хранятся в хешированном виде с использованием алгоритма Argon2 и «соли».
* Создание резервных копий — не реже 1 раза в 24 часа.
* При DDoS-атаке применяется фильтрация IP-адресов и ограничение запросов до устранения угрозы.
* Доступ к Меню Управления (МУ) разрешён только администраторам. Для других пользователей он блокируется.

1.3. Управление системой

* Через панель МУ администратор выполняет CRUD-операции над данными системы и БД.

1. Практичность

К удобству использования относятся следующие виды требований:

1. Пользовательский интерфейс
   1. В интерфейсе должны быть использованы моноширинные шрифты с поддержкой кириллических символов, позволяющие пользователю различать каждый символ.
   2. Цвета элементов системы должны сочетаться на основе рекомендаций WCAG, палитра цветов не должна содержать слишком яркие или слишком контрастные цвета (Яркость и цветовая насыщенность элементов должны обеспечивать достаточную читаемость независимо от различий в калибровке мониторов; Минимальный коэффициент контрастности между текстом и фоном - 4.5:1, рекомендуемый уровень: 7:1).
   3. Количество нажимаемых элементов для перехода к любой функции не должно превышать 3-4 нажатия.
   4. Название и описание элементов должно быть лаконичным и чётко сформулированным.
   5. После отправки запроса на сервер посредством форм или кнопок пользователь должен получать уведомление об успехе или провале операции
   6. Интерфейс приложения должен быть выполнен на русском языке
2. Документация
   1. руководство пользователя для ознакомления с основными функциями
   2. руководство администратора
   3. подробная документация функций и архитектуры системы
   4. правила использования приложения для пользователей
3. Квалификация пользователей:
   1. все пользователи системы должны владеть персональным компьютером/смартфоном на уровне пользователя
   2. пользователи с ролью «администратор» должны иметь базовое представление о работе функций системы
4. Готовность

Готовность включает такие характеристики системы, как:

1. Устойчивость к сбоям
   1. возможная периодичность сбоев – не более 2 раз в месяц, время реагирования составляет примерно 2-2,5 часа
   2. при большой одновременной нагрузке (более 1000 одновременных запросов от пользователей) должны быть предусмотрены функции ограничения запросов в минуту
2. Предсказуемость поведения
   1. информация, содержащаяся в настоящем списке под заголовком «сбои» и имеющая пункт №2
3. Веб-приложение должно предоставлять доступ к функциям 87% годового времени. Возможны перерывы на технические работы примерно 4-8 часов в месяц.
4. Для обеспечения безопасности пользовательских данных должны быть использованы https протокол, cookie-файлы и csrf-токены.
5. Все данные, необходимые для нормальной работы системы, должны храниться в базе данных, защищённой надёжным паролем и устойчивой к sql-инъекциям
6. Производительность

Производительность системы составляют следующие характеристики:

1. При учёте количества пользователей информационной системы (не более 800 человек одновременно) и типов поступающих от пользователей запросов, которые ограничены направленностью на решение конкретных задач, была определена допустимая средняя задержка в границах от 4 до 6 секунд
2. Шифрование и дешифровка данных должно выполняться за ~20 мс.
3. Время обработки запроса без файлов не должно превышать 5 секунд, с файлами – 9 секунд.
4. Максимальное время, необходимое на восстановление системы после сбоя, не должно превышать трёх часов;
5. Максимальное время для запуска и завершения работы системы – не более 3 минут;
6. потребление ресурсов
   1. время – администрирование системы
   2. деньги – зарплата администраторам, оплата интернета
   3. аппаратные ресурсы – дисковое пространство, ресурсы процессора и оперативной памяти
7. Поддерживаемость

К поддержке относятся:

1. Сервисное обслуживание производит системный администратор, при необходимости вносящий различные необходимые корректировки и обеспечивающий бесперебойную работу системы
2. Для использования информационной системы требуется наличие любого современного браузера (например, MS Edge, Chromium, Safari, Opera, Mozilla Firefox и другие)
3. При выводе продукта в эксплуатацию он должен полностью поддерживать русский язык.
4. Создание резервных копий должно осуществляться не реже, чем 1 раз в 2-ое суток
5. Используются локальные сторонние сервисы резервирования с минимальной нагрузкой на систему.
6. Ограничения

Возможны следующие ограничения:

1. Система должна корректно работать на серверах с ограниченными ресурсами (не менее 2 CPU и 4 ГБ ОЗУ)
2. Доступ к платформе возможен только при наличии интернет-соединения

**ВЫВОДЫ ПО 2 ГЛАВЕ**

Исходя из концептуальной модели управления объекта, было определено взаимодействие различных компонентов, а также был составлен идеальный сценарий работы информационной системы.

Функциональное моделирование, содержащее в себе контекстные диаграммы и декомпозиции «как было» и «как будет», помогло наглядно продемонстрировать различия между тем, как работала и как будут работать такие подсистемы, как подсистема заданий, подсистема видеоконференций и электронный журнал.

UML-диаграмма вариантов использования, отражающая отношения между актёрами и прецедентами, помогла описать все функциональности и поведения, возможные при взаимодействии пользователя с системой.

Благодаря классификации требований FURPS+ были определены функциональные требования, требования к удобству использования и надёжности, производительности и поддерживаемости, а также были выделены различные ограничения.

Таким образом, проведённый анализ показал возможности системы и помог определить необходимые требования.

1. **ПРОЕКТИРОВАНИЕ ИНФОРАМЦИОННОЙ СИСТЕМЫ**
2. ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ
   1. **Проектирование архитектуры информационной системы**

После определения требований к системе, было принято решение использовать многоуровневую архитектуру, состоящую из следующих уровней:

• Уровень представления – отвечает за отображение пользовательского графического интерфейс и за взаимодействие пользователя с приложением через графический интерфейс. Включает контроллеры, принимающие HTTP-запросы, выполняющие первичную валидацию и формирующие ответы.

• Уровень прикладной логики – отвечает за реализацию основной логики приложения. В этом слое выполняется проверка прав, обработка сущностей, вызов репозиториев и организация операций между сущностями.

• Уровень доступа к данным – взаимодействует с базой данных. Отвечает за выполнение CRUD-операций над объектами доменной модели, не содержит бизнес-логики.

• Уровень базы данных – отвечает за хранение и предоставление данных.

Также используется архитектурный паттерн CSR (Controller-Service-Repository), который способствует чёткому разделению ответственности в приложении и повышает читаемость и сопровождаемость кода. Описание паттерна:

1. Controller — принимает и обрабатывает входящие HTTP-запросы. Он отвечает за маршрутизацию, валидацию входных данных и передачу управления соответствующим сервисам.
2. Service — содержит бизнес-логику приложения. Сервисы инкапсулируют операции, которые могут включать обращение к данным через репозитории, обработку и принятие решений.
3. Repository — отвечает за взаимодействие с базой данных. Репозитории предоставляют абстракцию над механизмом хранения данных и реализуются с использованием JPA.

Такое разделение позволяет упростить тестирование, повторное использование компонентов и масштабирование приложения.

На рисунках 43-47 изображена диаграмма классов.

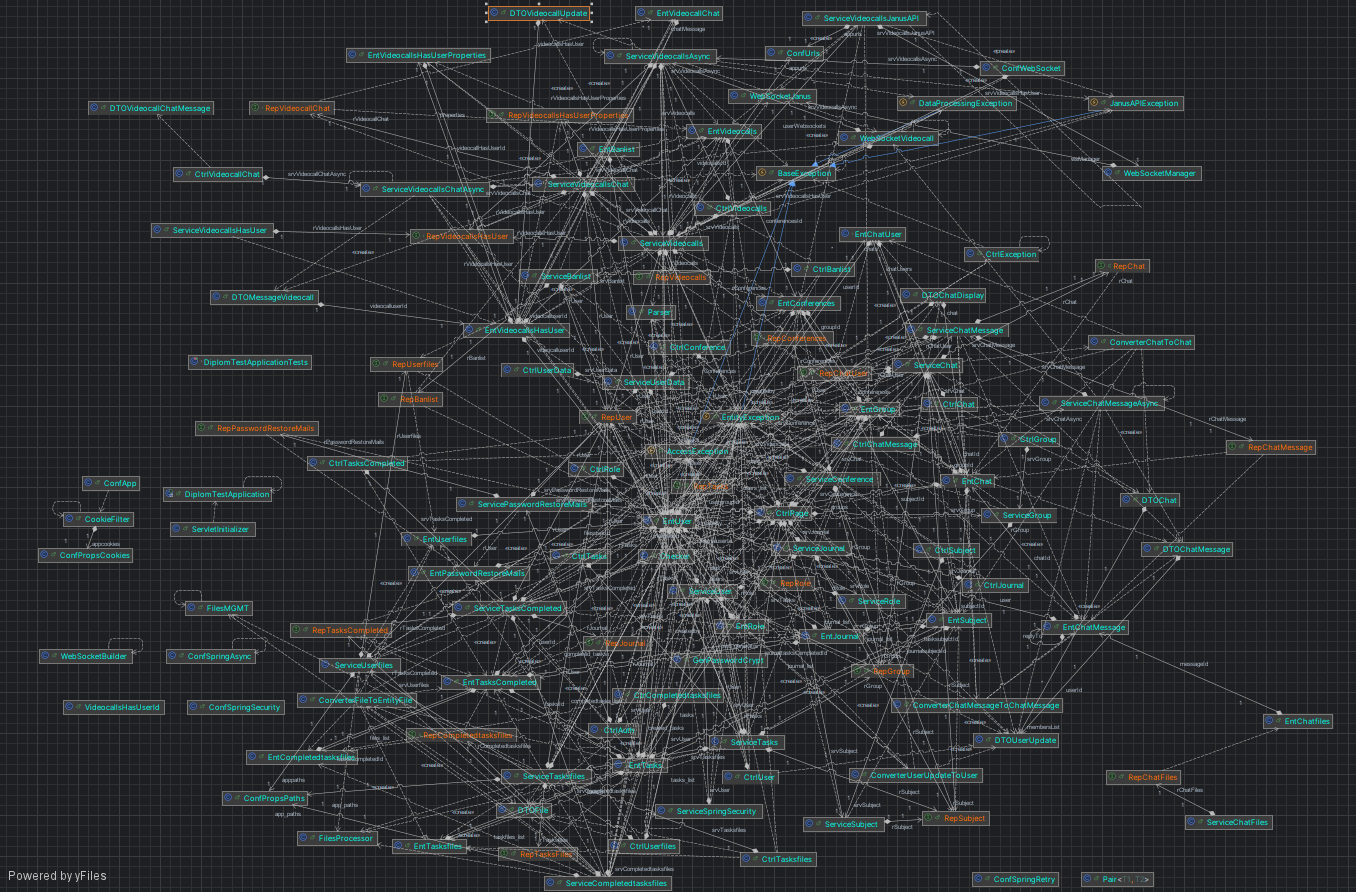


Рисунок 43 – диаграмма классов (1)

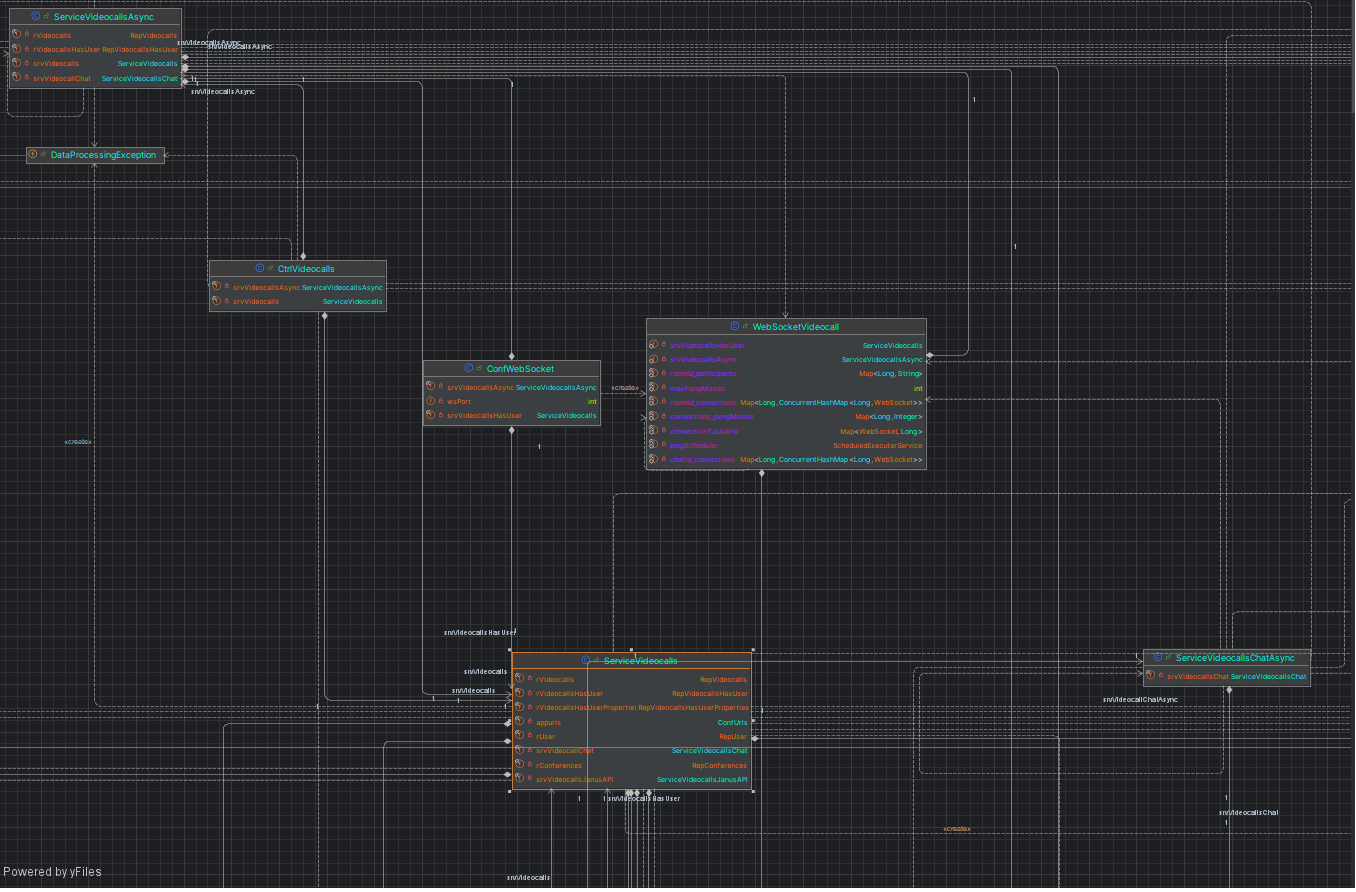


Рисунок 44 – диаграмма классов (2)



Рисунок 45 – диаграмма классов (3)

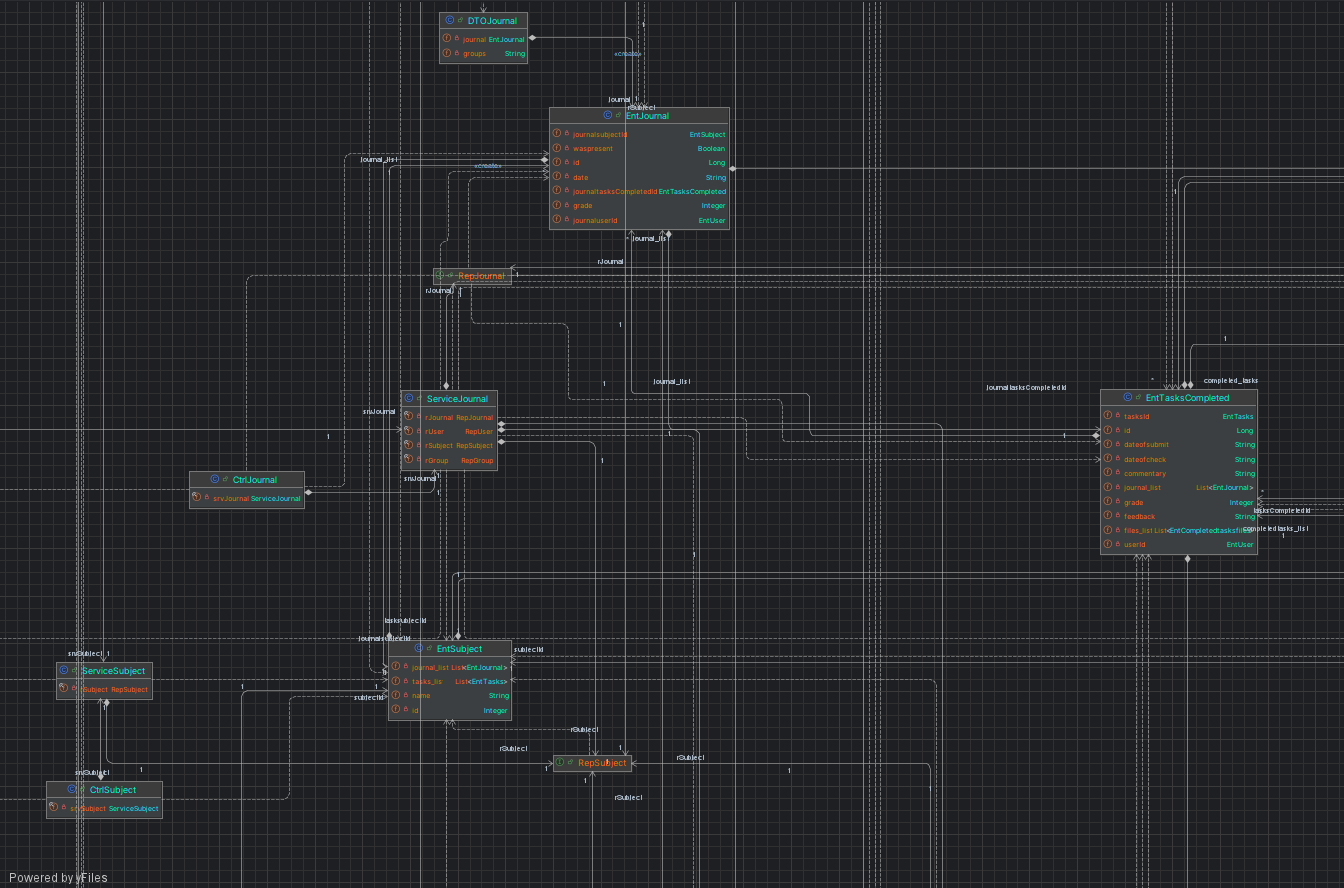


Рисунок 46 – диаграмма классов (4)

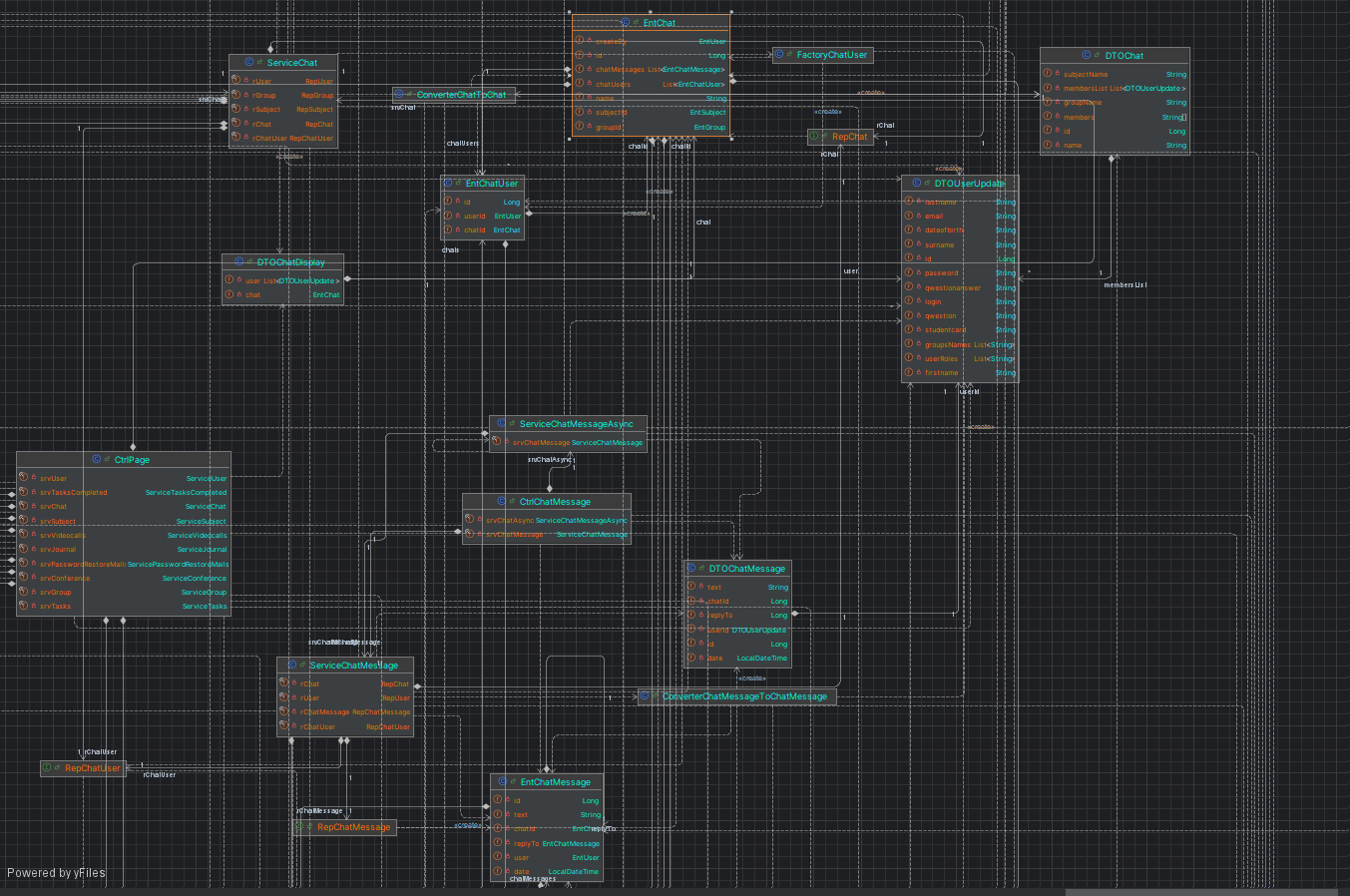


Рисунок 47 – диаграмма классов (5)

Классы, отвечающие за обработку и передачу данных запросов в сервисы:

1. CtrlCompletedtasksfiles – контроллер для управления файлами выполненных заданий;
2. CtrlAuth – контроллер авторизации и регистрации;
3. CtrlBanlist – контроллер для управления банами пользователей;
4. CtrlChat – контроллер для управления чатами;
5. CtrlChatMessage – контроллер для управления сообщениями в чате;
6. CtrlConference – контроллер для управления конференциями;
7. CtrlException – контроллер передачи сообщений об ошибках пользователю;
8. CtrlGroup – контроллер управления группами;
9. CtrlJournal – контроллер управления электронным журналом;
10. CtrlPage – контроллер выдачи страниц приложения;
11. CtrlRole – контроллер управления ролями пользователей;
12. CtrlSubject – контроллер управления предметами;
13. CtrlTasks – контроллер управления заданиями ;
14. CtrlTasksCompleted – контроллер управления выполненными заданиями;
15. CtrlTasksfiles – контроллер управления файлами заданий;
16. CtrlUser – контроллер управления пользователями;
17. CtrlUserData – контроллер получения заданий и конференций для пользователя;
18. CtrlUserfiles – контроллер управления файлами пользователей;
19. CtrlVideocallChat – контроллер управления чатом активной видеоконференции;
20. CtrlVideocalls – контроллер управления активными видеоконференциями;
21. ConverterChatMessageToChatMessage – конвертер DTO сообщения чата в сущность сообщения чата (и наоборот) ;
22. ConverterChatToChat – конвертер DTO чата в сущность чата (и наоборот);
23. ConverterFileToEntityFile – конвертер DTO файла в сущность файла (и наоборот);
24. ConverterUserUpdateToUser – конвертер DTO пользователя в сущность пользователя (и наоборот) ;
25. DTOChat – DTO (Data Transfer Object) чата;
26. DTOChatDisplay – DTO (Data Transfer Object) чата для отображения;
27. DTOChatMessage – DTO (Data Transfer Object) сообщения чата;
28. DTOFile – DTO (Data Transfer Object) файлов;
29. DTOMessageVideocall – DTO (Data Transfer Object) сообщения в чате активной видеоконференции;
30. DTOUserUpdate – DTO (Data Transfer Object) пользователя;
31. DTOVideocallChatMessage – DTO (Data Transfer Object) сообщения в чате активной видеоконференции;
32. DTOVideocallUpdate – DTO (Data Transfer Object) активной видеоконференции;

Интерфейсы репозиториев (для связи с базой данных):

1. RepBanlist – интерфейс репозитория для сущности EntBanlist;
2. RepChat – интерфейс репозитория для сущности EntChat;
3. RepChatFiles – интерфейс репозитория для сущности EntChatFiles;
4. RepChatMessage – интерфейс репозитория для сущности EntChatMessage;
5. RepChatUser – интерфейс репозитория для сущности EntChatUser;
6. RepCompletedtasksfiles – интерфейс репозитория для сущности EntCompletedtasksfiles;
7. RepConferences – интерфейс репозитория для сущности EntConferences;
8. RepGroup – интерфейс репозитория для сущности EntGroup;
9. RepJournal – интерфейс репозитория для сущности EntJournal;
10. RepPasswordRestoreMails – интерфейс репозитория для сущности EntPasswordRestoreMails;
11. RepRole – интерфейс репозитория для сущности EntRole;
12. RepSubject – интерфейс репозитория для сущности EntSubject;
13. RepTasks – интерфейс репозитория для сущности EntTasks;
14. RepTasksCompleted – интерфейс репозитория для сущности EntTasksCompleted;
15. RepTasksFiles – интерфейс репозитория для сущности EntTasksFiles;
16. RepUser – интерфейс репозитория для сущности EntUser;
17. RepUserfiles – интерфейс репозитория для сущности EntUserfiles;
18. RepVideocallChat – интерфейс репозитория для сущности EntVideocallChat;
19. RepVideocalls – интерфейс репозитория для сущности EntVideocalls;
20. RepVideocallsHasUser – интерфейс репозитория для сущности EntVideocallsHasUser;
21. RepVideocallsHasUserProperties – интерфейс репозитория для сущности EntVideocallsHasUserProperties;

Классы сервисов, использующих репозитории и обменивающихся данными с контроллерами:

1. ServiceChat – класс бизнес-логики управления чатами;
2. ServiceChatFiles – класс бизнес-логики управления файлами сообщений чата;
3. ServiceChatMessage – класс бизнес-логики управления сообщениями чата;
4. ServiceChatMessageAsync – класс выполнения асинхронных задач управления сообщениями чата;
5. ServiceCompletedtaskfiles – класс бизнес-логики управления файлами выполненных заданий;
6. ServiceTasks – класс бизнес-логики управления заданиями;
7. ServiceTasksCompleted – класс бизнес-логики управления выполненными заданиями;
8. ServiceTasksfiles – класс бизнес-логики управления файлами заданий;
9. ServiceUser – класс бизнес-логики управления пользователями;
10. ServiceUserData – класс бизнес-логики получения данных о заданиях и конференциях для пользователя;
11. ServiceUserfiles – класс бизнес-логики управления файлами пользователя;
12. ServiceVideocalls – класс бизнес-логики управления активными конференциями;
13. ServiceVideocallsAsync – класс выполнения асинхронных задач управления конференциями;
14. ServiceVideocallsChat – класс бизнес-логики управления чатом активной конференции;
15. ServiceVideocallsChatAsync – класс выполнения асинхронных задача управления чатом активной видеоконференции;
16. ServiceVideocallsJanusAPI – класс связи с API Janus;
17. ServiceBanlist – класс бизнес-логики управления банами;
18. ServiceConference – класс бизнес-логики управления конференциями;
19. ServiceGroup – класс бизнес-логики управления группами;
20. ServiceJournal – класс бизнес-логики управления электронным журналом;
21. ServicePasswordRestoreMails – класс бизнес-логики управления письмами для восстановления пароля;
22. ServiceRole – класс бизнес-логики управления ролями;
23. ServiceSpringSecurity – класс управления авторизацией пользователей;
24. ServiceSubject – класс бизнес-логики управления предметами;
25. ServiceVideocallsHasUser – класс бизнес-логики управления участниками активной конференции;

Классы сущностей (объектов), представляющих таблицы из базы данных:

1. EntBanlist – сущность бана;
2. EntChat – сущность чата;
3. EntChatfiles – сущность файлов сообщения чата;
4. EntChatMessage – сущность сообщения чата;
5. EntChatUser – сущность участника чата;
6. EntCompletedtasksfiles – сущность файлов выполненного задания;
7. EntConferences – сущность конференции;
8. EntGroup – сущность группы;
9. EntJournal – сущность электронного журнала;
10. EntPasswordRestoreMails – сущность письма для восстановления пароля;
11. EntRole – сущность роли;
12. EntSubject – сущность предмета;
13. EntTasks – сущность задания;
14. EntTasksCompleted – сущность выполненного задания;
15. EntTasksfiles – сущность файлов задания;
16. EntUser – сущность пользователя;
17. EntUserfiles – сущность файлов пользователя;
18. EntVideocallChat – сущность чата активной видеоконференции;
19. EntVideocalls – сущность активной видеоконференции;
20. EntVideocallsHasUser – сущность участника видеоконференции;
21. EntVideocallsHasUserProperties – сущность ограничений участника активной видеоконференции;

Классы исключений:

1. AccessException – исключение для индикации ошибки доступа;
2. BaseException – базовый класс исключений, используется для всех описанных исключений;
3. DataProcessingException – исключение для индикации ошибки обработки данных;
4. EntityException – исключение для индикации ошибки, связанной с обработкой сущности;
5. JanusAPIException – исключение для индикации ошибки, связанной с Janus;

Конфигурационные классы:

1. ConfApp – бин общей конфигурации приложения;
2. ConfPropsCookies – класс, хранящий константы из конфигурационного файла, связанные с cookie-файлами;
3. ConfPropsPath – класс, хранящий константы из конфигурационного файла, связанные с путями;
4. ConfSpringAsync – конфигурация асинхронных механизмов;
5. ConfSpringRetry – конфигурация повторения операций с базой данных при блокировках записей;
6. ConfSpringSecurity – конфигурация системы безопасности;
7. ConfUrls – класс, хранящий константы из конфигурационного файла, связанные с URL.
8. ConfWebSocket – конфигурация WebSocket;

Утилитарные и прочие классы:

1. CookieFilter – конфигурация cookie-файлов;
2. Checker – класс для проверок условий;
3. FilesMGMT – класс для сохранения файлов;
4. Pair – класс Пара;
5. Parser – класс для получения или формирования данных из структуры данных;
6. WebSocketJanus – WebSocket для связи с Janus;
7. WebSocketVideocall – WebSocket для конференций и чата;
8. WebSocketManager – класс, хранящий все активные конференции;
9. GenPasswordCrypt – класс для хэширования паролей;
10. DiplomTestApplication – класс запуска приложения.
    1. **Инфологическое проектирование**

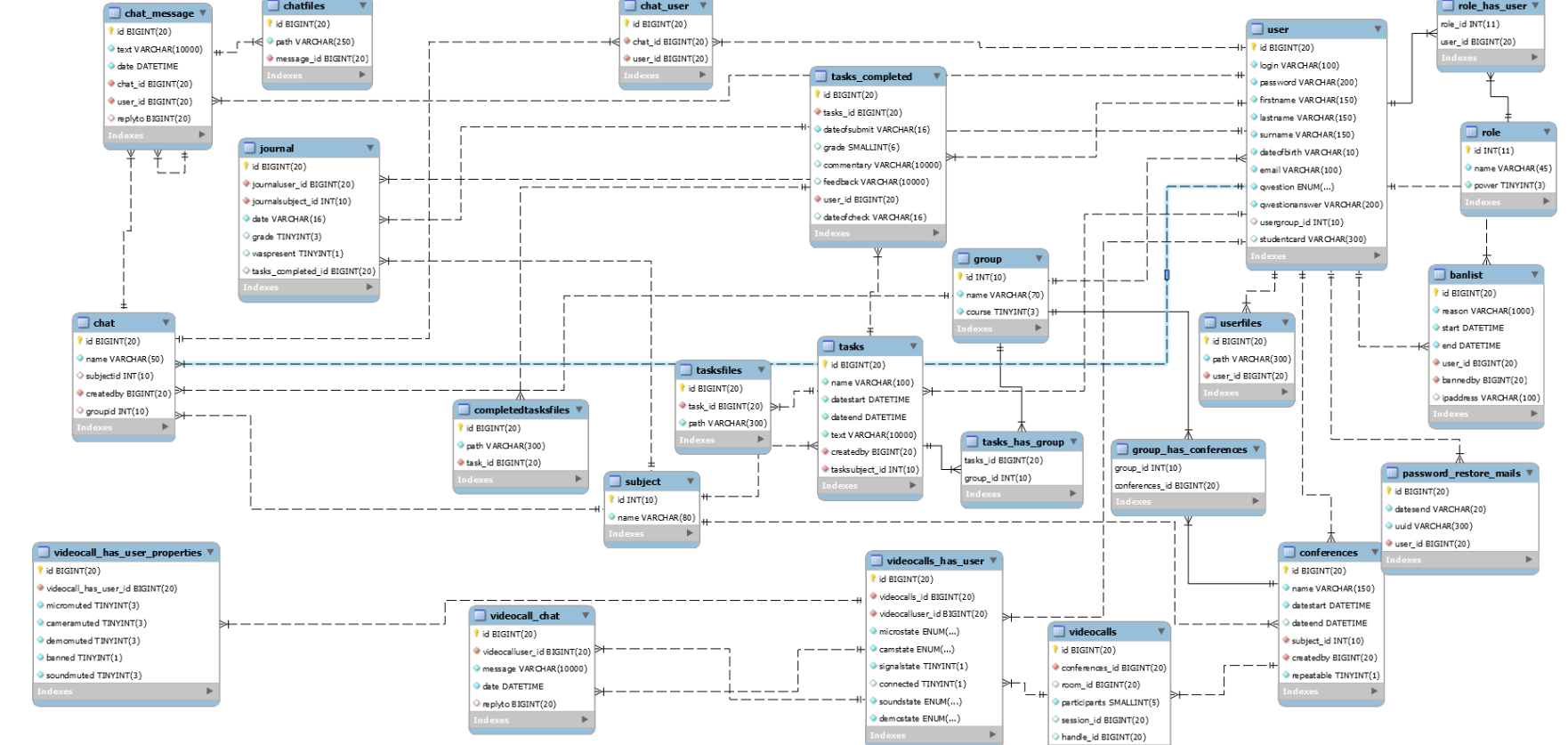
Инфологическую модель базы данных видно на рисунке 48:

Рисунок 48 – Инфологическая модель базы данных

Таблицы:

* 1. subject

Предназначена для хранения учебных дисциплин.

Поля:

* id (INT PK)– идентификатор
* name (VARCHAR) – название дисциплины
  1. journal

Предназначена для хранения записей журнала успеваемости.

Поля:

* id (BIGINT PK) – идентификатор
* journaluser\_id (BIGINT FK) – идентификатор пользователя, создавшего запись
* journalsubject\_id (INT FK) – идентификатор предмета, к которому относится запись
* date (VARCHAR) – дата создания записи
* grade (INT) – оценка
* waspresent (BOOLEAN) – присутствие на занятии
* tasks\_completed\_id (BIGINT FK) – идентификатор задания, к которому относится запись
  1. group

Предназначена для хранения групп.

Поля:

* id (INT PK) – идентификатор
* name (VARCHAR) – название группы
* course (TINYINT) – курс
  1. tasks\_has\_group

Предназначена для хранения списка заданий для каждой группы.

Поля:

* tasks\_id (BIGINT FK) – идентификатор задания
* group\_id (INT FK) – идентификатор группы
  1. user

Предназначена для хранения профилей пользователей.

Поля:

* id (BIGINT PK) – идентификатор
* login (VARCHAR) – логин
* password (VARCHAR) – пароль
* firstname (VARCHAR) – имя
* lastname (VARCHAR) – фамилия
* surname (VARCHAR) – отчество
* dateofbirth (VARCHAR) – дата рождения
* email (VARCHAR) – адрес электронной почты
* qwestion (ENUM {‘PETNAME’,’MOTHERLASTNAME’,’RESERVEPW’}) – тип проверочного вопроса
* qwestionanswer (VARCHAR) – ответ на проверочный вопрос
* usergroup\_id (INT FK) – идентификатор группы
* studentcard (VARCHAR) – номер студенческого билета
  1. userfiles

Предназначена для хранения файлов пользователя.

Поля:

* id (BIGINT PK) – идентификатор
* path (VARCHAR) – путь к файлу
* user\_id (BIGINT FK) – идентификатор пользователя
  1. role

Предназначена для хранения ролей.

Поля:

* id (INT) – идентификатор
* name (VARCHAR) – название роли
* power (TINYINT)
  1. role\_has\_user

Предназначена для хранения ролей пользователей.

Поля:

* role\_id (INT FK) – идентификатор роли
* user\_id (BIGINT FK) – идентификатор пользователя
  1. password\_restore\_mails

Предназначена для хранения писем для восстановления пароля.

Поля:

* id (INT PK) – идентификатор
* datesend (VARCHAR) – дата и время отправки
* uuid (VARCHAR) – уникальный идентификатор письма
* user\_id (BIGINT) – идентификатор пользователя
  1. chat

Предназначена для хранения списка чатов.

Поля:

* id (INT PK) – идентификатор
* name (VARCHAR) – название канала
* createdby (BIGINT FK) – идентификатор пользователя, создавшего чат
* group\_id (INT FK) – идентификатор группы, для которой создан чат
* subjectid (INT FK) – идентификатор предмета, которому посвящён чат
  1. chat\_user

Предназначена для связи таблицы пользователей с таблицей чата.

Поля:

* id (BIGINT PK) – идентификатор
* chat\_id (INT FK) – идентификатор чата
* user\_id (BIGINT FK) – идентификатор пользователя
  1. tasks

Предназначена для хранения списка заданий.

Поля:

* id (BIGINT PK) – идентификатор
* name (VARCHAR) – название
* datestart (VARCHAR) – дата размещения
* dateend (VARCHAR) – срок сдачи
* text (VARCHAR) – текстовое описание
* createdby (BIGINT FK) – идентификатор пользователя, создавшего задание
* tasksubject\_id (INT FK) – идентификатор предмета, к которому относится задание
  1. tasksfiles

Предназначена для хранения файлов, связанных с заданиями.

Поля:

* id (BIGINT PK) – идентификатор
* task\_id (BIGINT FK) – идентификатор задания
* path (VARCHAR) – путь к файлу
  1. tasks\_completed

Предназначена для хранения сданных студентами заданий.

Поля:

* id (BIGINT PK) – идентификатор
* tasks\_id (BIGINT FK) – идентификатор оригинального задания
* dateofsubmit (VARCHAR) – дата сдачи
* grade (SMALLINT) – оценка
* commentary (VARCHAR) – комментарий (текстовый ответ) студента
* feedback (VARCHAR) – комментарий преподавателя
* user\_id (BIGINT) – идентификатор пользователя, сдавшего задание
* dateofcheck (VARCHAR) – дата проверки задания
  1. completedtasksfiles

Предназначена для хранения файлов, связанных с выполненными заданиями.

Поля:

* id (BIGINT PK) – идентификатор
* path (VARCHAR) – путь к файлу
* task\_id (BIGINT FK) – идентификатор выполненного задания
  1. conferences

Предназначена для хранения назначенных видеоконференций.

Поля:

* id (BIGINT PK) – идентификатор
* name (VARCHAR) – название
* datestart (VARCHAR) – дата проведения
* dateend (VARCHAR) – дата окончания
* group\_id (INT FK) – идентификатор группы, участвующей в конференции
* subject\_id (INT FK) – идентификатор предмета, которому посвящена видеоконференция
* createdby (BIGINT FK) – идентификатор пользователя, создавшего видеоконференцию
* repeatable (TINYINT) – периодичность проведения в днях
  1. videocalls

Предназначена для хранения активных видеоконференций.

Поля:

* id (BIGINT PK) – идентификатор
* conferences\_id (BIGINT FK) – идентификатор конференции
* room\_id (BIGINT) – идентификатор комнаты SFU
* participants (SMALLINT) – количество подключенных участников
* session\_id (BIGINT) – id сессии SFU
* handle\_id (BIGINT) – id handle SFU
  1. videocalls\_has\_user

Предназначена для хранения списка пользователей, участвующих в видеоконференции.

Поля:

* id (BIGINT PK) - идентификатор
* videocalls\_id (BIGINT FK) – идентификатор активной видеоконференции
* user\_id (BIGINT FK) – идентификатор пользователя
* microstate (ENUM {‘OFF’,’ON’,’MUTED\_BY\_ADMIN’}) – состояние микрофона пользователя
* camstate (ENUM {‘OFF’,’ON’,’MUTED\_BY\_ADMIN’}) – состояние веб-камеры пользователя
* soundstate (ENUM {‘OFF’,’ON’,’MUTED\_BY\_ADMIN’}) – состояние звука пользователя
* demostate (ENUM {‘OFF’,’ON’,’MUTED\_BY\_ADMIN’}) – состояние демонстрации пользователя
* connected (BOOLEAN) – подключён ли участник
  1. videocall\_chat

Предназначена для хранения истории сообщений чата видеоконференции.

Поля:

* id (BIGINT PK) – идентификатор
* text (VARCHAR) – текст сообщения
* date (VARCHAR) – дата отправки
* videocalluser\_id (BIGINT FK) – идентификатор пользователя, отправившего сообщение
* replyto (BIGINT FK) – идентификатор участника, упомянутого в сообщении
  1. videocalls\_has\_user\_properties

Предназначена для хранения ограничений участника конференции

Поля:

* id (BIGINT PK) – идентификатор
* videocall\_has\_user\_id (BIGINT FK) – идентификатор участника конференции
* micromuted (TINYINT) – сила роли участника, запретившего микрофон
* cameramuted (TINYINT) – сила роли участника, запретившего камеру
* demomuted (TINYINT) – сила роли участника, запретившего демонстрацию
* soundmuted (TINYINT) – сила роли участника, запретившего звук
* banned (BOOLEAN) – забанен ли участник
  1. banlist

Предназначена для хранения банов.

Поля:

* id (BIGINT PK) – идентификатор
* reason (VARCHAR) – причина бана
* start (DATETIME) – дата бана
* end (DATETIME) – дата окончания бана
* user\_id (BIGINT FK) – идентификатор забанненого пользователя
* bannedby (BIGINT FK) – идентификатор пользователя, выдавшего бан
* ipaddress (VARCHAR) – ip-адрес
  1. chat\_message

Предназначена для хранения сообщений чата.

Поля:

* id (BIGINT PK) – идентификатор
* text (VARCHAR) – текст сообщения
* date (DATETIME) – дата отправки
* chat\_id (BIGINT FK) – идентификатор чата
* user\_id (BIGINT FK) – идентификатор пользователя, отправившего сообщение
* replyto (BIGINT FK) – идентификатор сообщения, на которое отвечают
  1. chatfiles

Предназначена для хранения файлов сообщений чата.

Поля:

* id (BIGNT PK) – идентификатор
* path (VARCHAR) – путь к файлу
* message\_id (BIGINT FK) – идентификатор сообщения
  1. **Проектирование интерфейса**

Эскизы страниц системы.

«Вход» – страница системы, показанная на рисунке 49:

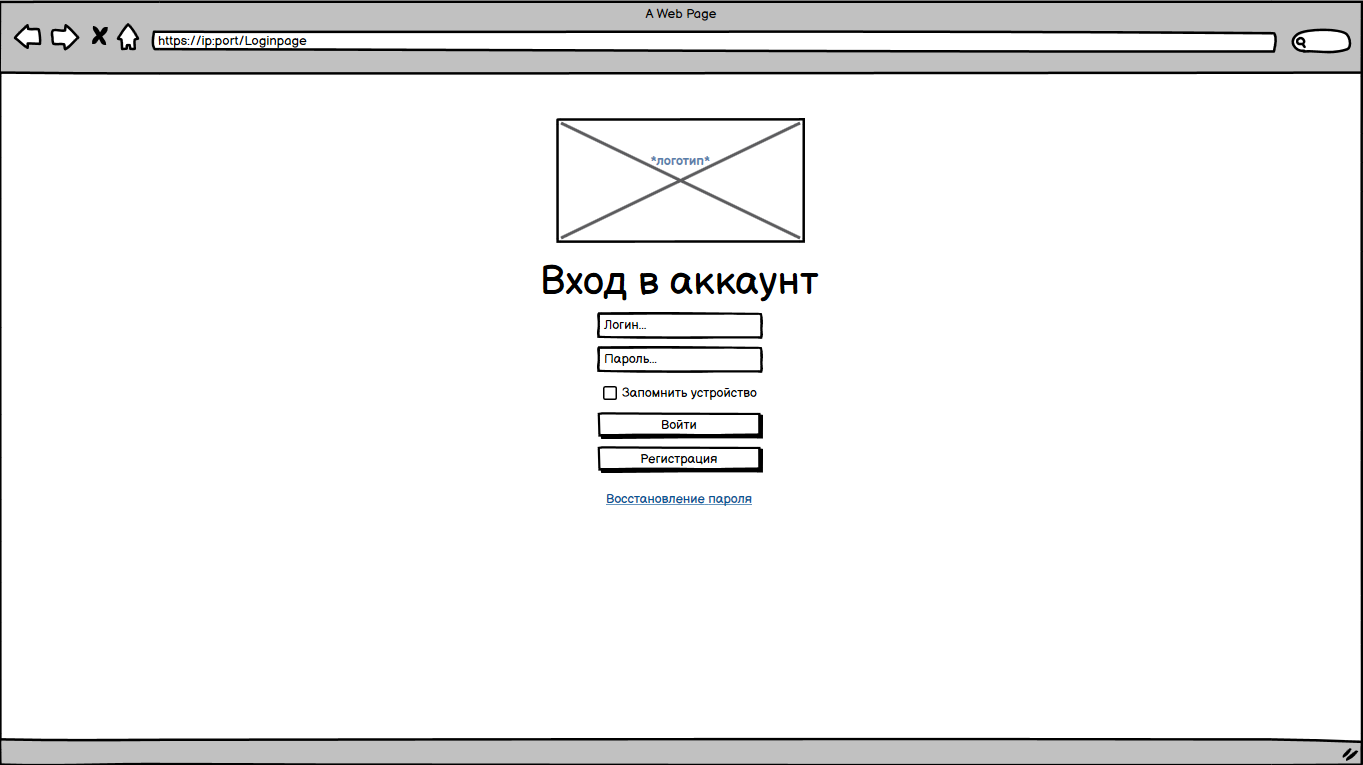


Рисунок 49 – «Вход» – эскиз страницы входа в систему

Страницу регистрации в информационной системе можно увидеть на рисунке 50:

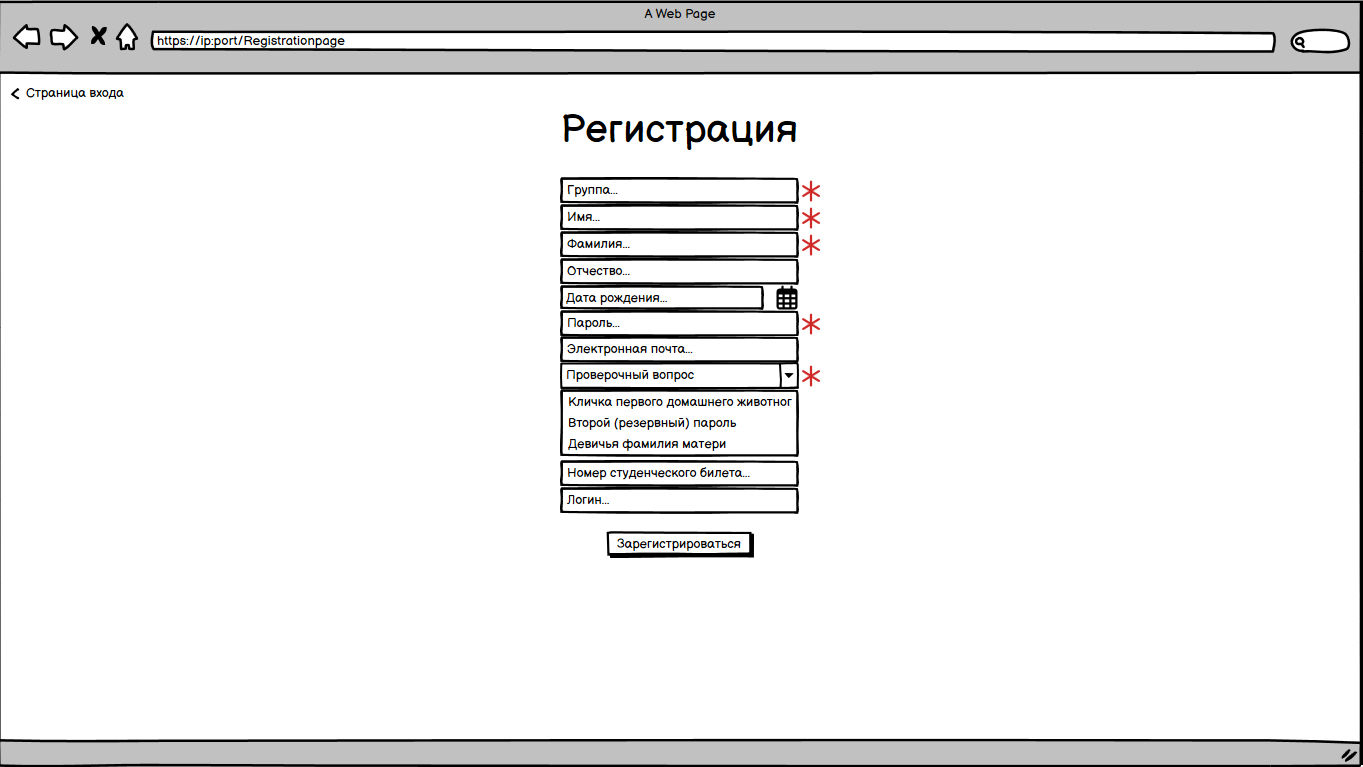


Рисунок 50 – Эскиз страницы регистрации в информационной системе

«Восстановление пароля» – страница ИС, представлена на рисунке 51:

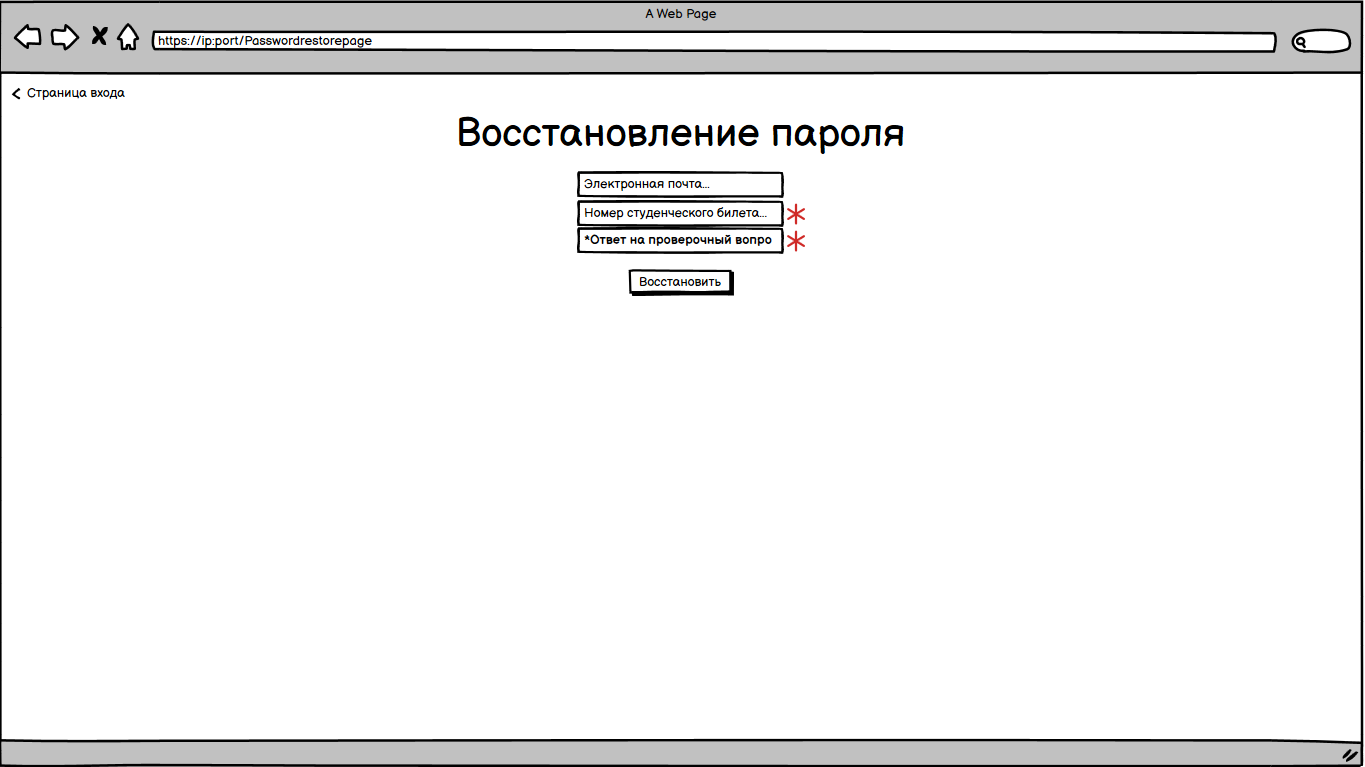


Рисунок 51 – «Восстановление пароля» – эскиз страницы системы

Главная страница системы изображена на рисунке 52:

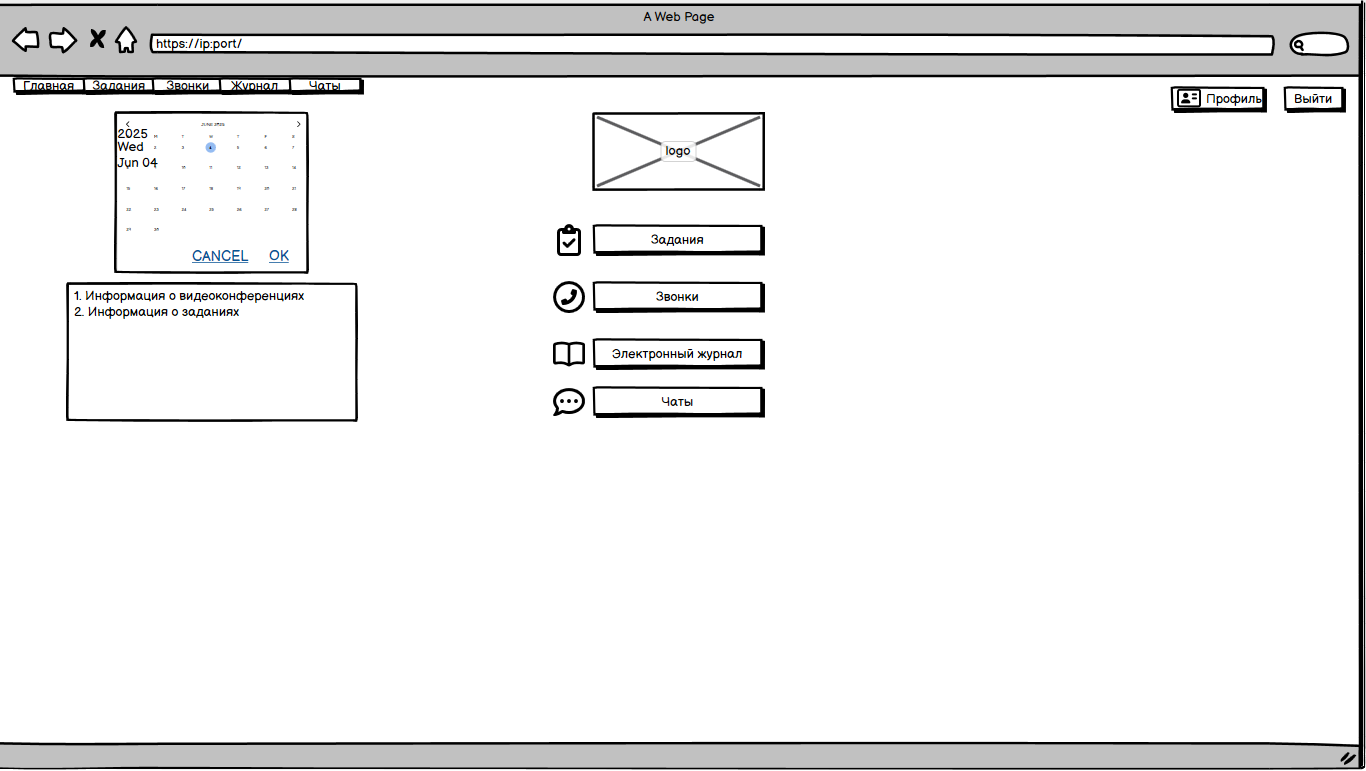


Рисунок 52 – Эскиз главной страницы системы

Страница профиля представлена на рисунке 53:

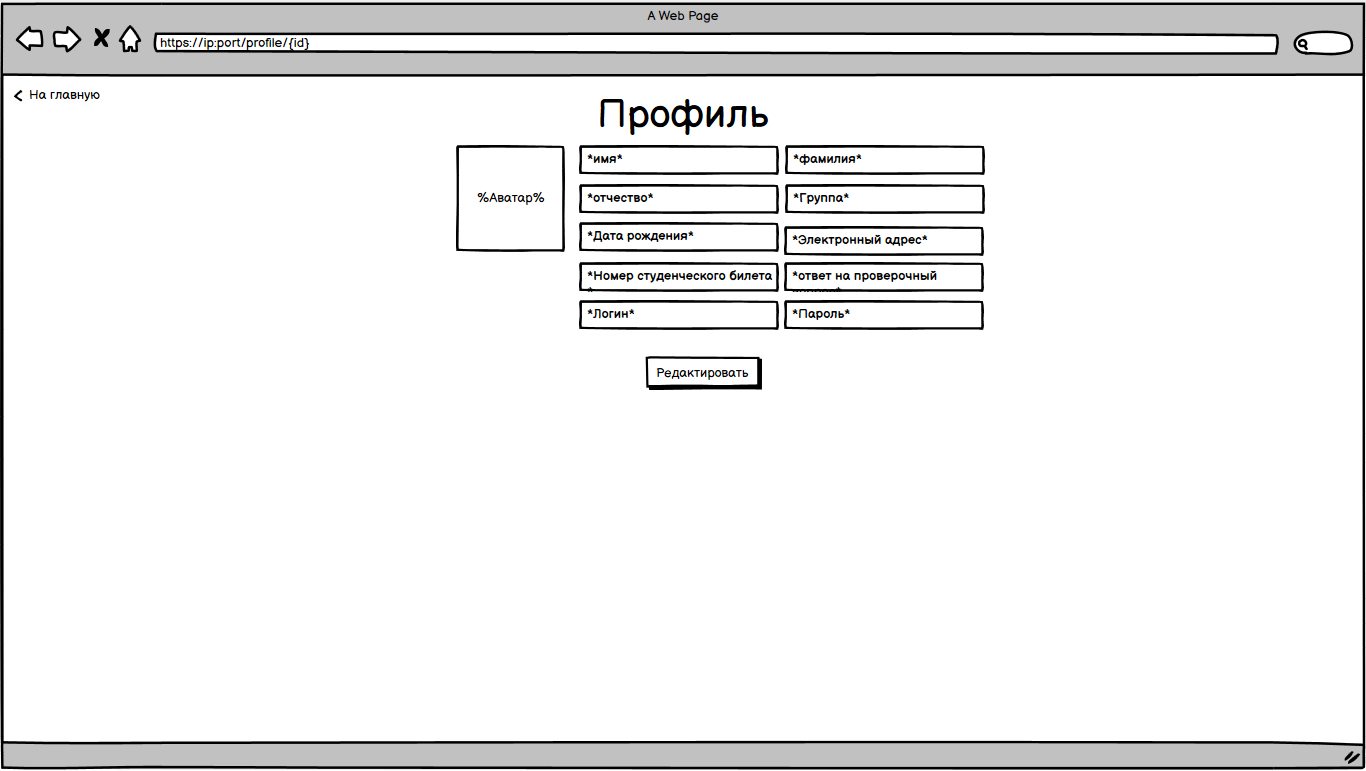


Рисунок 53 – Эскиз страницы профиля

Страница обновления профиля показана на рисунке 54:

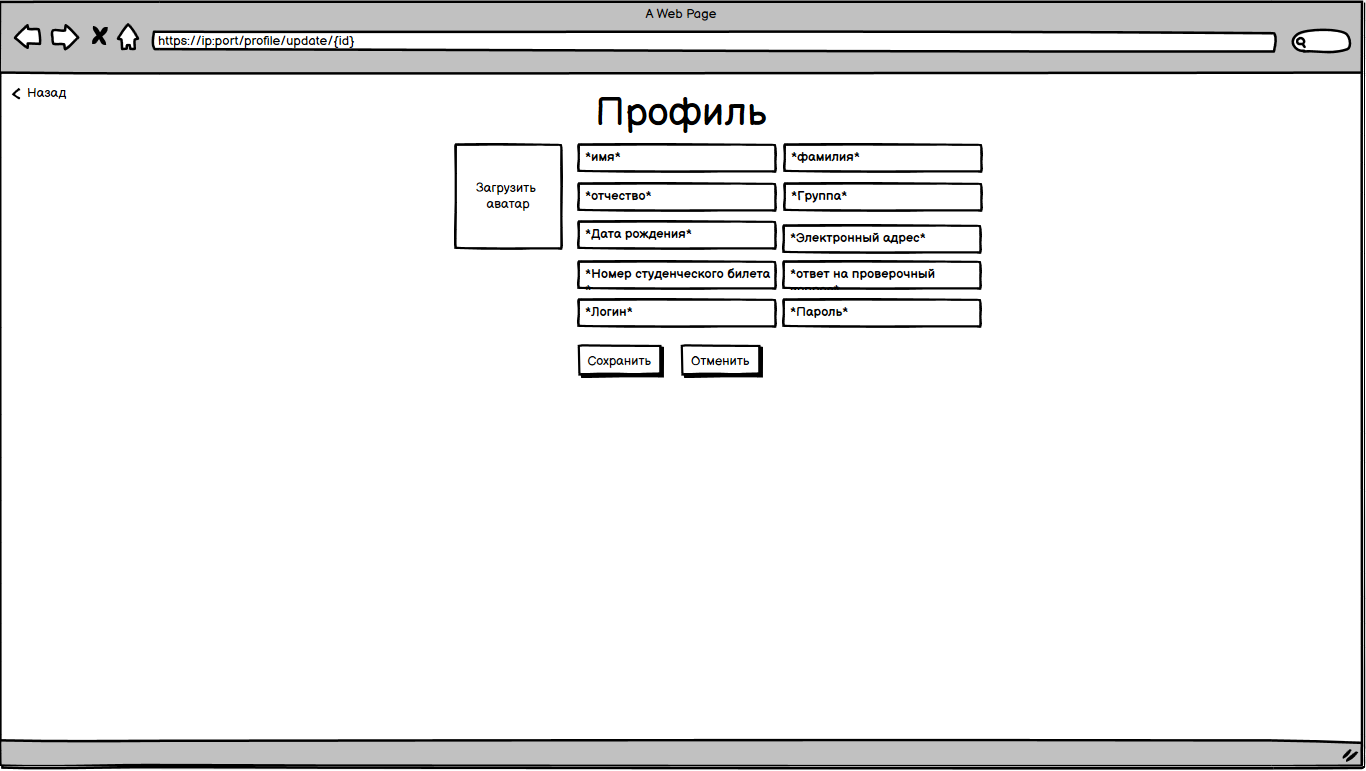


Рисунок 54 – Эскиз страницы обновления профиля

Страницу с заданиями видно на рисунке 55:

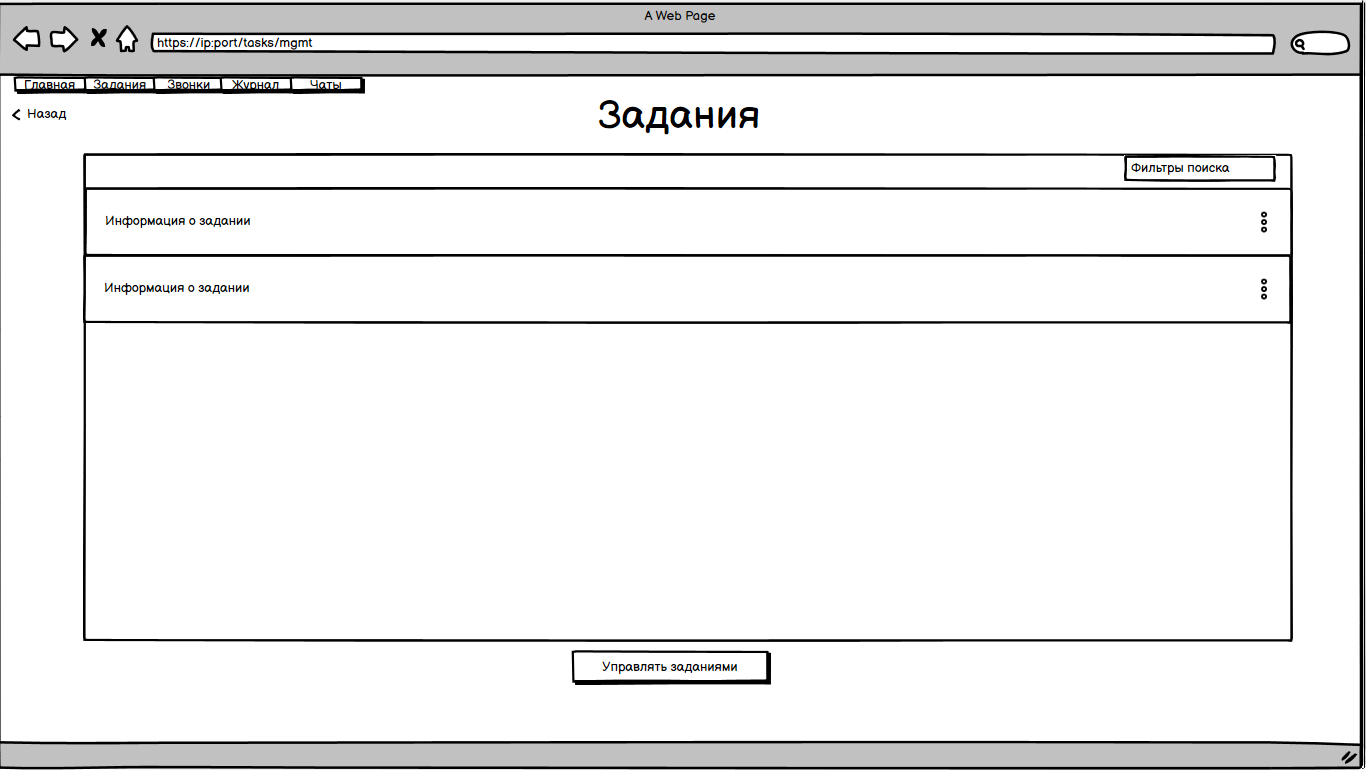


Рисунок 55 – Эскиз страницы заданий

Страницу создания задания видно на рисунке 56:

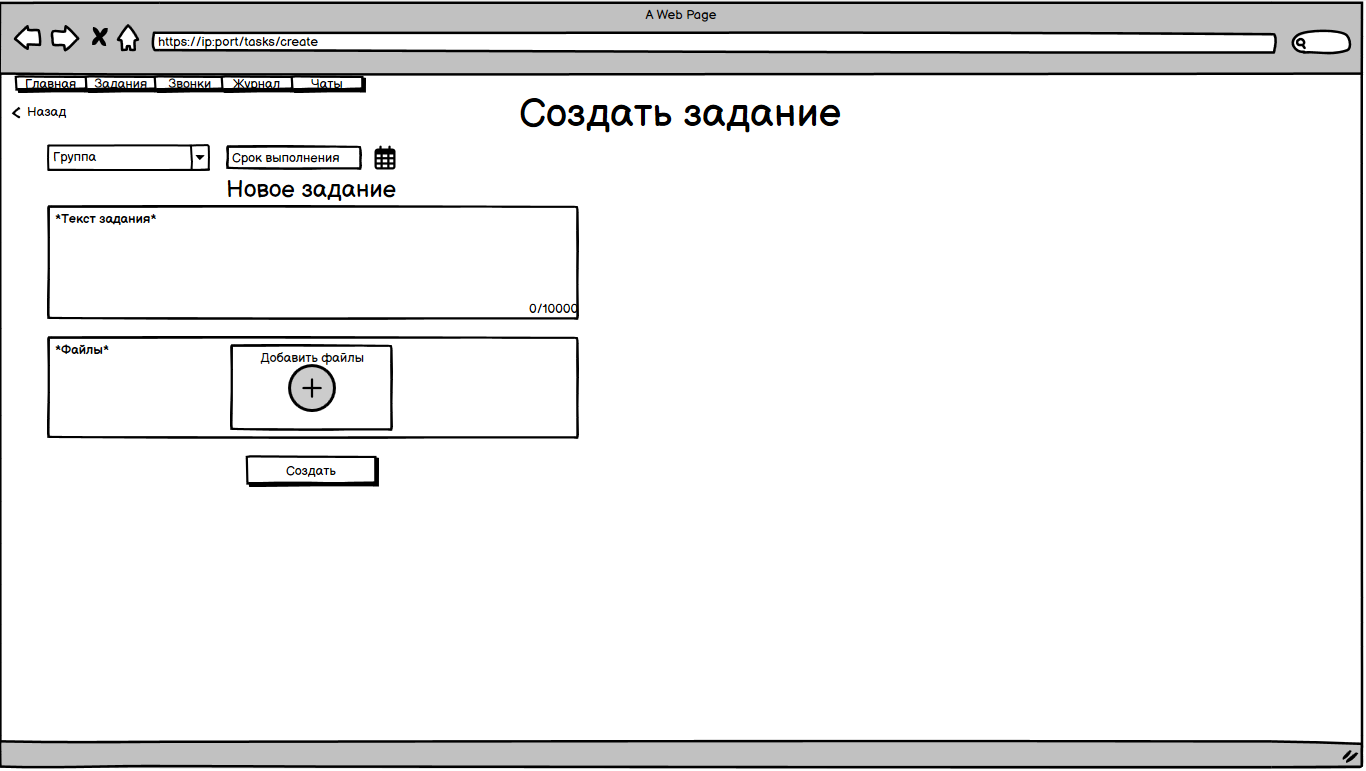


Рисунок 56 – Эскиз страницы создания задания

Страница проверки задания изображена на рисунке 57:

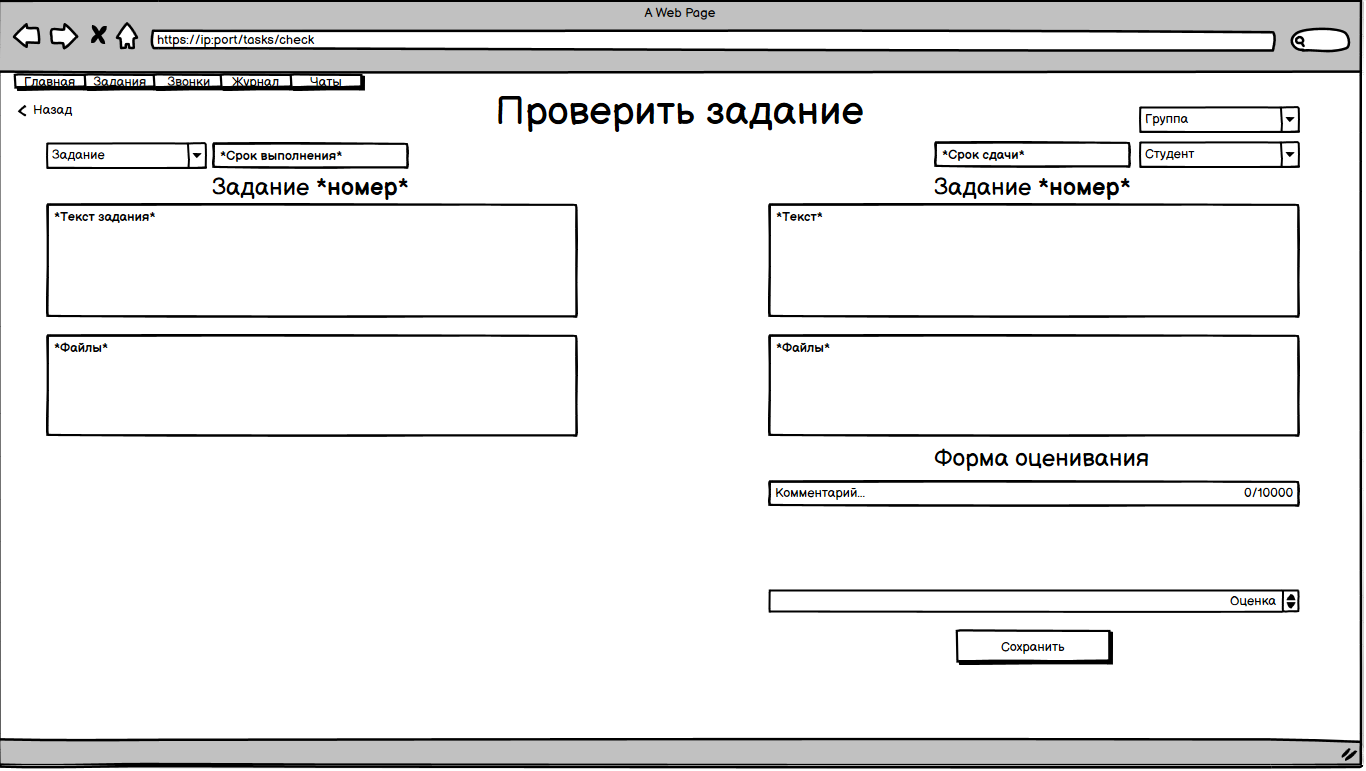


Рисунок 57 – Эскиз страницы проверки заданий

Страницу сдачи задания видно на рисунке 58:

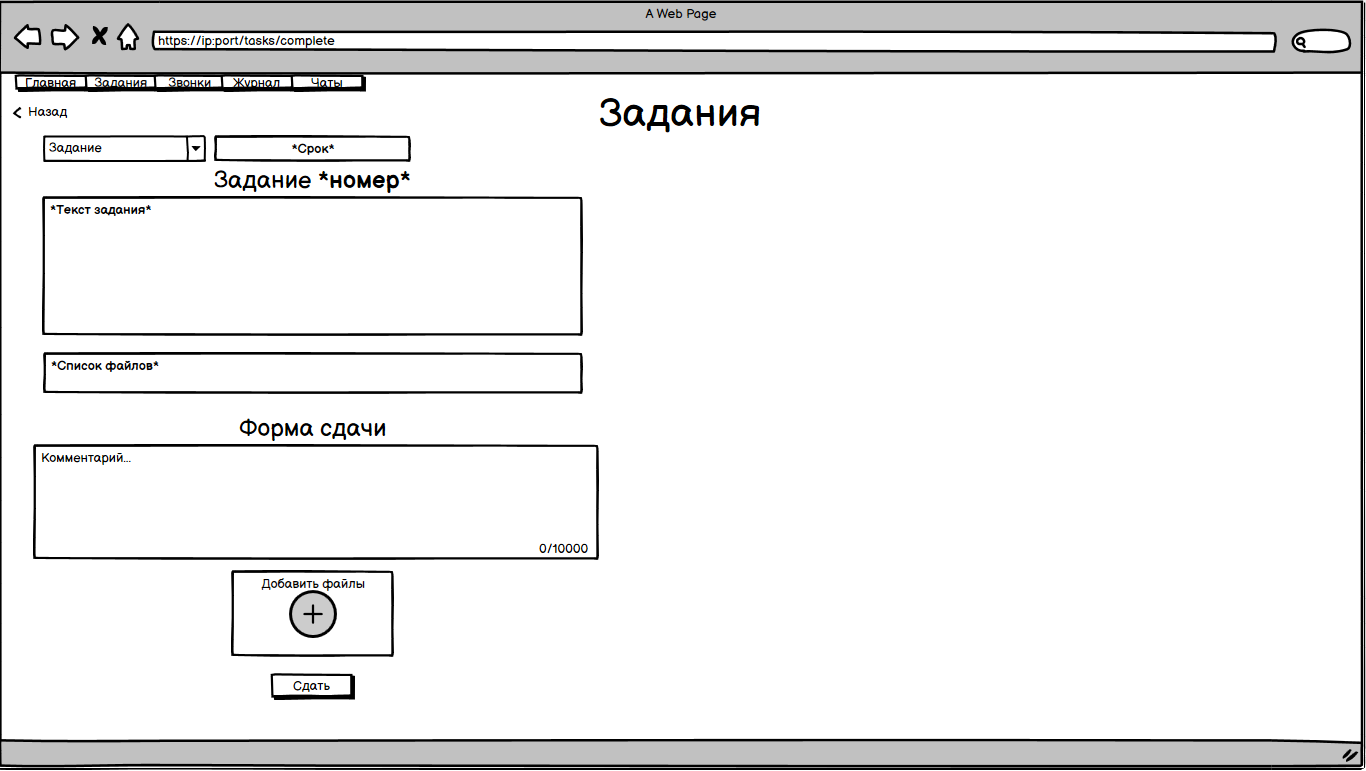


Рисунок 58 – Эскиз страницы сдачи задания

Страница обновления задания изображена на рисунке 59:

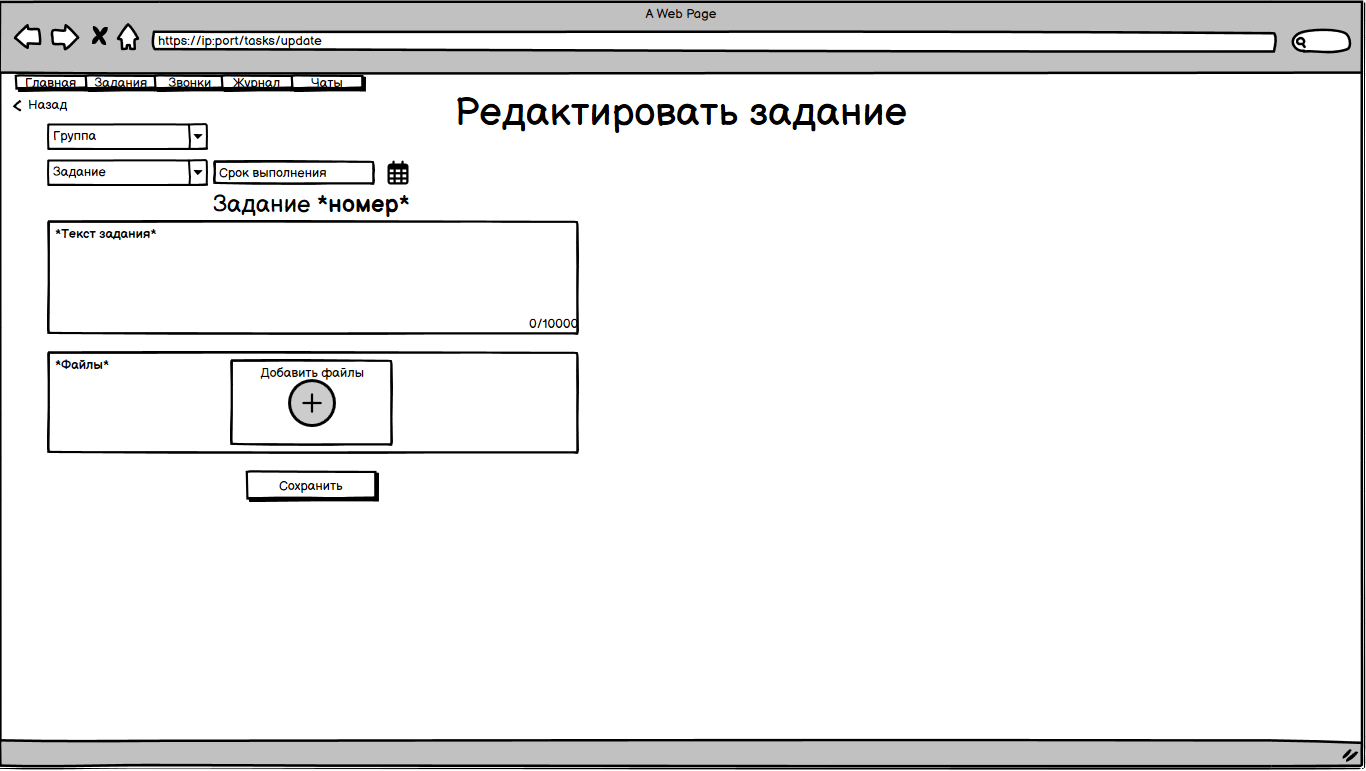


Рисунок 59 – Эскиз страницы обновления задания

Страница журнала (вид для студентов) представлена на рисунке 60:

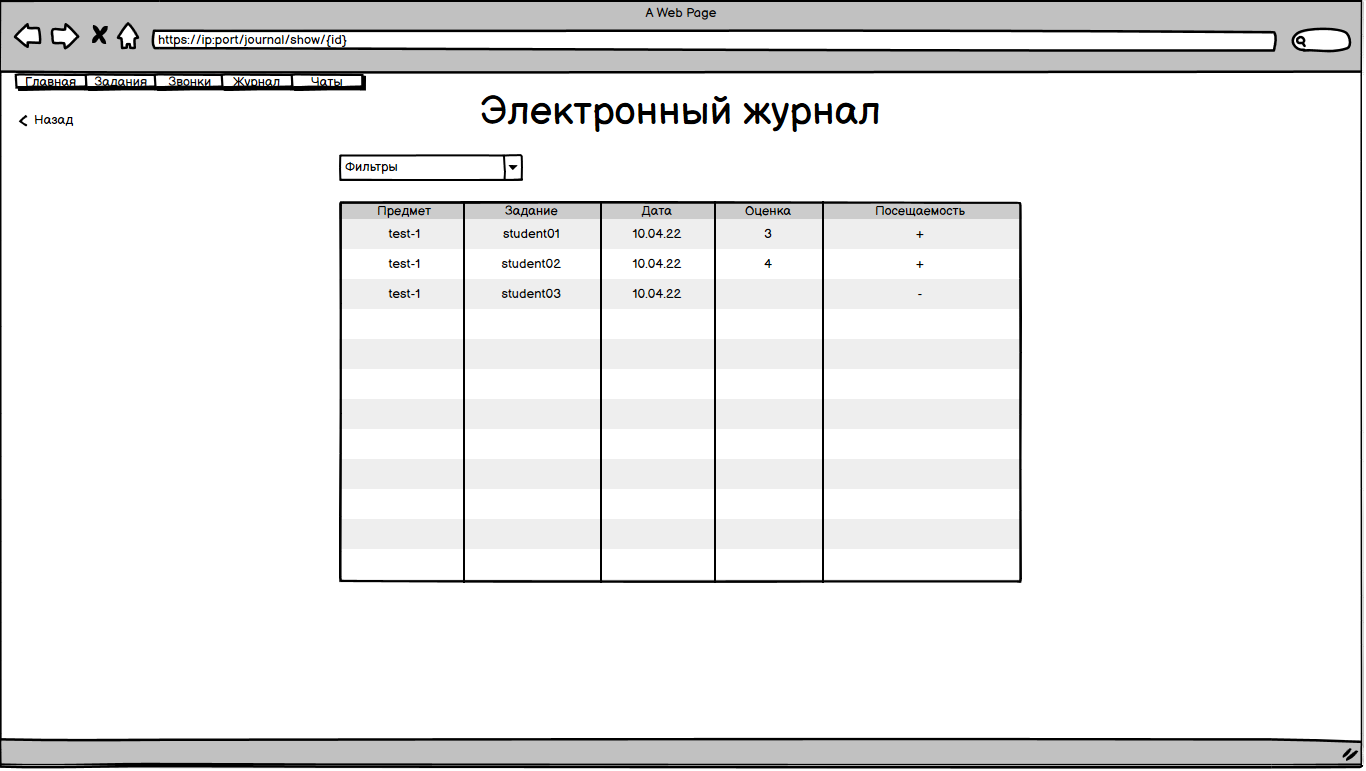


Рисунок 60 – Эскиз страницы журнала (роль участника системы – «студент»)

Страница журнала (вид для преподавателей) представлена на рисунке 61:

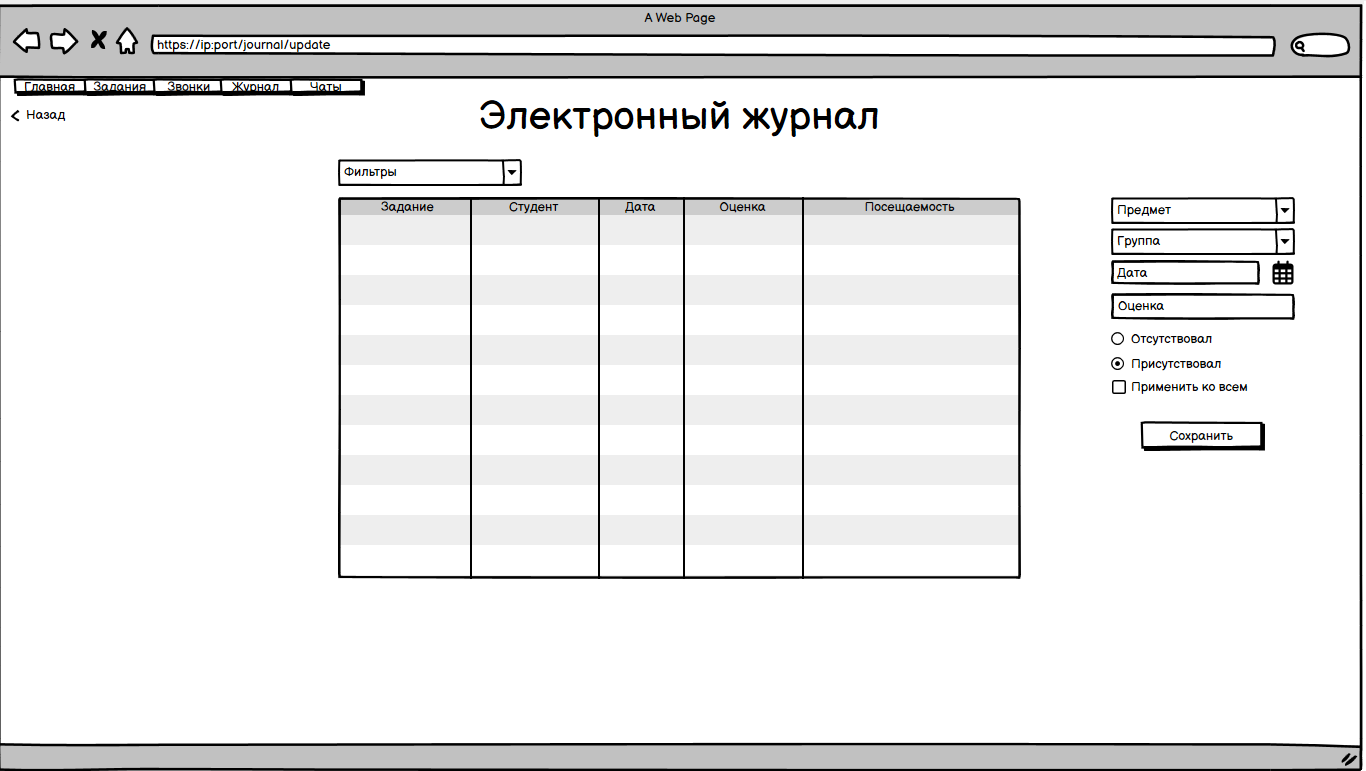


Рисунок 61 – Эскиз страницы журнала (роль участника системы – «преподаватель»)

Страницу видеоконференций можно увидеть на рисунке 62:

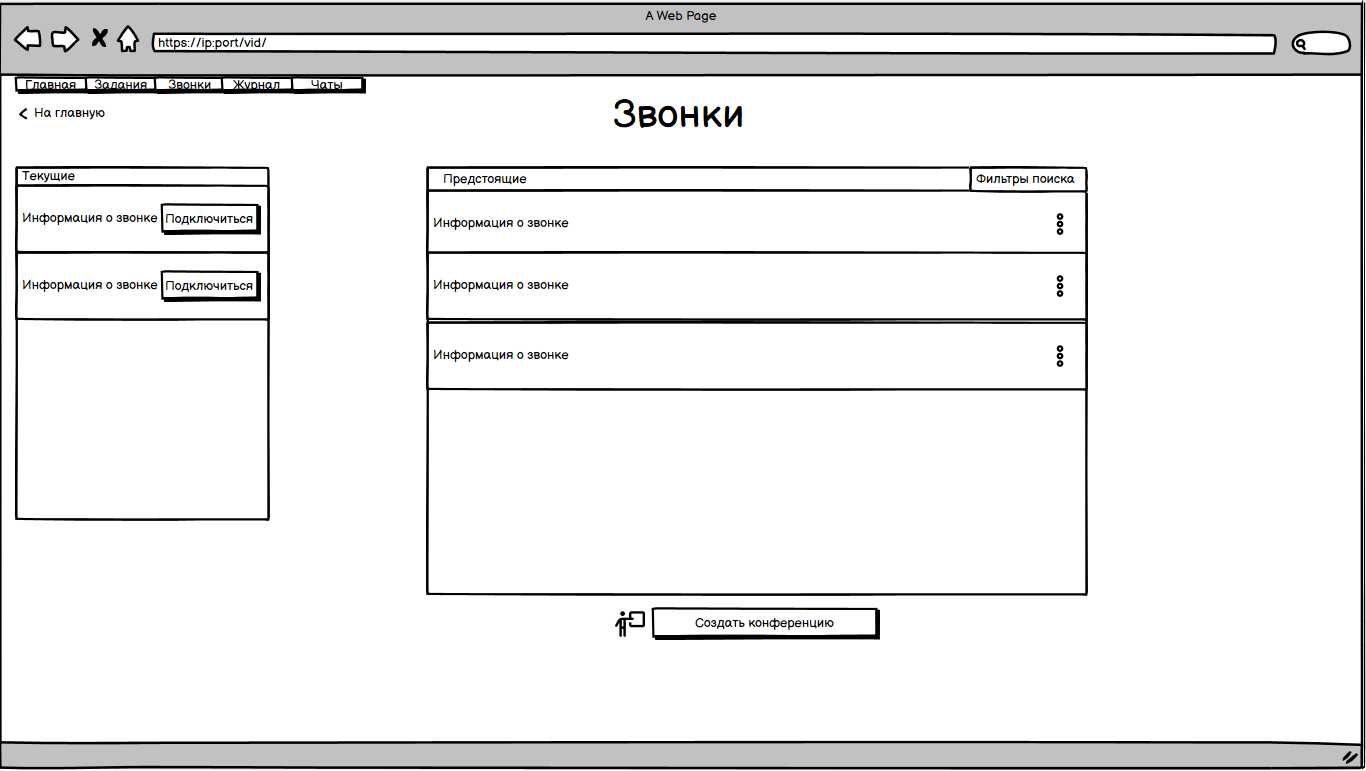


Рисунок 62 – Эскиз страницы видеоконференций

Страница создания видеоконференции изображена на рисунке 63:

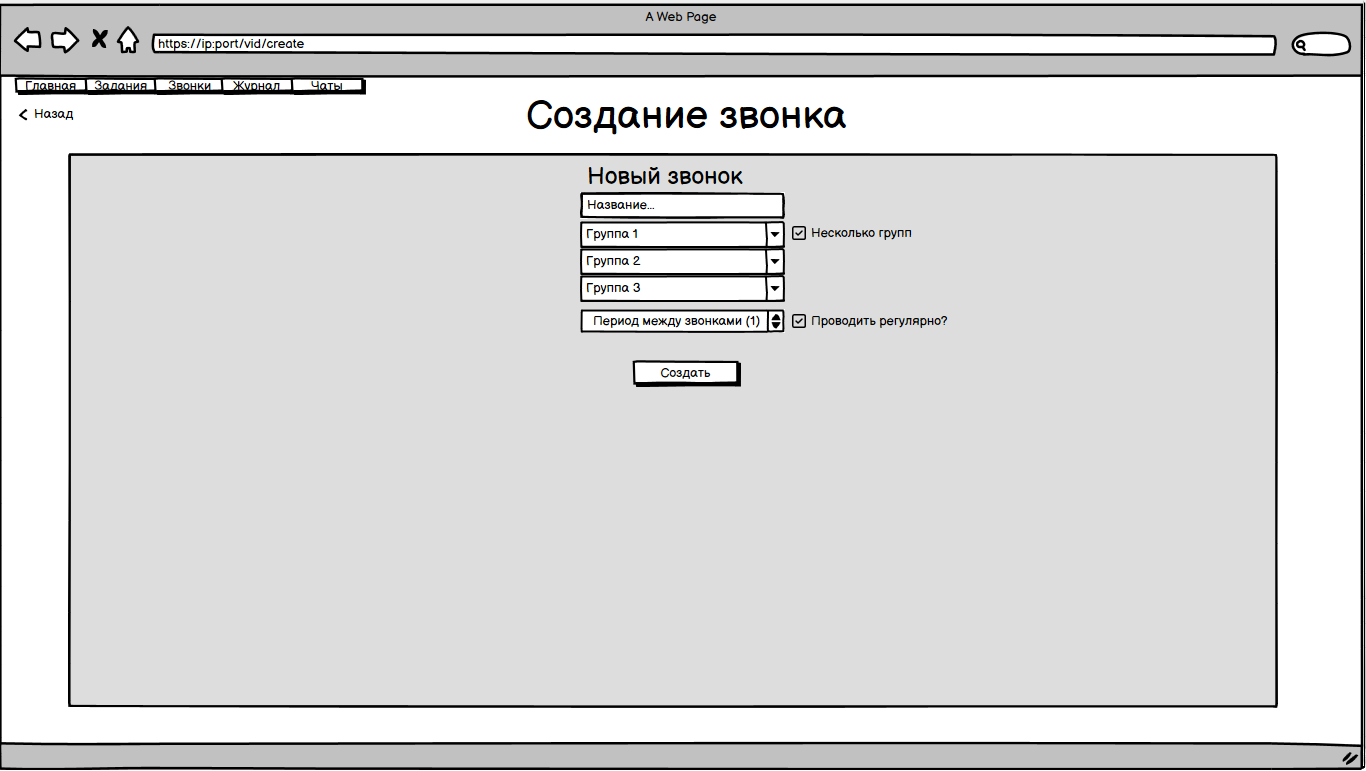


Рисунок 63 – Эскиз страницы создания видеоконференции

Страница обновления видеоконференции представлена на рисунке 64:

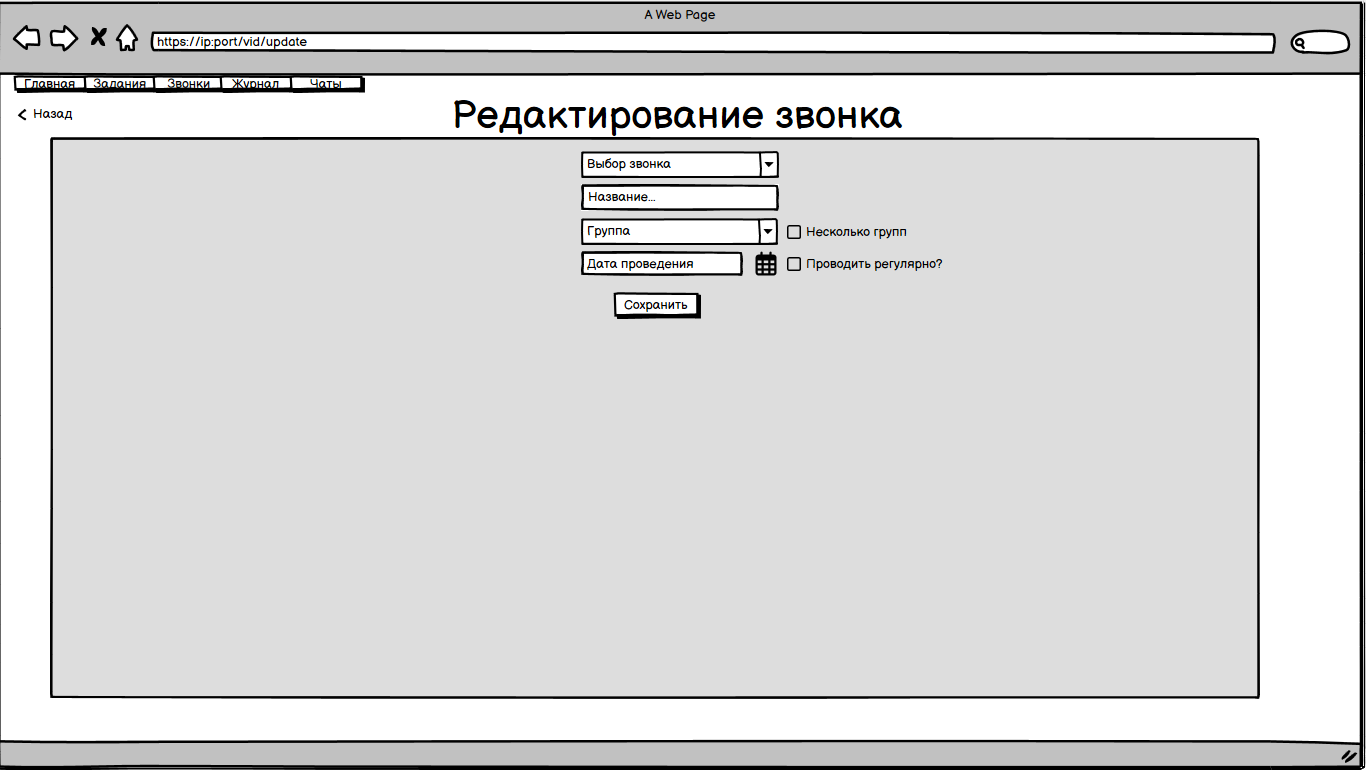


Рисунок 64 – Эскиз страницы обновления видеоконференции

Страница видеозвонка представлена на рисунке 65:

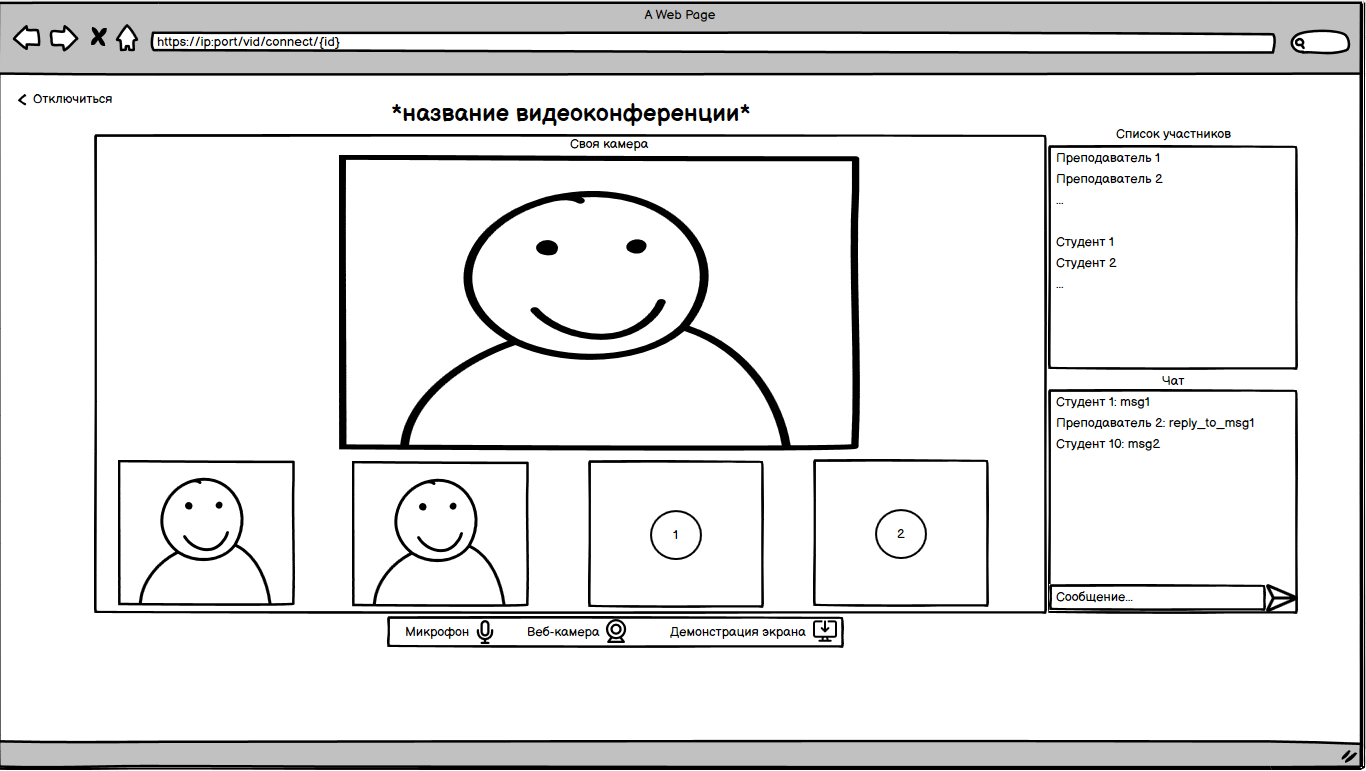


Рисунок 65 – Эскиз страницы видеозвонка

Страница чатов представлена на рисунке 66:

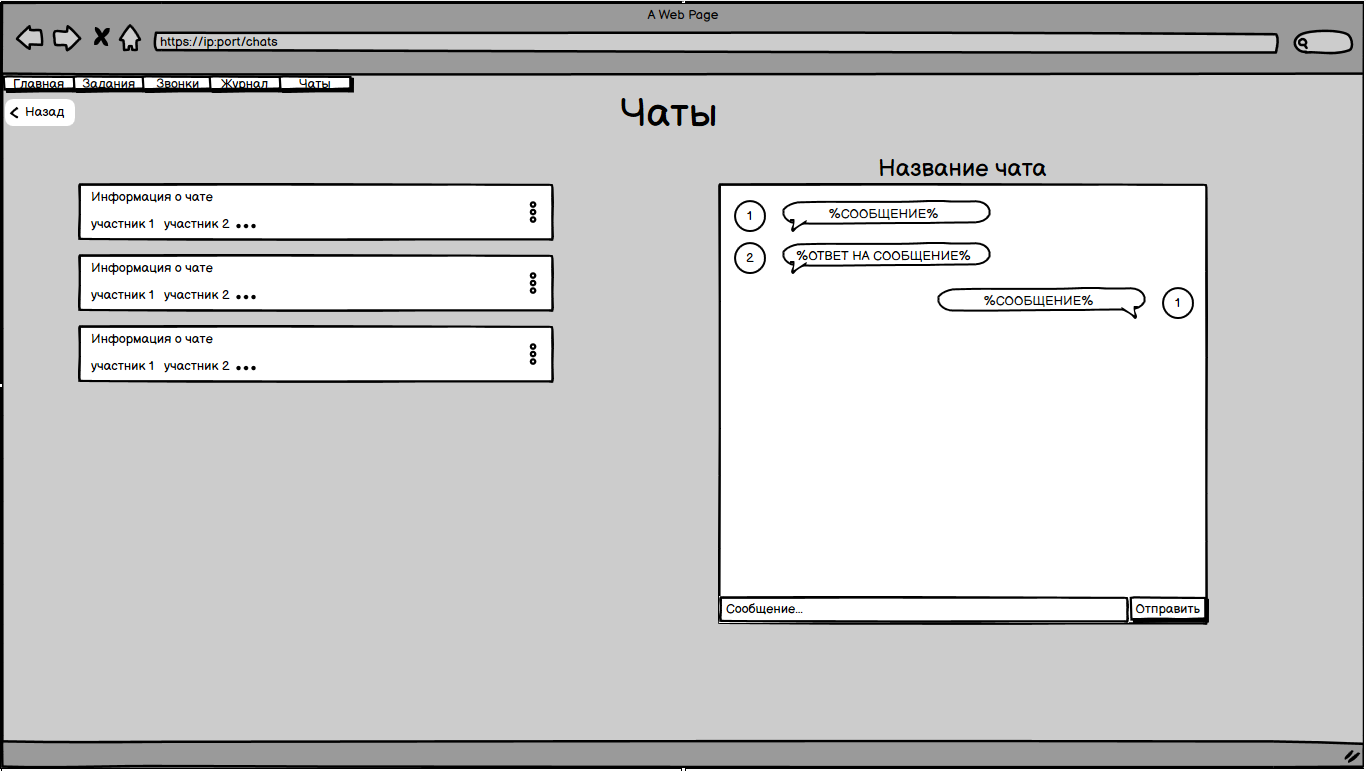


Рисунок 66 – Эскиз страницы чатов

* 1. **Программная реализация проекта**

Технологии для реализации системы были выбраны на основе требований к ней. Стек технологий своевременно обновляется, тем самым обеспечивая повышенную безопасность и оснащаясь новыми опциями, которые, расширяют область применения технологий, а также функциональность.

1. HTML5

HTML5 — это актуальная версия языка разметки, применяемого для формирования структуры веб-документов. Он определяет расположение элементов интерфейса, таких как заголовки, изображения, формы и медиаконтент. HTML5 расширяет возможности предыдущих версий, предоставляя встроенную поддержку видео, аудио и семантической разметки без необходимости использования сторонних плагинов.

1. CSS3

CSS3 отвечает за оформление веб-страниц и визуальное поведение элементов интерфейса. С помощью CSS3 реализованы адаптивные макеты, анимации, тени, переходы и другие графические эффекты. Это позволяет добиться современного и единообразного дизайна без вмешательства в HTML-код [13].

1. JavaScript

JavaScript — скриптовый язык, который делает веб-интерфейс динамичным. С его помощью реализуются интерактивные элементы, реагирующие на действия пользователя, такие как клики, ввод текста и другие события. Также язык применяется для работы с данными, отправки запросов на сервер, обновления DOM-структуры страницы без перезагрузки и прочего. [12, 13]

1. Java

Java – язык программирования, который часто используется для создания серверной части веб-приложений. Java обеспечивает логику работы приложения, обработку данных, взаимодействие с базами данных. Используется на серверной стороне для обработки логики приложения, работы с данными и взаимодействия с клиентом. Для разработки применяются [6, 7]:

* Spring Boot — инструмент, ускоряющий создание и настройку Java-приложений с минимальной конфигурацией [11].
* Spring Security — компонент для настройки защиты приложения, включая проверку подлинности и контроль доступа. Используется для обеспечения безопасности [11].
* Spring Scheduler — модуль, позволяющий запускать задачи по расписанию или через заданные интервалы времени [11].
* Spring Mailing Service — сервис для отправки email-сообщений из приложения, включая уведомления, подтверждения и прочее [11].
* Jakarta (ранее Java EE) — набор стандартов и API для построения масштабируемых серверных решений [10].
* Hibernate — технология, преобразующая объекты Java в таблицы базы данных, упрощая взаимодействие с СУБД [10].
* Thymeleaf — шаблонизатор для генерации HTML на серверной стороне, интегрированный с Spring [11].

1. Maven

Apache Maven представляет собой систему управления проектами, в которой задаются зависимости, плагины и фазы сборки. Благодаря POM-файлу автоматизируется компиляция, а также тестирование, упаковка и развертывание проекта.

1. MySQL

MySQL — это СУБД, основанная на модели клиент-сервер. Она обеспечивает хранение и быстрый доступ к данным, а также поддерживает SQL-запросы, транзакции и масштабирование. Используется для хранения данных и обмена ими с приложением. [9, 14]

1. Apache Tomcat

Tomcat — веб-сервер и контейнер сервлетов, предназначенный для запуска Java-приложений, использующих спецификации Jakarta. Он обеспечивает выполнение серверной логики, обработку HTTP-запросов и управление жизненным циклом веб-компонентов.

1. REST API

REST API — это модель взаимодействия между клиентом и сервером, основанная на передаче данных через стандартные HTTP-методы. Службы REST используют JSON или XML в качестве формата обмена, позволяя создавать масштабируемые и легко интегрируемые веб-сервисы.

1. NGINX

NGINX — высокопроизводительный веб-сервер и прокси-сервер, который может обслуживать статические ресурсы, перенаправлять запросы и распределять нагрузку. Часто используется как балансировщик между клиентом и несколькими backend-сервисами. Используется для фильтрации запросов.

1. Janus Gateway

Janus — это сервер WebRTC-сигналинга, позволяющий организовывать видео- и аудиосвязь в реальном времени. Он действует как посредник между участниками мультимедийного обмена, обеспечивая поддержку различных потоков, конференций и подключений. Используется для организации видеоконференций. [10]

1. Coturn

Coturn — это сервер для реализации NAT-traversal с использованием протокола TURN/STUN. Он необходим для корректной работы WebRTC в условиях, когда прямое соединение между клиентами невозможно из-за сетевых ограничений. [10]

1. WebRTC

WebRTC — это набор технологий и API, позволяющих браузерам и мобильным приложениям вести аудио- и видеосвязь без установки дополнительных плагинов. Он обеспечивает P2P-соединения, передачу мультимедиа и обмен данными в реальном времени.

1. IntelliJ IDEA

IntelliJ IDEA — профессиональная среда разработки, поддерживающая множество языков, включая Java и JavaScript. Обладает мощными средствами рефакторинга, автоматизации, анализа кода и интеграции с фреймворками, системами сборки и системами контроля версий. [1]

### **Пользовательский путь**

Когда пользователь входит в систему, его встречает страница входа, показанная на рисунке 67:



Рисунок 67 – Страница входа

На ней ему предлагается ввести свои данные учётной записи (логин, пароль) и указать требуется ли сохранить его сессию. Если у пользователя нет аккаунта, он может завести его на странице регистрации, которая представлена на рисунке 68:

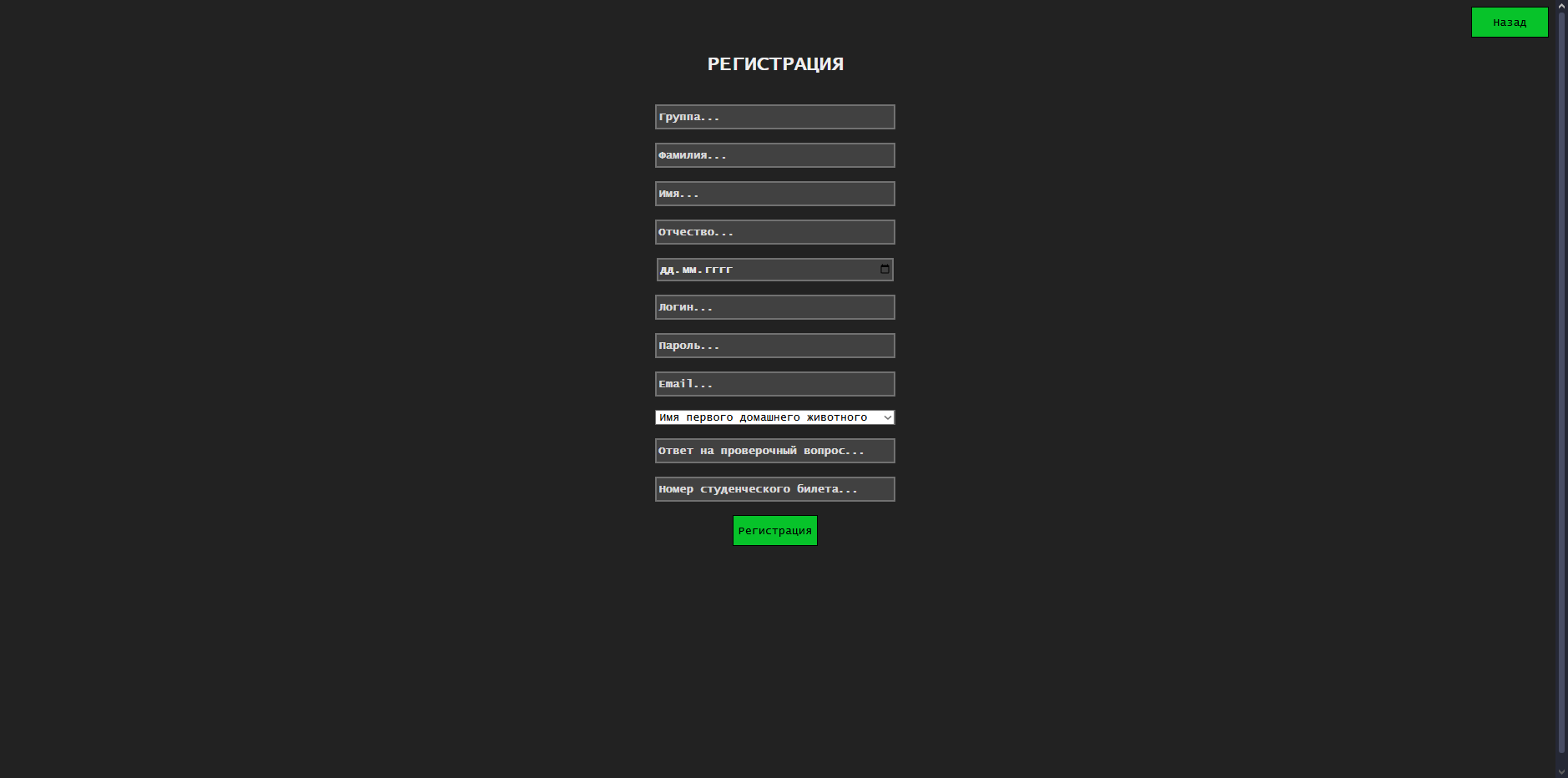


Рисунок 68 – Страница регистрации

Для регистрации в системе потребуется указать группу, фамилию, имя, логин, пароль, выбрать “проверочный” вопрос, ответ на него. Также опционально можно указать отчество, номер студенческого билета (преподаватели могут указать номер служебного удостоверения) и email (эти 2 параметра сильно упростят восстановление пароля). После успешной регистрации, пользователь вернётся на страницу входа, где должен будет ввести свои новые данные. В случае потери пароля, пользователь может восстановить его на странице восстановления пароля, которая представлена на рисунке 67:

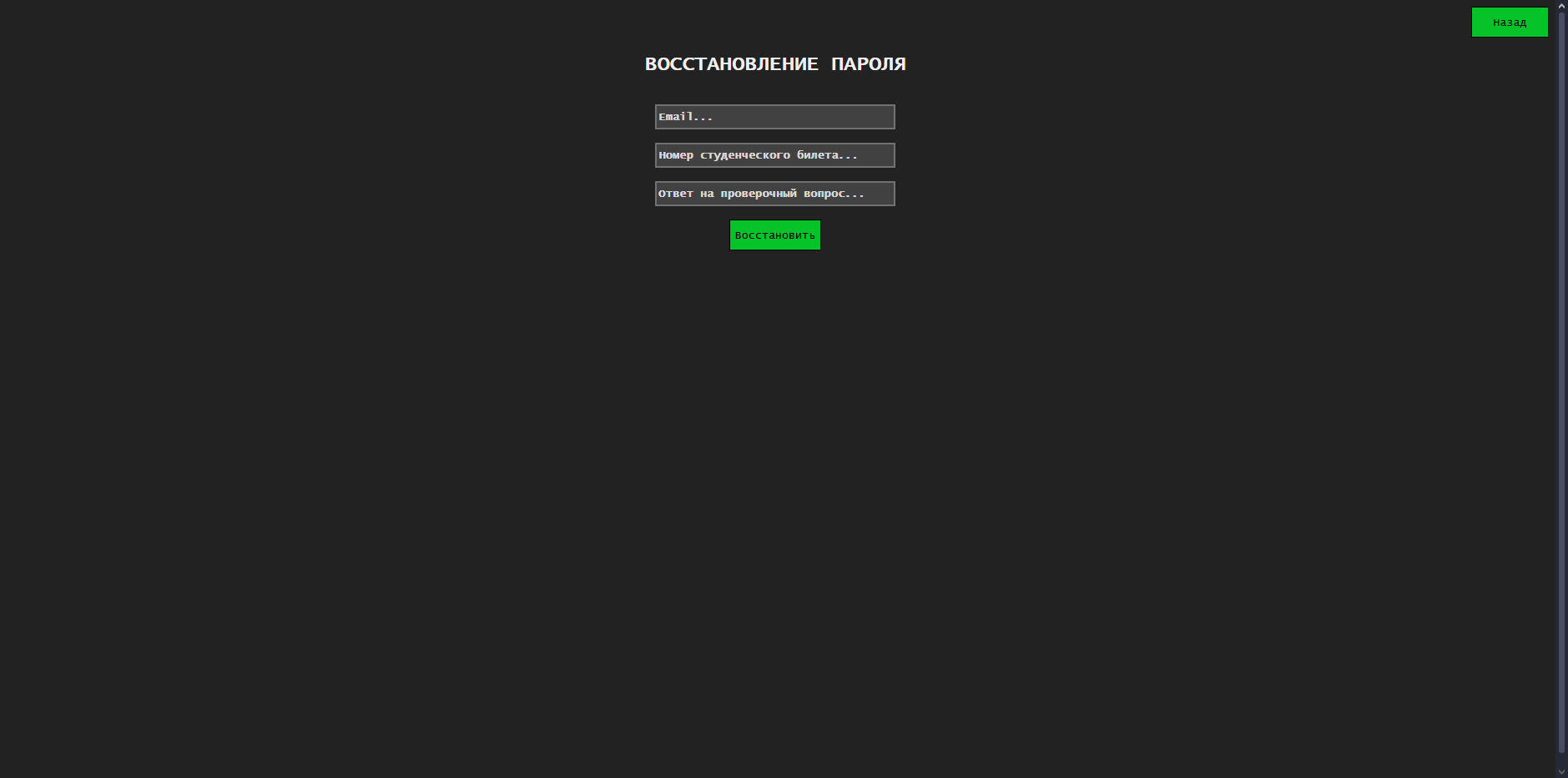


Рисунок 69 – Страница восстановления пароля

Пользователю предлагается ввести номер студенческого билета, ответ на проверочный вопрос и email (на случай потери доступа к указанному в профиле). Если номер студенеского билета не указан и указан новый email, на старый email будет выслано письмо о попытке входа в аккаунт и создастся заявка у администратора, которую ему будет необходимо проверить на признаки попытки взлома.

В случае успешной авторизации, пользователь попадает на главную страницу, где сможет перейти на страницу заданий, звонков (далее по тексту - видеоконференций), электронного журнала (далее по тексту - журнала), чатов и своего профиля. Также администраторы системы смогут перейти в меню администратора по соответствующей кнопке. На главной странице представлен календарь с предстоящими и прошедшими событиями (задания, видеоконференции). Главная страница представлена на рисунке 70:

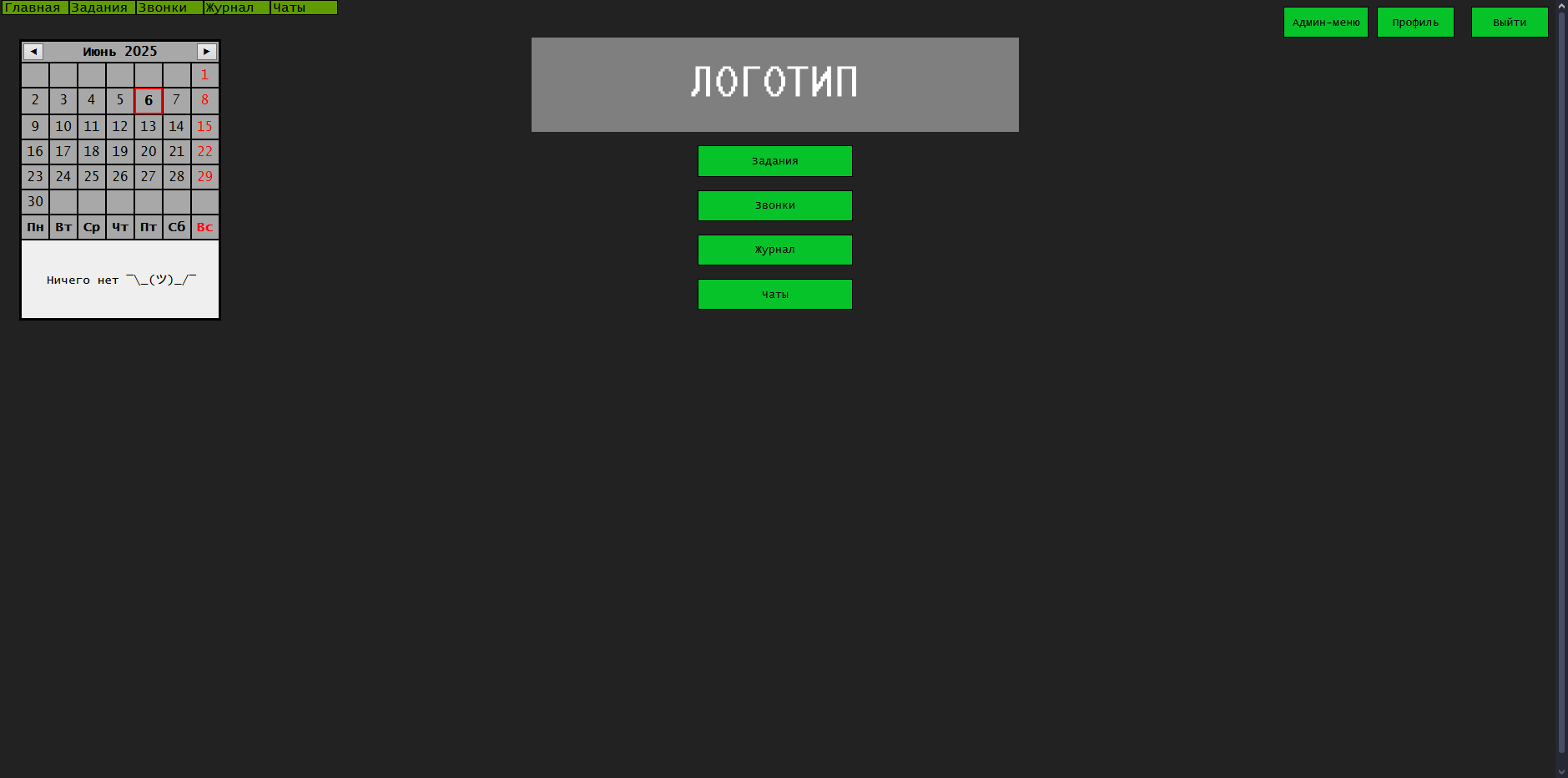


Рисунок 70 – Главная страница

На странице профиля пользователь может посмотреть данные своей учетной записи (логин, пароль, роли, группу, номер студенческого билета, имя, фамилию, отчество, дату рождения, email, “проверочный” вопрос и ответ на него) и при необходимости отредактировать её. Редактировать роли или посмотреть старый пароль нельзя. Страница профиля представлена на рисунке 71:

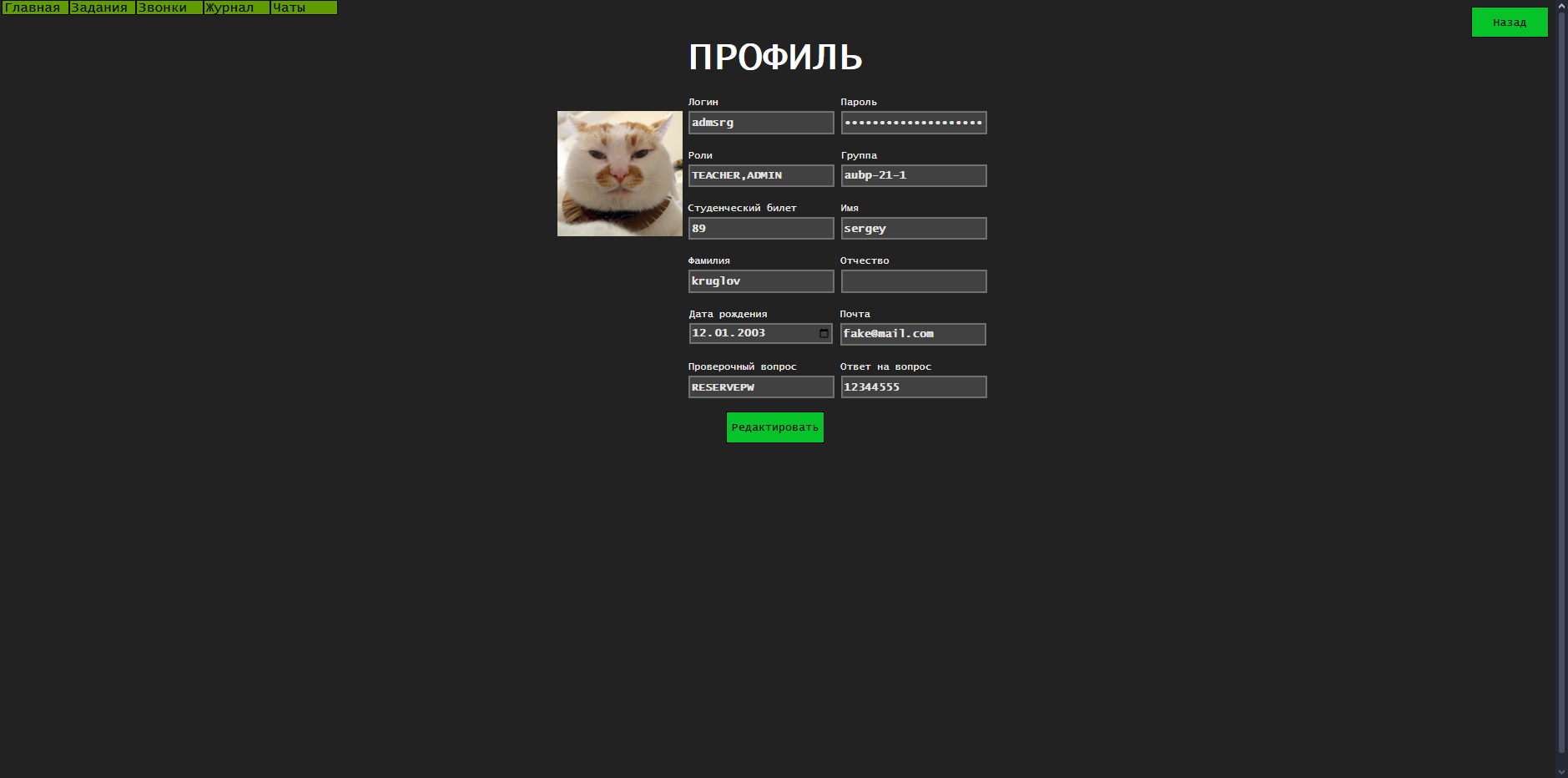


Рисунок 71 – Страница профиля

Попадая на страницу чатов, пользователь сможет увидеть в левой части страницы список чатов, участником которых он является, фильтры и кнопку создания чата, а в правой части форму отправки сообщения и окно чата. При выборе чата, в окне чата появятся все сообщения этого чата, а в верхней части окна – название чата. Страница чатов представлена на рисунке 72:

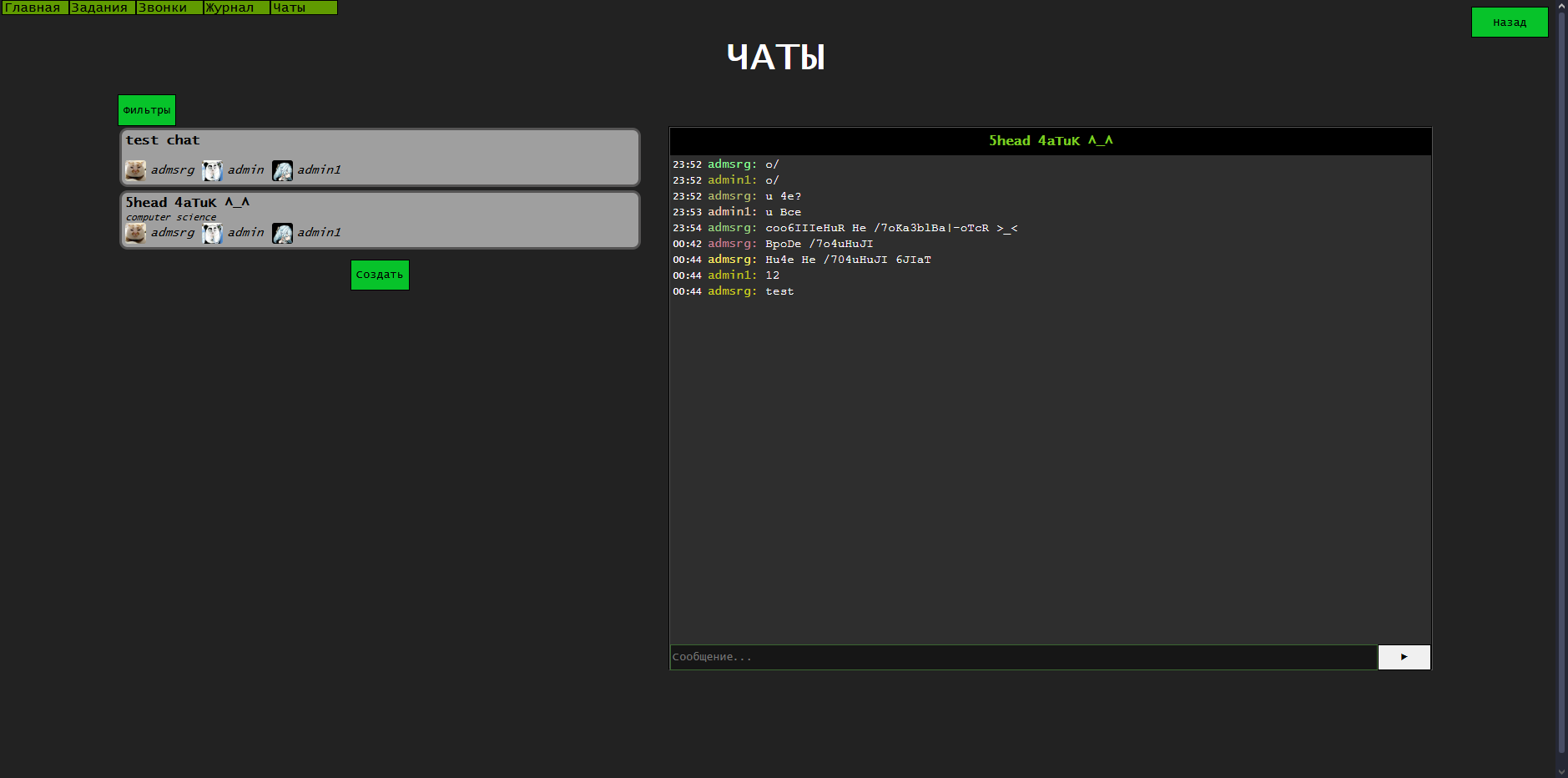


Рисунок 72 – Страница чатов

Страница задания отличается по содержимому для ролей «Студент» и «Преподаватель». Пользователь с ролью «Студент» сможет посмотреть дату выдачи, дату окончания задания, название (выделено жирным увеличенным шрифтом) и его описание, а также прикрепленные файлы. Также есть форма сдачи, где можно написать текстовый ответ и прикрепить свои файлы. После сдачи рядом с кнопкой «сдать повторно» появится дата сдачи. После проверки предподавателем появится дата проверки, оценка и комментарий от него. Страница задания представлена на рисунке 73:

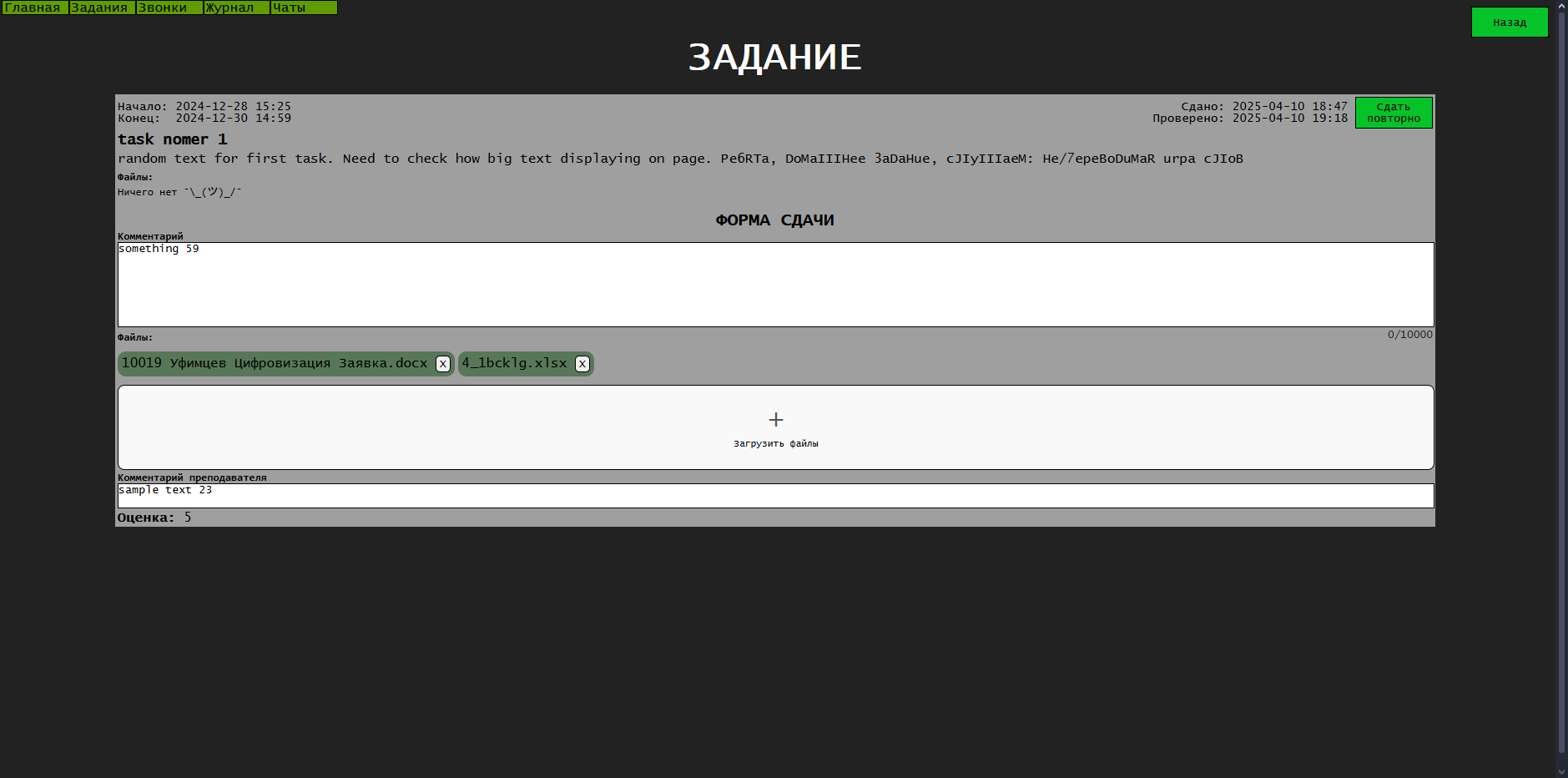


Рисунок 73 – Страница задания (вид от лица пользователя с ролью «Студент»)

Представление страницы задания для пользователя с ролью «Преподаватель» отличается наличием полей выбора студентов и группы, а также формой проверки. Страница задания представлена на рисунке 74:

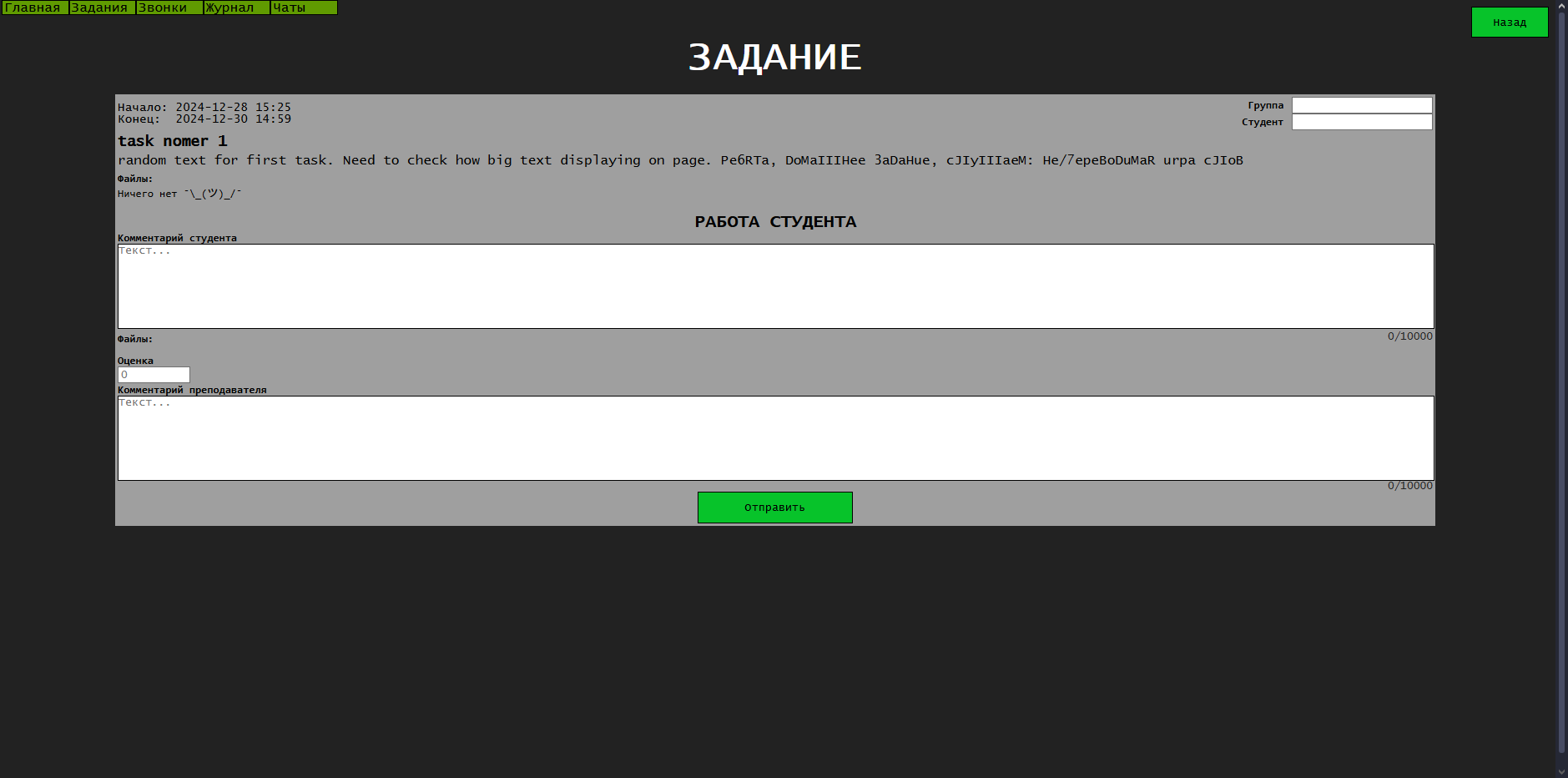


Рисунок 74 – Страница задания (вид от лица пользователя с ролью «Преподаватель»)

На странице заданий пользователь сможет увидеть список заданий. Пользователь с ролью «Преподаватель» видит созданные им задания, а пользователь с ролью «Студент» видит задания, которые ему назначили. Для облегчения поиска задания добавлены фильтры. Страница заданий представлена на рисунке 75:

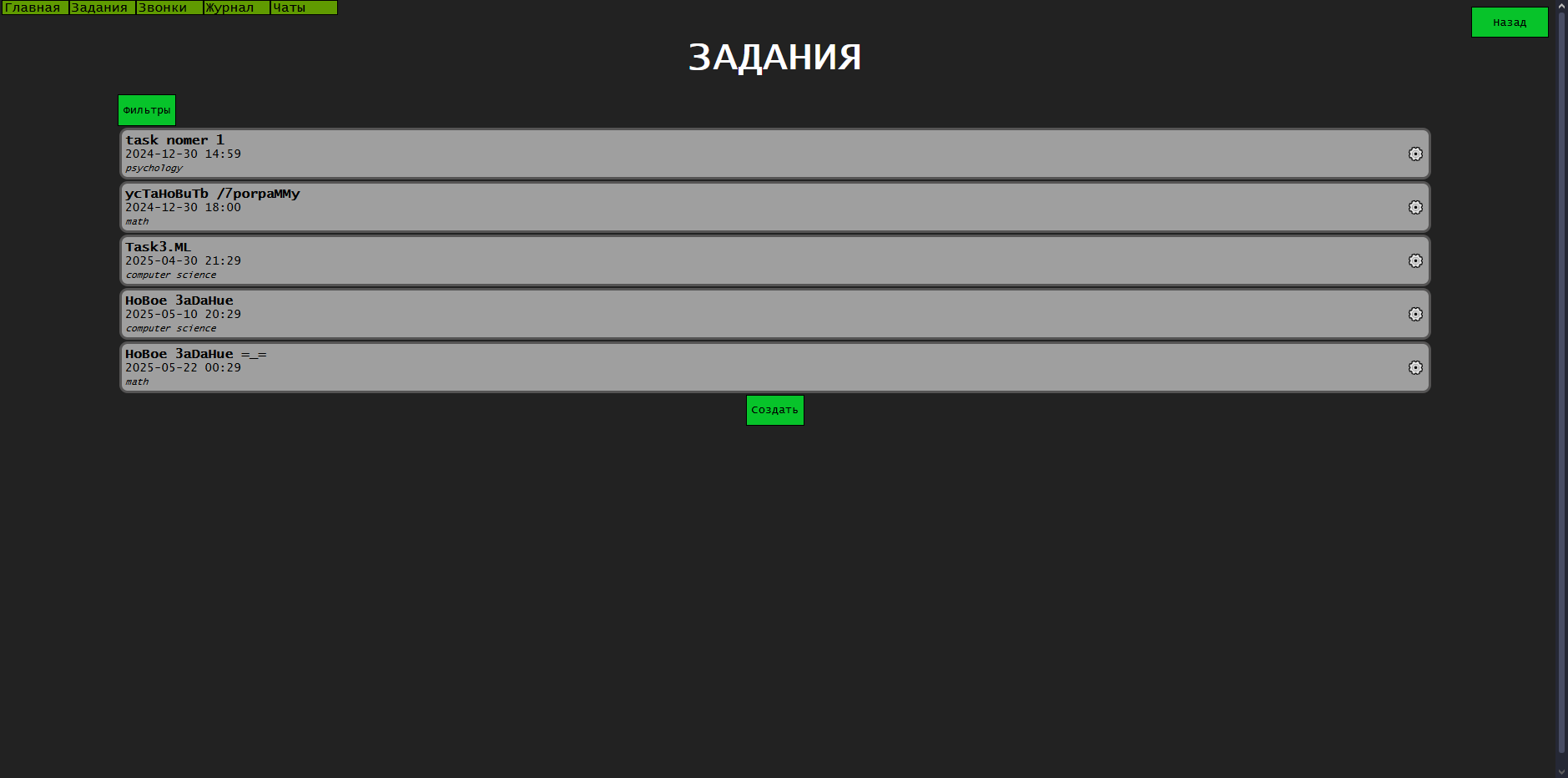


Рисунок 75 – Страница заданий

На странице создания задания пользователь может указать название, описание, указать дату окончания задания, предмет, группу и добавить файлы. Страница создания задания представлена на рисунке 76:

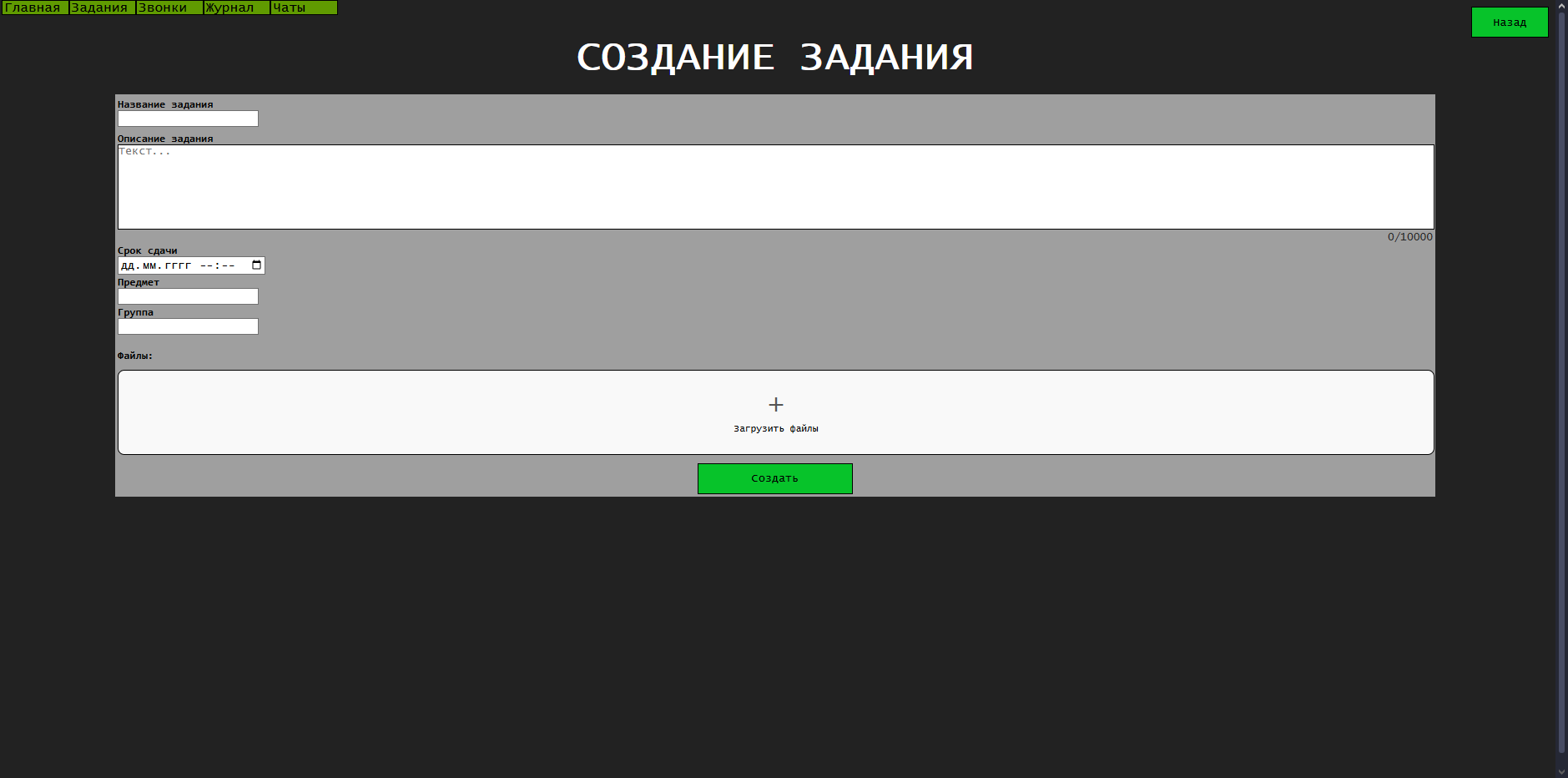


Рисунок 76 – Страница создания задания

На странице обновления задания та же самая форма, что и при создании, но она при выборе задания для изменения она заполняется его информацией. Страница обновления задания представлена на рисунке 77:

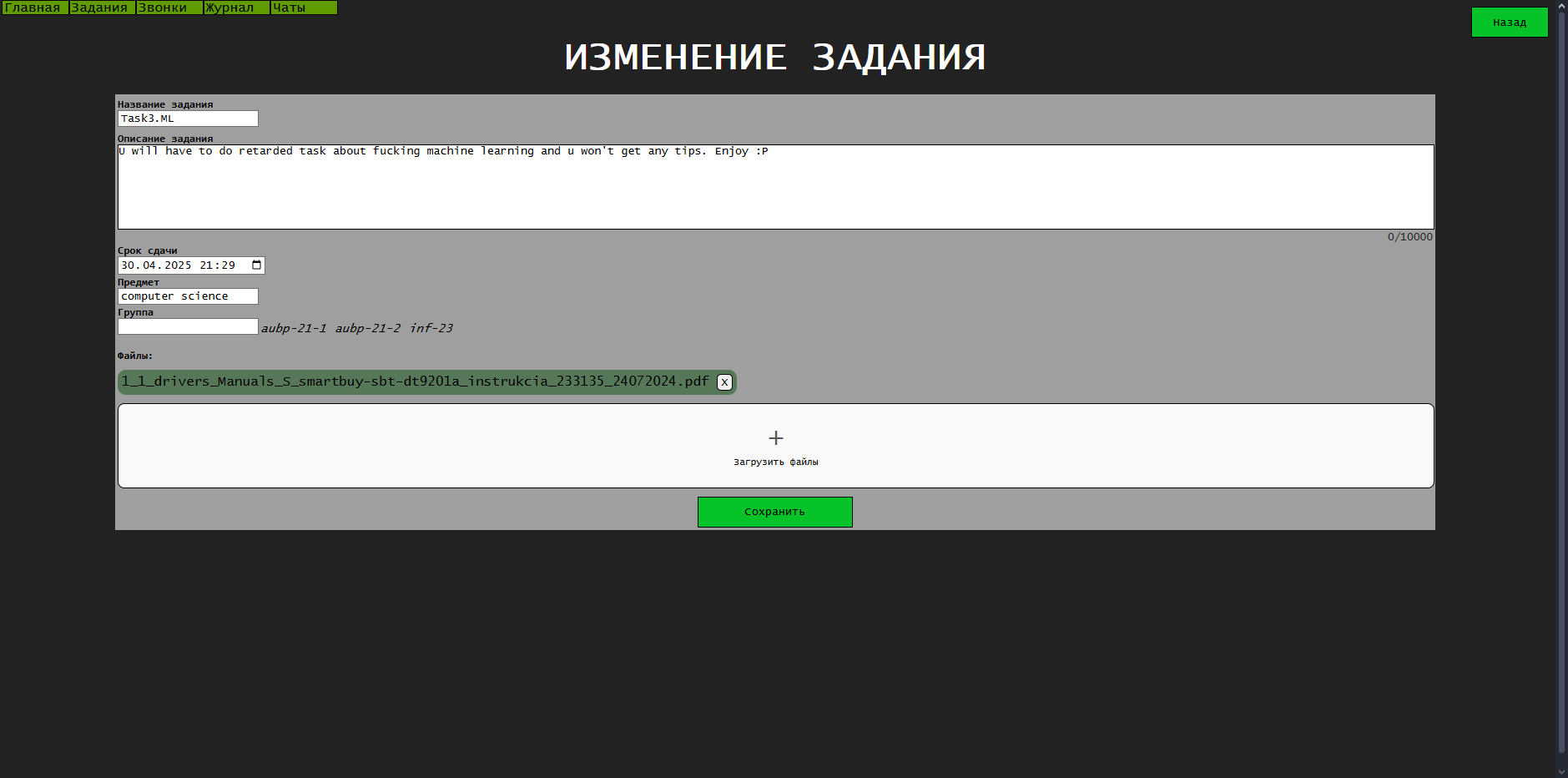


Рисунок 77 – Страница обновления задания

Представление страницы журнала зависит от роли пользователя. Пользователь с ролью «Студент» сможет посмотреть таблицу своей успеваемости, в которой указаны предмет, задание, дата, оценка и присутствие. Если задание/оценка/присутствие не указаны в записи, они отмечаются “==”. Для удобства навигации добавлены фильтры.

Страница журнала представлена на рисунке 78:

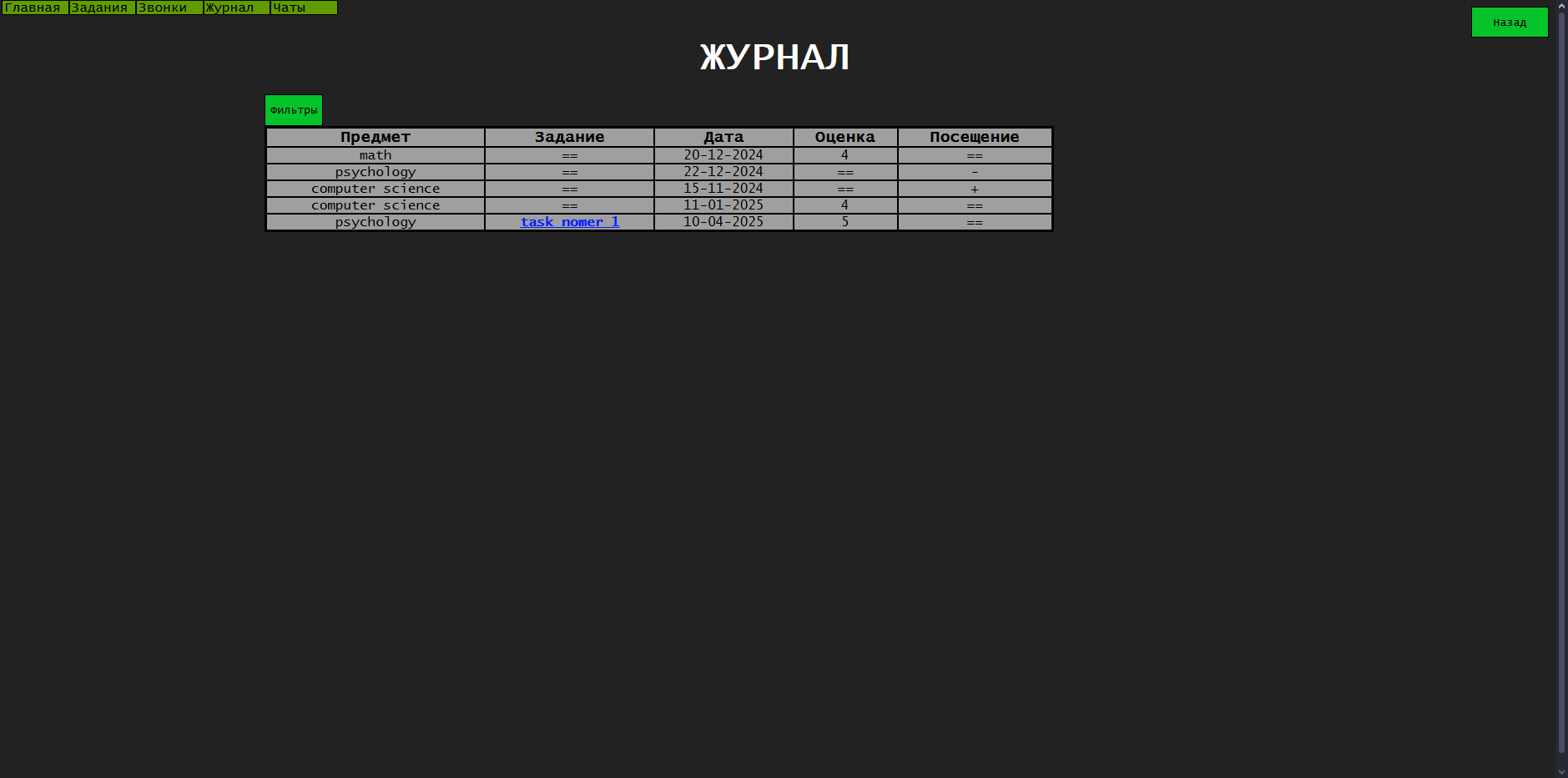


Рисунок 78 – Страница журнала (вид от лица пользователя с ролью «Студент»)

Пользователь с ролью «Преподаватель» на странице журнала видит похожую таблицу, но к ней добавляются столбцы ФИО студента и список групп, в которых он состоит. Также справа от таблицы есть форма для добавления записи в журнал, её изменения или удаления. В ней можно указать дату, студента, предмет, оценку, присутсвие. Также есть функция создания записей сразу для всей группы. Страница журнала представлена на рисунке 79:

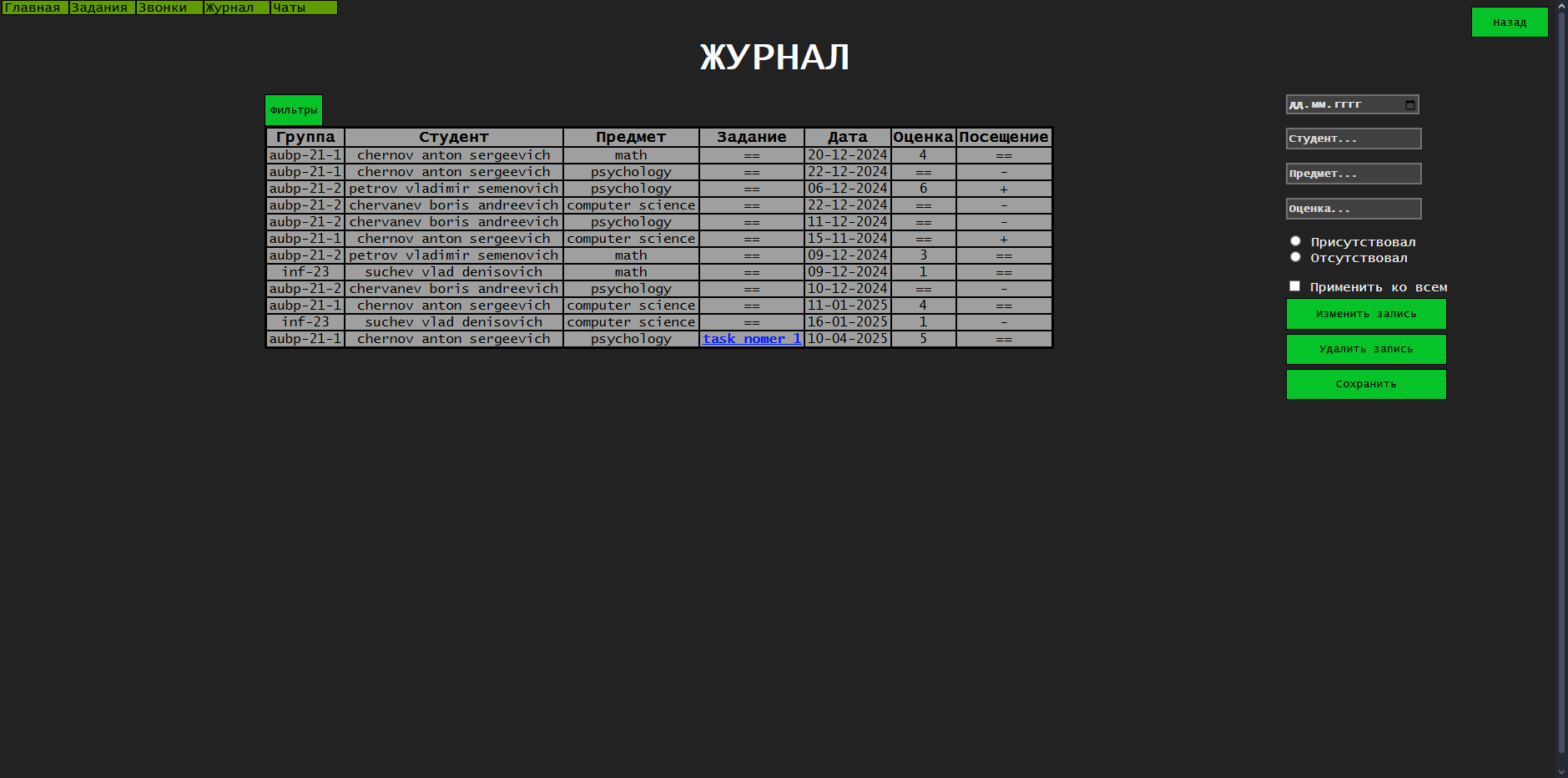


Рисунок 79 – Страница журнала (вид от лица пользователя с ролью «Преподаватель»)

На странице видеоконференций пользователь сможет увидеть два списка видеоконференций. Слева находится список активных на данный момент конференций, а справа – общий список всех доступных конференций. Для удобства навигации оба списка снабжены фильтрами. Страница конференций представлена на рисунке 80:

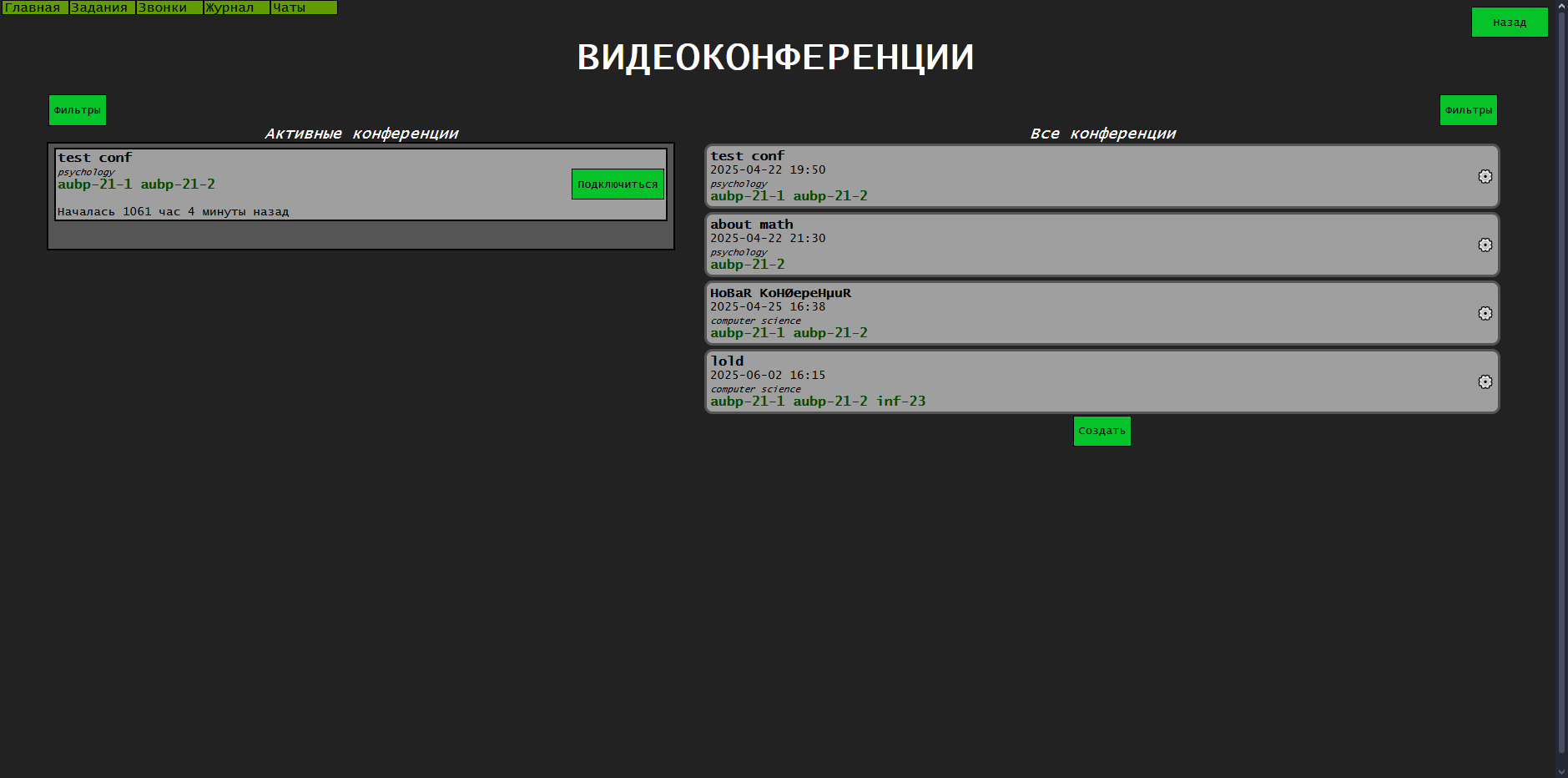


Рисунок 80 – Страница конференций

На странице создания видеоконференции пользователь может указать название, дату проведения, указать многоразовая ли конференция (одноразовые конференции удаляются из общего списка после проведения), предмет и группу.

Страница создания конференции представлена на рисунке 81:

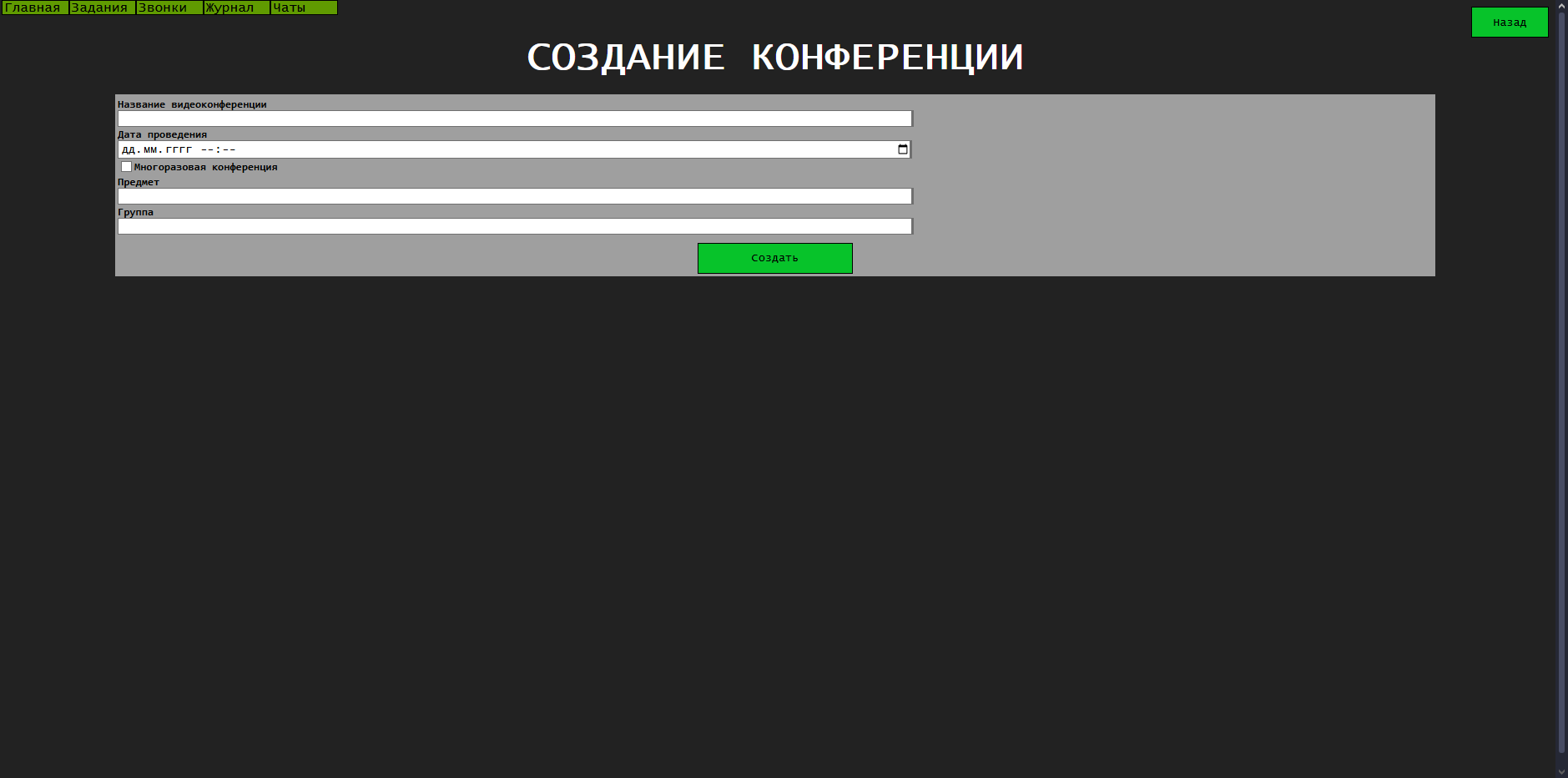


Рисунок 82 – Страница создания конференции

Форма на страница обновления конференции идентична форме создания конференции, но при выборе конференции для обновления она заполняется её информацией. Страница обновления конференции представлена на рисунке 82:

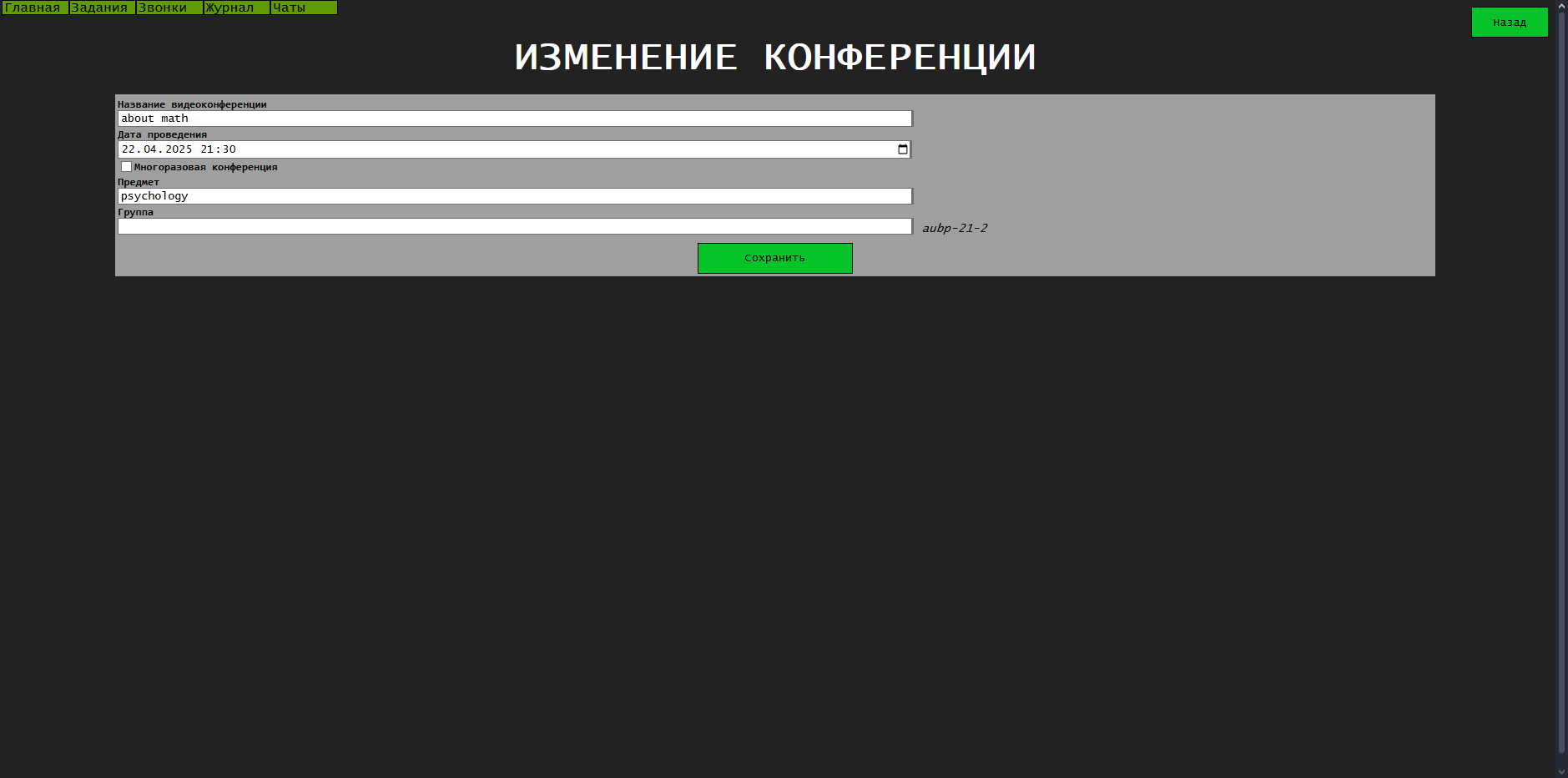


Рисунок 82 – Страница обновления конференции

Подключившись к конференции, пользователь увидит большое окно по центру, где будут находится его видеокамера и видеокамеры участников. Справа находится список участников и чат конференции с функцией упоминания участников. Каждый участник может управлять своей периферией, а именно: включать/отключать микрофон, веб-камеру, звук, демонстрацию экрана. Пользователь с ролью «Преподаватель» и выше сможет также запрещать управление своей периферией другим участникам. Страница активной конференции представлена на рисунке 83:

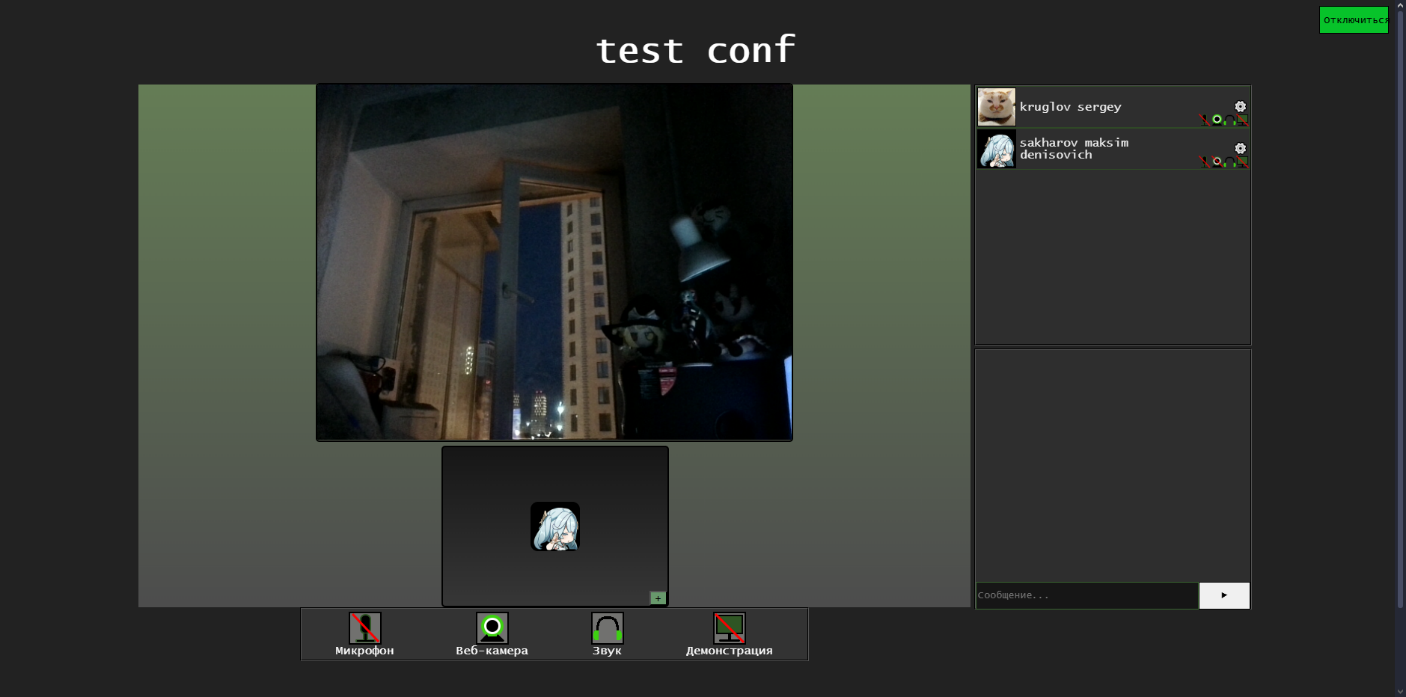


Рисунок 83 – Страница активной конференции (своя камера активна, чужие - нет)

Страница активной конференции представлена на рисунке 84:

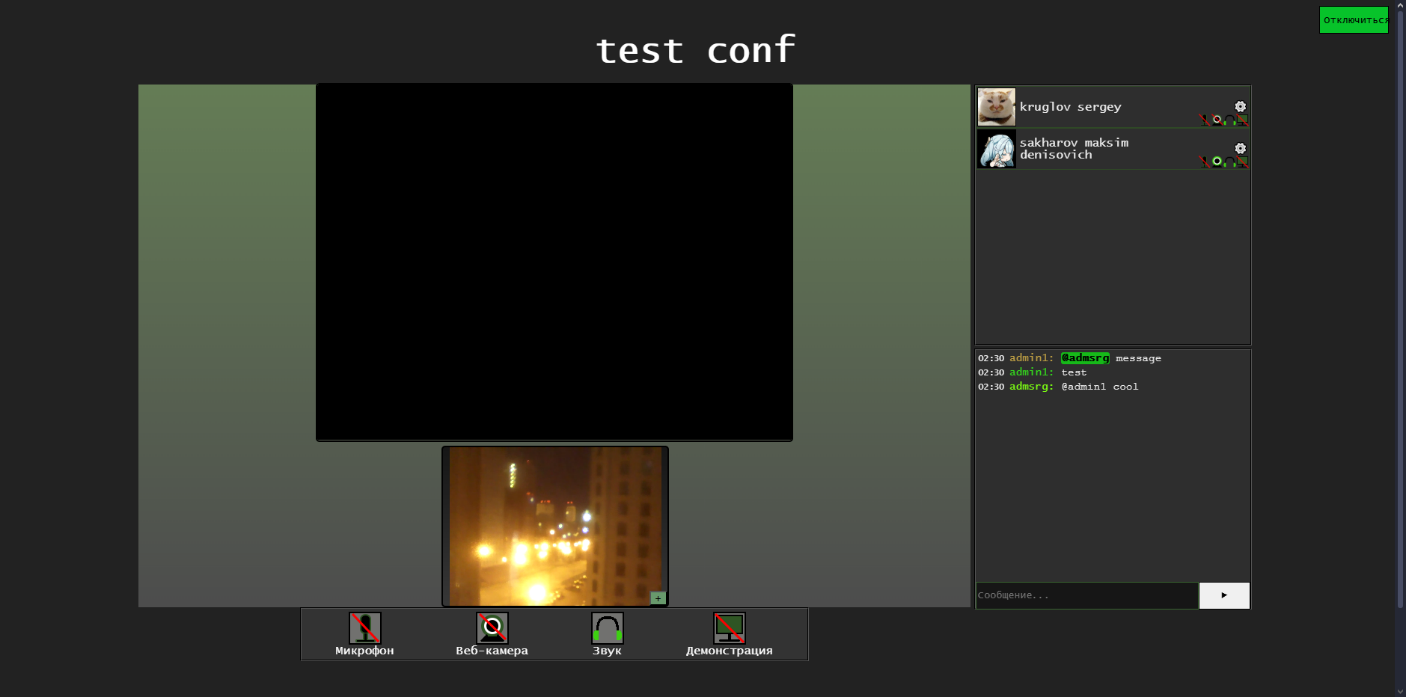


Рисунок 84 – Страница активной конференции (своя камера неактивна, чужие - активны)

### **3.4.2 Дерево проекта**

Дерево проекта изображено на рисунке 85-91:

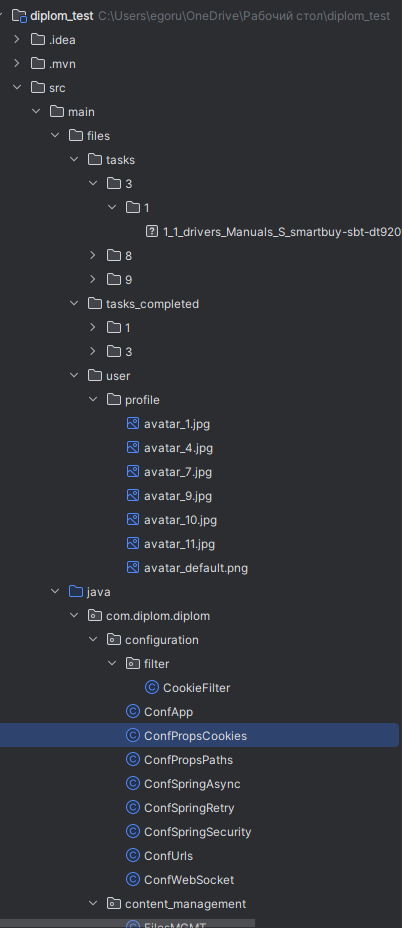
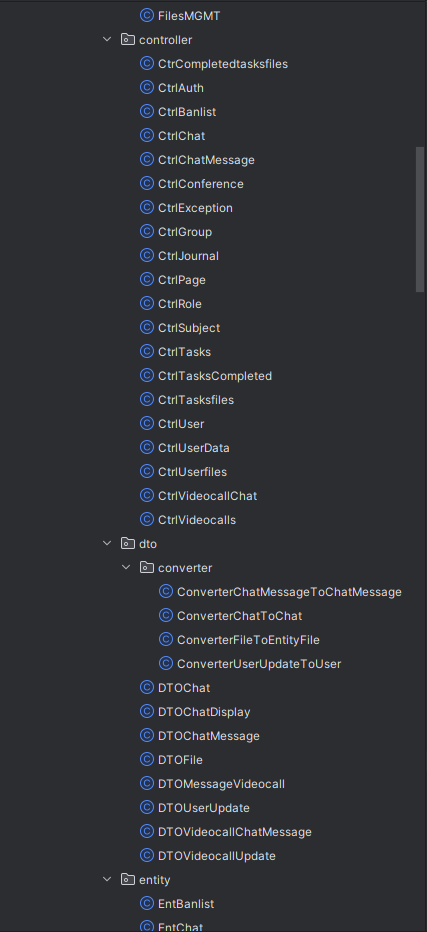


Рисунок 86 – Дерево проекта (2) Рисунок 85 – Дерево проекта (1)

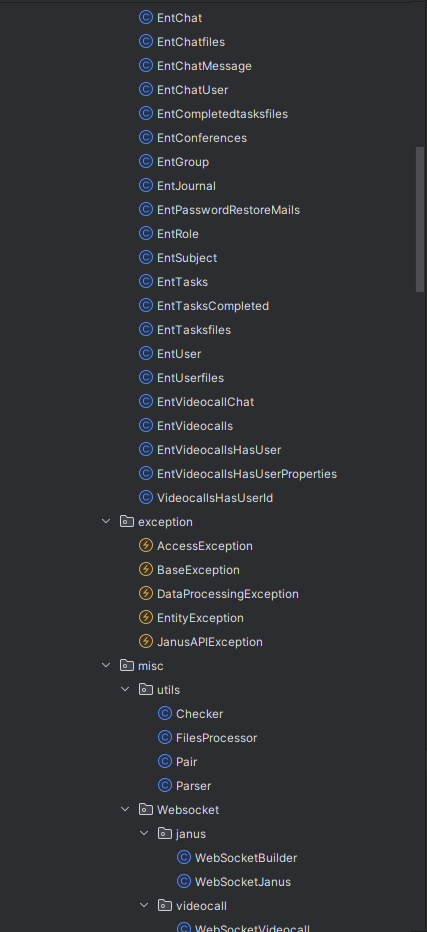
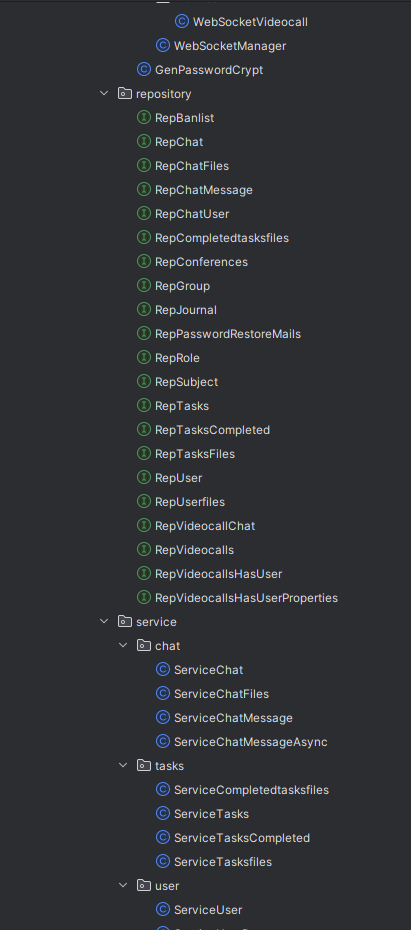


Рисунок 88 – Дерево проекта (4) Рисунок 87 – Дерево проекта (3)

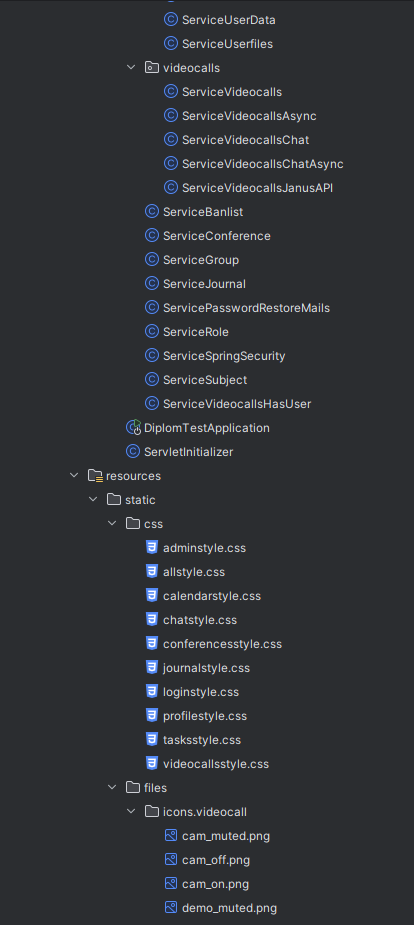


Рисунок 89 – Дерево проекта (5)

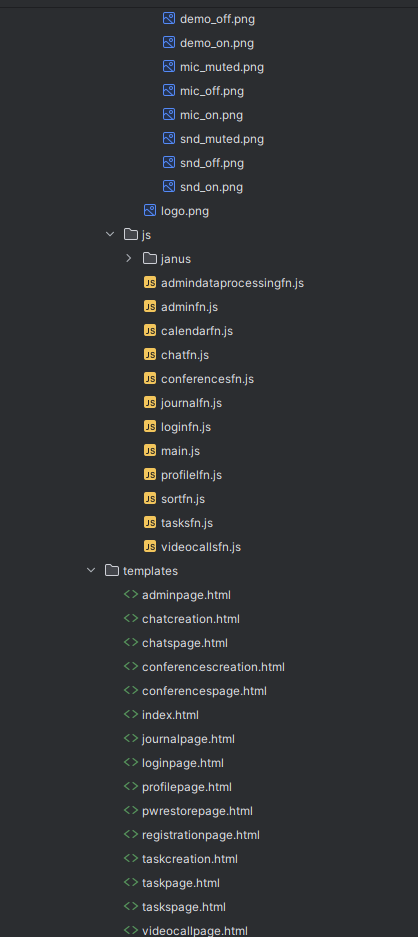


Рисунок 90 – Дерево проекта (6)

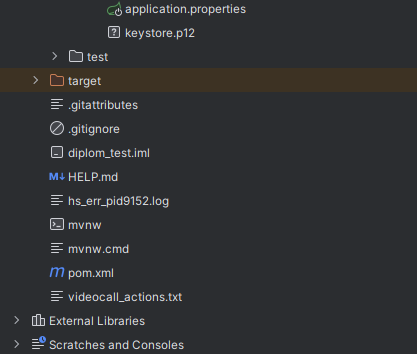


Рисунок 91 – Дерево проекта (7)

Подробное описание дерева проекта:

1. src – директория, хранящая в себе исходный код и файлы приложения
   1. main – директория, хранящая преимущественно файлы с кодом
      1. java.com.diplom.diplom – директория, хранящая java-классы и java-интерфейсы
         1. configuration – директория, хранящая конфигурационные классы
            1. filter – директория, хранящая фильтры
         2. content\_management – директория, хранящая классы, отвечающие за обработку медиаконтента (файлы и т.д.)
         3. controller – директория, хранящая классы контроллеров
         4. dto – директория, хранящая DTO-классы
            1. converter – директория, хранящая классы для конвертации DTO в сущность и наоборот
         5. entity – директория, хранящая классы сущностей
         6. exception – директория, хранящая классы исключений
         7. misc – директория, хранящая прочие классы
            1. utils – директория, хранящая утилитарные классы
            2. websocket.janus – директория, хранящая классы WebSocket для Janus
            3. websocket.videocall – директория, хранящая классы серверного WebSocket
         8. repository – директория, хранящая интерфейсы репозиториев
         9. service – директория, хранящая классы сервисов
            1. chat – директория, хранящая классы сервисов, связанных с чатами
            2. tasks – директория, хранящая классы сервисов, связанных с заданиями
            3. user – директория, хранящая классы сервисов, связанных с пользователями
            4. videocalls – директория, хранящая классы сервисов, связанных с активными видеоконференциями
      2. resources – директория, хранящая web-ресурсы и конфигурационные файлы
         1. static – директория, хранящая web-ресурсы (css-стили, js-скрипты и иконки)
            1. css – директория, хранящая css-файлы
            2. files – директория, хранящая файлы (изображения, иконки и т.д.)
            3. js – директория, хранящая javascript-файлы
         2. templates – директория, хранящая html-страницы
      3. files – директория, хранящая пользовательские файлы
         1. tasks – директория, хранящая файлы заданий
         2. tasks\_completed – директория, хранящая файлы выполненных заданий
         3. user.profile -директория, хранящая аватары пользователей

### **3.4.3 Листинги**

### Листинг 1 является листингом контроллера заданий, который отвечает за создание, удаление и изменение заданий.

Листинг 1 – контроллер заданий

@RestController  
public class CtrlTasks {  
 private final ServiceTasks srvTasks;  
  
 public CtrlTasks(ServiceTasks srvTasks) {  
 this.srvTasks = srvTasks;  
 }  
  
 @GetMapping("/getTasks")  
 public @ResponseBody List<EntTasks> getTasks(){  
 return srvTasks.getTasks();  
 }  
  
 @PostMapping(value = "/tasks/addTask",consumes = {"multipart/form-data"})  
 @Secured({"ROLE\_ADMIN","ROLE\_TEACHER"})  
 public @ResponseBody EntTasks addTask(@RequestPart(value="taskdata") EntTasks task, @RequestPart(value="groups") String groups, @RequestParam(value="subjectname") String subjectname, @RequestParam(value="user\_id") Long user\_id, @RequestPart(value = "files",required = false)MultipartFile[] files) throws EntityException {  
 return srvTasks.addTask(task,groups,subjectname,user\_id,files);  
 }  
  
 @DeleteMapping("/tasks/{id}/deleteTask")  
 @Secured({"ROLE\_ADMIN","ROLE\_TEACHER"})  
 public @ResponseBody EntTasks deleteTask(@PathVariable Long id, @AuthenticationPrincipal UserDetails userDetails) throws AccessException, EntityException {  
 return srvTasks.deleteTaskById(id,userDetails);  
 }  
  
 @PatchMapping("/tasks/update/{id}")  
 @Secured({"ROLE\_ADMIN","ROLE\_TEACHER"})  
 public @ResponseBody EntTasks updTask(@RequestPart(value="taskdata") EntTasks task, @PathVariable Long id, @RequestParam(value="subjectname",required = false) String subjectname, @RequestPart(value="groups",required = false) String groups, @RequestPart(value = "files",required = false)MultipartFile[] files, @AuthenticationPrincipal UserDetails userDetails) throws AccessException, EntityException {  
 return srvTasks.updateTask(task,id,subjectname,groups,files,userDetails);  
 }  
}

Листинг 2 является листингом контроллера выполненных заданий, который отвечает за создание, удаление и изменение выполненных заданий

Листинг 2 – контроллер выполненных заданий

@RestController  
public class CtrlTasksCompleted {  
 private final ServiceTasksCompleted srvTasksCompleted;  
  
 public CtrlTasksCompleted(ServiceTasksCompleted srvTasksCompleted) {  
 this.srvTasksCompleted = srvTasksCompleted;  
 }  
  
 @GetMapping("/getTasksCompleted")  
 public @ResponseBody List<EntTasksCompleted> getTasksCompleted(){  
 return srvTasksCompleted.getTasksCompleted();  
 }  
  
 @GetMapping("/task/{taskId}/getTaskCompletedByUserId")  
 public @ResponseBody EntTasksCompleted getTaskCompletedByUseridAndTaskid(@RequestParam(value = "userId") Long userId, @PathVariable Long taskId){  
 return srvTasksCompleted.getCompletedTaskByUserIdAndTaskId(userId,taskId);  
 }  
  
 @PostMapping("/task/{task\_id}/addTaskCompleted")  
 public @ResponseBody EntTasksCompleted addTasksCompleted(@RequestPart(value="taskdata") EntTasksCompleted task, @RequestParam(value = "user\_id") Long user\_id, @PathVariable Long task\_id, @RequestPart(value="files",required = false) MultipartFile[] files, @AuthenticationPrincipal UserDetails userDetails) throws AccessException, EntityException {  
 return srvTasksCompleted.addTask(task,task\_id,user\_id,files,userDetails);  
 }  
  
 @DeleteMapping("/delTasksCompletedById")  
 public @ResponseBody EntTasksCompleted delTasksCompletedById(Long id, @AuthenticationPrincipal UserDetails userDetails) throws AccessException, EntityException {  
 return srvTasksCompleted.delTaskById(id,userDetails);  
 }  
  
 @PatchMapping("/task/{task\_id}/updateTaskCompleted")  
 public @ResponseBody EntTasksCompleted updTasksCompleted(EntTasksCompleted task, @AuthenticationPrincipal UserDetails userDetails) throws AccessException, EntityException {  
 return srvTasksCompleted.updateTaskByUser(task,null,userDetails);  
 }  
  
 @PatchMapping("/task/{task\_id}/checkTaskCompleted")  
 @Secured({"ROLE\_ADMIN","ROLE\_TEACHER"})  
 public @ResponseBody EntTasksCompleted updateTaskCompletedByCheck(@RequestBody EntTasksCompleted task, @RequestParam(value = "taskId",required = false) Long task\_id, @RequestParam(value = "userId",required = false) Long user\_id, @AuthenticationPrincipal UserDetails userDetails) throws AccessException, EntityException {  
 srvTasksCompleted.updateTaskByCheck(task,userDetails);  
 return task;  
 }  
}

Листинг 3 является листингом контроллера электронного журнала, который позволяет создавать, изменять, удалить и получать записи электронного журнала.

Листинг 3 – контроллер электронного журнала

@RestController  
public class CtrlJournal {  
 private final ServiceJournal srvJournal;  
  
 public CtrlJournal(ServiceJournal srvJournal) {  
 this.srvJournal = srvJournal;  
 }  
  
 @GetMapping(path = "/getJournalByUserName")  
 public @ResponseBody List<EntJournal> getAllJournalForUserByName(@RequestParam String name) throws EntityException {  
 return srvJournal.getAllJournalForUserByName(name);  
 }  
  
 @PostMapping(path = "/addJournalUser")  
 @Secured({"ROLE\_ADMIN","ROLE\_TEACHER"})  
 public @ResponseBody EntJournal addJournalUser(@RequestParam(value = "journaluserId") Long journaluserId, @RequestParam(value = "journalsubjectId") Integer journalsubjectId, @RequestBody EntJournal journal) throws EntityException {  
 return srvJournal.addJournalUser(journal,journalsubjectId,journaluserId);  
 }  
  
 @PostMapping(path = "/addJournalGroup")  
 @Secured({"ROLE\_ADMIN","ROLE\_TEACHER"})  
 public @ResponseBody List<EntJournal> addJournalGroup(@RequestParam(value = "journaluserId") Long journaluserId, @RequestParam(value = "journalsubjectId") Integer journalsubjectId, @RequestBody EntJournal journal) throws EntityException {  
 return srvJournal.addJournalGroup(journal,journalsubjectId,journaluserId);  
 }  
  
 @DeleteMapping(path = "/delJournalUser/{id}")  
 @Secured({"ROLE\_ADMIN","ROLE\_TEACHER"})  
 public @ResponseBody EntJournal delJournalUser(@PathVariable Long id, @AuthenticationPrincipal UserDetails userDetails) throws AccessException, EntityException {  
 return srvJournal.delJournalUserById(id,userDetails);  
 }  
  
 @PatchMapping(path = "/updJournalUser/{id}")  
 @Secured({"ROLE\_ADMIN","ROLE\_TEACHER"})  
 public @ResponseBody EntJournal updJournalUser(@RequestParam(value = "journalsubjectId") Long journaluserId, @RequestParam(value = "journalsubjectId") Integer journalsubjectId, @RequestBody EntJournal journal, @PathVariable Long id, @AuthenticationPrincipal UserDetails userDetails) throws AccessException, EntityException {  
 return srvJournal.updJournalUser(journal,journalsubjectId,journaluserId,id,userDetails);  
 }  
}

Листинг 4 является листингом контроллера конференций, который позволяет создавать, изменять и удалять конференции.

Листинг 4 – контроллер конференций

@RestController  
public class CtrlConference {  
 private final ServiceConference srvConference;  
  
 public CtrlConference(ServiceConference srvConference) {  
 this.srvConference = srvConference;  
 }  
  
 @GetMapping("/getConferences")  
 public @ResponseBody List<EntConferences> getConferences(){  
 return srvConference.getConferences();  
 }  
  
 @PostMapping("/conference/create/addConference")  
 @Secured({"ROLE\_ADMIN","ROLE\_TEACHER"})  
 public @ResponseBody EntConferences addConference(@RequestPart(value="conferencedata") EntConferences conference, @RequestPart(value="groups") String groups, @RequestParam(value="subjectname") String subjectname, @AuthenticationPrincipal UserDetails userDetails) throws EntityException {  
 return srvConference.addConference(conference,groups,subjectname,userDetails);  
 }  
  
 @PatchMapping("/conference/{id}/update/updateConference")  
 @Secured({"ROLE\_ADMIN","ROLE\_TEACHER"})  
 public @ResponseBody EntConferences updateConference(@RequestPart(value="conferencedata") EntConferences conference, @PathVariable Long id, @RequestPart(value="groups") String groups, @RequestParam(value="subjectname") String subjectname, @AuthenticationPrincipal UserDetails userDetails) throws AccessException, EntityException {  
 return srvConference.updateConferenceById(conference,id,groups,subjectname,userDetails);  
 }  
  
 @DeleteMapping("/conferences/{id}/deleteConference")  
 @Secured({"ROLE\_ADMIN","ROLE\_TEACHER"})  
 public @ResponseBody EntConferences deleteConferenceById(@PathVariable Long id, @AuthenticationPrincipal UserDetails userDetails) throws AccessException, EntityException {  
 return srvConference.deleteConferenceById(id,userDetails);  
 }  
}

Листинг 5 является листингом контроллера активных конференций, который позволяет присоединяться, отключаться, управлять периферией участников.

Листинг 5 – контроллер активных конференций

@RestController  
public class CtrlVideocalls {  
 private final ServiceVideocalls srvVideocalls;  
 private final ServiceVideocallsAsync srvVideocallsAsync;  
  
 public CtrlVideocalls(ServiceVideocalls srvVideocalls, ServiceVideocallsAsync srvVideocallsAsync) {  
 this.srvVideocalls = srvVideocalls;  
 this.srvVideocallsAsync = srvVideocallsAsync;  
 }  
  
 @GetMapping("/videocall/{id}/user/getData")  
 public @ResponseBody EntVideocallsHasUser getUserData(@PathVariable Long id, @AuthenticationPrincipal UserDetails userDetails) throws AccessException, EntityException {  
 return srvVideocalls.getVideocallsHasUserByUserDetailsAndVideocallId(userDetails,id);  
 }  
  
 @GetMapping("/videocall/{id}/join")  
 public @ResponseBody CompletableFuture<ResponseEntity<?>> joinVideocall(@PathVariable Long id, @AuthenticationPrincipal UserDetails userDetails) throws URISyntaxException, ExecutionException, InterruptedException, AccessException, EntityException, JanusAPIException {  
 return srvVideocallsAsync.joinVideocalls(id,userDetails);  
 }  
  
 @PostMapping("/videocall/{confId}/add")  
 public @ResponseBody EntVideocalls addVideocall(@PathVariable Long confId, @AuthenticationPrincipal UserDetails userDetails) throws URISyntaxException, ExecutionException, InterruptedException, EntityException, JanusAPIException {  
 return srvVideocalls.addVideocall(confId);  
 }  
  
 @PostMapping("/videocall/{id}/leave")  
 public @ResponseBody CompletableFuture<ResponseEntity<?>> leaveVideocall(@PathVariable Long id, @RequestParam(value = "reason") ServiceVideocalls.LeaveReasons reason, @AuthenticationPrincipal UserDetails userDetails) throws ExecutionException, InterruptedException, AccessException, EntityException, JanusAPIException {  
 return srvVideocallsAsync.leaveVideocall(id,reason,userDetails);  
 }  
  
 @PostMapping("/videocall/{id}/user/update")  
 public @ResponseBody CompletableFuture<ResponseEntity<?>> updateVideocallByAction(  
 @PathVariable Long id,  
 @RequestParam(value = "self") boolean self,  
 @RequestParam(value = "userUpdatedId",required = false) Long userId,  
 @RequestParam(value = "action") ServiceVideocalls.UpdateActions action,  
 @RequestParam(value = "state",required = false) EntVideocallsHasUser.defaultStates state,  
 @AuthenticationPrincipal UserDetails userDetails  
 ) throws AccessException, EntityException, DataProcessingException {  
 return self ? srvVideocallsAsync.updateUserSelfByAction(id,action,state,userDetails) : srvVideocallsAsync.updateUserOtherByAction(id,userId,action,state,userDetails);  
 }  
}

Листинг 6 является листингом контроллера чатов, который позволяет создавать, изменять и удалять чаты

Листинг 6 – контроллер чатов

@RestController  
public class CtrlChat {  
 private final ServiceChat srvChat;  
  
 @Autowired  
 public CtrlChat(ServiceChat srvChat) {  
 this.srvChat = srvChat;  
 }  
  
 @PostMapping("/chat/create/addChat")  
 public @ResponseBody DTOChat addChat(@RequestBody DTOChat chat, @AuthenticationPrincipal UserDetails userDetails) throws EntityException, AccessException {  
 return srvChat.addChat(chat,userDetails);  
 }  
  
  
}

Листинг 7 является листингом контроллера сообщений чата, позволяющим создавать, получать и удалять сообщения чата

Листинг 7 – контроллер сообщений чата

@RestController  
public class CtrlChatMessage {  
 private final ServiceChatMessage srvChatMessage;  
 private final ServiceChatMessageAsync srvChatAsync;  
  
 @Autowired  
 public CtrlChatMessage(ServiceChatMessage srvChatMessage, ServiceChatMessageAsync srvChatAsync) {  
 this.srvChatMessage = srvChatMessage;  
 this.srvChatAsync = srvChatAsync;  
 }  
  
 @GetMapping("/getMessages/{chatid}")  
 public @ResponseBody List<DTOChatMessage> getMessagesInChat(@PathVariable Long chatid, @RequestParam(value = "page",defaultValue = "0") int page, @AuthenticationPrincipal UserDetails userDetails) throws EntityException, AccessException {  
 return srvChatMessage.getMessagesInChat(chatid,page,userDetails);  
 }  
  
 @PostMapping("/addMessage/{chatid}")  
 public @ResponseBody CompletableFuture<ResponseEntity<?>> addMessage(@PathVariable Long chatid, @RequestParam(value="replyTo",required = false) Long replyId, @RequestBody EntChatMessage entChatMessage, @AuthenticationPrincipal UserDetails userDetails) throws EntityException, AccessException {  
 return srvChatAsync.addMessageToChat(chatid,replyId,entChatMessage,userDetails);  
 }  
}

Листинг 8 является листингом контроллера пользователей, позволяющим создавать, изменять и удалять аккаунты пользователей.

Листинг 8 – контроллер пользователей

@RestController  
public class CtrlUser {  
 private final ServiceUser srvUser;  
  
 public CtrlUser(ServiceUser srvUser) {  
 this.srvUser = srvUser;  
 }  
  
 @GetMapping("/getUsers")  
 public @ResponseBody List<DTOUserUpdate> getUsers(@RequestParam(value="page") int page){  
 return srvUser.getUsersAll(page);  
 }  
  
 @GetMapping(path = "/admin/menu/getUsersAll")  
 @Secured("ROLE\_ADMIN")  
 public @ResponseBody List<DTOUserUpdate> getUsersAll(){  
 return srvUser.getUsersDTO();  
 }  
  
 @GetMapping(path="/admin/menu/getUserById/{id}")  
 @Secured("ROLE\_ADMIN")  
 public @ResponseBody DTOUserUpdate getUserById(@PathVariable Long id) throws EntityException {  
 return srvUser.getDTOUserById(id);  
 }  
  
 @GetMapping(path = "/admin/menu/getUserByLogin")  
 @Secured("ROLE\_ADMIN")  
 public @ResponseBody EntUser getUserByLogin(UserDetails userDetails) throws EntityException {  
 return srvUser.getUserByLogin(userDetails);  
 }  
  
 @DeleteMapping(path="/admin/menu/delUser/{id}")  
 @Secured("ROLE\_ADMIN")  
 public @ResponseBody DTOUserUpdate delUserById(@PathVariable Long id,@AuthenticationPrincipal UserDetails userDetails) throws AccessException, EntityException {  
 return srvUser.delUserById(id, userDetails);  
 }  
  
 @PatchMapping(path = "/updUser/{id}")  
 public @ResponseBody EntUser updUser(@AuthenticationPrincipal UserDetails userDetails, @RequestBody DTOUserUpdate user, @PathVariable Long id, @RequestParam(value = "group\_name",required = false) String group\_name, @RequestParam(value="admin",required = false,defaultValue = "false") boolean admin) throws AccessException, EntityException {  
 return srvUser.updUser(user,id,group\_name,userDetails,admin);  
 }  
}

Листинг 9 является листингом репозитория заданий, предоставляющим инструменты для создания запросов в таблицу заданий в базе данных

Листинг 9 – репозиторий заданий

@Repository  
public interface RepTasks extends JpaRepository<EntTasks,Long> {  
 List<EntTasks> findAllByGroups(EntGroup group);  
 List<EntTasks> findAllByCreatedby(EntUser user);  
 @Query("SELECT t FROM EntTasks t JOIN t.groups g WHERE g.id = :groupId AND t.dateend BETWEEN :start AND :end")  
 List<EntTasks> findTasksByGroupAndMonth(@Param("groupId") Long groupId, @Param("start") LocalDateTime start, @Param("end") LocalDateTime end);  
}

Листинг 10 является листингом репозитория выполненных заданий, предоставляющим инструменты для создания запросов в таблицу выполненных заданий в базе данных

Листинг 10 – репозиторий выполненных заданий

public interface RepTasksCompleted extends JpaRepository<EntTasksCompleted,Long> {  
 List<EntTasksCompleted> findAllByTasksId(EntTasks tasksId);  
 Optional<EntTasksCompleted> findByUserIdAndTasksId(EntUser userId, EntTasks tasksId);  
}

Листинг 11 является листингом репозитория электронного журнала, предоставляющим инструменты для создания запросов в таблицу электронного журнала в базе данных

Листинг 11 – репозиторий электронного журнала

@Repository  
public interface RepJournal extends CrudRepository<EntJournal,Long> {  
 List<EntJournal> findAllByJournaluserId(EntUser user);  
 Optional<EntJournal> findByJournaltasksCompletedId(EntTasksCompleted journaltasksCompletedId);  
}

Листинг 12 является листингом репозитория конференций, предоставляющим инструменты для создания запросов в таблицу конференций в базе данных

Листинг 12 – репозиторий конференций

@Repository  
public interface RepConferences extends JpaRepository<EntConferences,Long> {  
 List<EntConferences> findAllByGroupId(EntGroup groupId);  
 List<EntConferences> findAllByCreatedby(EntUser user);  
  
 @Query("SELECT t FROM EntConferences t WHERE t.datestart BETWEEN :start AND :end")  
 List<EntConferences> findTasksByMonth(@Param("start") LocalDateTime start, @Param("end") LocalDateTime end);  
  
 @Lock(LockModeType.*PESSIMISTIC\_WRITE*)  
 @Query("SELECT c FROM EntConferences c WHERE c.id = :id")  
 Optional<EntConferences> findByIdwithLock(@Param("id") Long id);  
}

Листинг 13 является листингом репозитория активных конференций, предоставляющим инструменты для создания запросов в таблицу активных конференций в базе данных

Листинг 13 – репозиторий активных конференций

@Repository  
public interface RepVideocallsHasUser extends CrudRepository<EntVideocallsHasUser, Long> {  
 @Query("SELECT v FROM EntVideocallsHasUser v WHERE v.videocallsId = :call AND v.videocalluserId = :user")  
 Optional<EntVideocallsHasUser> findByVideocalluserId(@Param("user") EntUser videocalluserId, @Param("call") EntVideocalls call);  
 void deleteAllByVideocallsId(EntVideocalls videocallsId);  
 List<EntVideocallsHasUser> findAllByVideocallsId(EntVideocalls videocall);  
  
 @Lock(LockModeType.*PESSIMISTIC\_WRITE*)  
 @Query("SELECT h FROM EntVideocallsHasUser h WHERE h.videocalluserId = :user AND h.videocallsId = :call")  
 Optional<EntVideocallsHasUser> findByVideocalluserIdwithLock(@Param("user") EntUser user, @Param("call") EntVideocalls videocall);  
  
 Optional<EntVideocallsHasUser> findByVideocalluserId\_Id(Long id);  
  
 @Modifying  
 @Query("UPDATE EntVideocallsHasUser v SET v.connected = :connected WHERE v.videocalluserId.id = :userId")  
 void updateConnectionStatus(@Param("userId") Long userId, @Param("connected") boolean connected);  
  
}

Листинг 14 является листингом чатов, предоставляющим инструменты для создания запросов в таблицу чатов в базе данных

Листинг 14 – репозиторий чатов

@Repository  
public interface RepChat extends JpaRepository<EntChat, Long> {  
}

Листинг 15 является листингом репозитория сообщений чата, предоставляющим инструменты для создания запросов в таблицу сообщений чата в базе данных

Листинг 15 – репозиторий сообщений чата

@Repository  
public interface RepChatMessage extends JpaRepository<EntChatMessage, Long> {  
 Page<EntChatMessage> findAllByChatId(EntChat chatId, Pageable pageable);  
}

Листинг 16 является листингом репозитория пользователей, предоставляющим инструменты для создания запросов в таблицу пользователей в базе данных

Листинг 16 – репозиторий пользователей

@Repository  
public interface RepUser extends JpaRepository<EntUser,Long> {  
 Optional<EntUser> findByLogin(String login);  
 Optional<EntUser> findByFirstname(String name);  
 Optional<EntUser> findByStudentcard(String cardnumber);  
 Page<EntUser> findAll(Pageable pageable);  
 List<EntUser> findAllByLoginIn(List<String> logins);  
 List<EntUser> findAllByUsergroupId(EntGroup group);  
 List<EntUser> findAllByRolesIn(List<EntRole> roles);  
}

Листинг 17 является листингом сервиса заданий, реализующим бизнес-логику для работы с заданиями

Листинг 17 – сервис заданий

@Service  
public class ServiceTasks {  
 private final RepTasks rTasks;  
 private final RepUser rUser;  
 private final RepSubject rSubject;  
 private final RepGroup rGroup;  
 private final ServiceTasksfiles srvTasksfiles;  
  
 @Autowired  
 public ServiceTasks(RepTasks rTasks, RepUser rUser, RepSubject rSubject, RepGroup rGroup, ServiceTasksfiles srvTasksfiles) {  
 this.rTasks = rTasks;  
 this.rUser = rUser;  
 this.rSubject = rSubject;  
 this.rGroup = rGroup;  
 this.srvTasksfiles = srvTasksfiles;  
 }  
  
 public List<EntTasks> getTasks(){  
 return rTasks.findAll();  
 }  
  
 public List<EntTasks> getTasksByGroupId(EntGroup group){  
 return rTasks.findAllByGroups(group);  
 }  
  
 public List<EntTasks> getTasksCreatedByUser(EntUser user){  
 return rTasks.findAllByCreatedby(user);  
 }  
  
 public EntTasks getTaskById(Long id) throws EntityException {  
 return rTasks.findById(id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "task not found when getting by id "+id,  
 "Задание не найдено",  
 EntTasks.class  
 ));  
 }  
  
 @Transactional  
 public EntTasks addTask(EntTasks task, String groups, String subjectname, Long user\_id, MultipartFile[] files) throws EntityException {  
 EntUser user=rUser.findById(user\_id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user with id "+user\_id+" not found when adding task",  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 EntSubject subject=rSubject.findByName(subjectname).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "subject "+subjectname+" not found when adding task",  
 "Предмет не найден",  
 EntSubject.class  
 ));  
 List<EntGroup> groupList = Arrays.*stream*(groups.split(","))  
 .map(group -> {  
 try {  
 return rGroup.findByName(group)  
 .orElseThrow(() -> new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "group " + group + " not found when adding task",  
 "Группа не найдена",  
 EntGroup.class  
 ));  
 } catch (EntityException e) {  
 throw new RuntimeException(e);  
 }  
 })  
 .toList();  
 task.setCreatedby(user);  
 task.setTasksubjectId(subject);  
 task.setGroups(groupList);  
 rTasks.save(task);  
 if(files!=null){  
 srvTasksfiles.addFiles(files,task,user);  
 }  
 return task;  
 }  
  
 @Transactional  
 public EntTasks deleteTaskById(Long id, UserDetails userDetails) throws AccessException, EntityException {  
 EntTasks task=rTasks.findById(Objects.*requireNonNull*(id)).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "task not found when deleting by id "+id,  
 "Задание не найдено",  
 EntTasks.class  
 ));  
 if(!Checker.*checkUserIdentity*(userDetails,task.getCreatedby(),rUser)){  
 throw new AccessException(HttpStatus.*FORBIDDEN*,"user identity check failed","Ошибка доступа",userDetails);  
 }  
 FilesProcessor.*deleteDirectory*(task.getTaskfiles\_list().getFirst().getPath().substring(0,task.getTaskfiles\_list().getFirst().getPath().lastIndexOf('/')));  
 rTasks.delete(task);  
 return task;  
 }  
  
 @Transactional  
 public EntTasks updateTask(EntTasks newtask, Long taskId, String subjectname, String groups, MultipartFile[] files, UserDetails userDetails) throws EntityException, AccessException {  
 EntTasks task=rTasks.findById(taskId).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "task not found when updating by id "+taskId,  
 "Задание не найдено",  
 EntTasks.class  
 ));  
 if(!Checker.*checkUserIdentity*(userDetails,task.getCreatedby(),rUser)){  
 throw new AccessException(HttpStatus.*FORBIDDEN*,"user identity check failed","Ошибка доступа",userDetails);  
 }  
 if(!task.getTasksubjectId().getName().equals(Objects.*requireNonNull*(subjectname))){  
 EntSubject subject=rSubject.findByName(subjectname).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "subject "+subjectname+" not found when adding task",  
 "Предмет не найден",  
 EntSubject.class  
 ));  
 task.setTasksubjectId(subject);  
 }  
 if(groups!=null) {  
 List<String> newgroups = Arrays.*stream*(groups.split(","))  
 .map(String::trim)  
 .filter(s -> !s.isEmpty())  
 .toList();  
 List<String> oldgroups = task.getGroups().stream()  
 .map(EntGroup::getName)  
 .toList();  
 if (!new HashSet<>(oldgroups).equals(new HashSet<>(newgroups))) {  
 List<EntGroup> groupList = new ArrayList<>();  
 for (String newgroup : newgroups) {  
 EntGroup group = rGroup.findByName(newgroup).orElseThrow(() -> new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "group " + newgroup + " not found when adding task",  
 "Группа не найдена",  
 EntGroup.class  
 ));  
 groupList.add(group);  
 }  
 task.setGroups(groupList);  
 }  
 }  
 task.setDateend(newtask.getDateend());  
 task.setText(newtask.getText());  
 task.setName(newtask.getName());  
 if(files!=null){  
 srvTasksfiles.addFiles(files,task,task.getCreatedby());  
 }  
 rTasks.save(task);  
 return task;  
 }  
}

Листинг 18 является листингом сервиса выполненных заданий, реализующим бизнес-логику для работы с выполненными заданиями

Листинг 18 – сервис выполненных заданий

@Service  
public class ServiceTasksCompleted {  
 private final RepTasksCompleted rTasksCompleted;  
 private final RepTasks rTasks;  
 private final RepUser rUser;  
 private final RepJournal rJournal;  
 private final ServiceCompletedtasksfiles srvCompletedtasksfiles;  
  
 public ServiceTasksCompleted(RepTasksCompleted rTasksCompleted, RepTasks rTasks, RepUser rUser, RepJournal rJournal, ServiceCompletedtasksfiles srvCompletedtasksfiles) {  
 this.rTasksCompleted = rTasksCompleted;  
 this.rTasks = rTasks;  
 this.rUser = rUser;  
 this.rJournal = rJournal;  
 this.srvCompletedtasksfiles = srvCompletedtasksfiles;  
 }  
  
 public List<EntTasksCompleted> getTasksCompleted(){  
 return rTasksCompleted.findAll();  
 }  
  
 public List<EntTasksCompleted> getTasksCompletedByTask(EntTasks task){  
 return rTasksCompleted.findAllByTasksId(task);  
 }  
  
 public List<EntUser> getStudentsFromCompletedTasksList(EntTasks task){  
 List<EntTasksCompleted> completedTasksList=getTasksCompletedByTask(task);  
 List<EntUser> studentList=new ArrayList<>();  
 for(EntTasksCompleted completedTask:completedTasksList){  
 studentList.add(completedTask.getUserId());  
 }  
 return studentList;  
 }  
  
 public EntTasksCompleted getCompletedTaskByUserIdAndTaskId(Long userId, Long taskId){  
 try {  
 EntUser user = rUser.findById(userId).orElseThrow(() -> new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user not found when getting completed task by user id " + userId,  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 EntTasks task = rTasks.findById(taskId).orElseThrow(() -> new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "task not found when getting completed by task id " + taskId,  
 "Задание не найдено",  
 EntTasks.class  
 ));  
 return rTasksCompleted.findByUserIdAndTasksId(user, task).orElseThrow(() -> new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "task not found when getting completed by task id " + taskId + " and user id " + userId,  
 "Выполненное задание не найдено",  
 EntTasksCompleted.class  
 ));  
 }catch (RuntimeException | EntityException e){  
 return null;  
 }  
 }  
  
 @Transactional  
 public EntTasksCompleted addTask(EntTasksCompleted completedtask, Long task\_id, Long user\_id, MultipartFile[] files, UserDetails userDetails) throws EntityException, AccessException {  
 EntTasks task=rTasks.findById(task\_id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "task not found when adding new completed by id "+task\_id,  
 "Задание не найдено",  
 EntTasks.class  
 ));  
 EntUser user=rUser.findById(user\_id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user not found when adding new completed by id "+user\_id,  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 Optional<EntTasksCompleted> task\_in\_db=rTasksCompleted.findByUserIdAndTasksId(user,task);  
 if(task\_in\_db.isPresent()){  
 completedtask.setId(task\_in\_db.get().getId());  
 return updateTaskByUser(completedtask,files,userDetails);  
 }  
 completedtask.setTasksId(task);  
 completedtask.setUserId(user);  
 completedtask=rTasksCompleted.save(completedtask);  
 if(files!=null){  
 srvCompletedtasksfiles.addFiles(files,completedtask,user\_id);  
 }  
 return completedtask;  
 }  
  
 @Transactional  
 public EntTasksCompleted delTaskById(Long id, UserDetails userDetails) throws AccessException, EntityException {  
 EntTasksCompleted task=rTasksCompleted.findById(id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "completed task not found when deleting by id "+id,  
 "Выполненное задание не найдено",  
 EntTasksCompleted.class  
 ));  
 if(!Checker.*checkUserIdentity*(userDetails,task.getTasksId().getCreatedby(),rUser)){  
 throw new AccessException(HttpStatus.*FORBIDDEN*,"user identity check failed","Ошибка доступа",userDetails);  
 }  
 rTasksCompleted.delete(task);  
 return task;  
 }  
  
 @Transactional  
 public EntTasksCompleted updateTaskByUser(EntTasksCompleted newtask, MultipartFile[] files, UserDetails userDetails) throws EntityException, AccessException {  
 EntTasksCompleted task=rTasksCompleted.findById(newtask.getId()).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "task not found when updating by id "+newtask.getId(),  
 "Выполненное задание не найдено",  
 EntTasksCompleted.class  
 ));  
 if(!Checker.*checkUserIdentity*(userDetails,task.getTasksId().getCreatedby(),rUser)){  
 throw new AccessException(HttpStatus.*FORBIDDEN*,"user identity check failed","Ошибка доступа",userDetails);  
 }  
 task.setCommentary(newtask.getCommentary());  
 task.setDateofsubmit(newtask.getDateofsubmit());  
 if(files!=null){  
 srvCompletedtasksfiles.addFiles(files,newtask,newtask.getUserId().getId());  
 }  
 rTasksCompleted.save(task);  
 return task;  
 }  
  
 @Transactional  
 public EntTasksCompleted updateTaskByCheck(EntTasksCompleted task, UserDetails userDetails) throws AccessException, EntityException {  
 EntTasksCompleted originaltask=rTasksCompleted.findById(task.getId()).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "task not found when checking and updating by id "+task.getId(),  
 "Выполненное задание не найдено",  
 EntTasksCompleted.class  
 ));  
 if(!Checker.*checkUserIdentity*(userDetails,originaltask.getTasksId().getCreatedby(),rUser)){  
 throw new AccessException(HttpStatus.*FORBIDDEN*,"user identity check failed","Ошибка доступа",userDetails);  
 }  
 originaltask.setFeedback(task.getFeedback());  
 originaltask.setDateofcheck(task.getDateofcheck());  
 originaltask.setGrade(task.getGrade());  
 rTasksCompleted.save(originaltask);  
 Optional<EntJournal> journal\_in\_db=rJournal.findByJournaltasksCompletedId(originaltask);  
 EntJournal journal;  
 DateTimeFormatter inputFormat = DateTimeFormatter.*ofPattern*("yyyy-MM-d HH:mm");  
 DateTimeFormatter outputFormat = DateTimeFormatter.*ofPattern*("yyyy-MM-dd");  
 LocalDateTime dateTime = LocalDateTime.*parse*(originaltask.getDateofcheck(), inputFormat);  
 String formattedDate = dateTime.format(outputFormat);  
 if(journal\_in\_db.isPresent()){  
 journal=journal\_in\_db.get();  
 journal.setGrade(originaltask.getGrade());  
 journal.setDate(formattedDate);  
 }else{  
 journal =new EntJournal(null, originaltask.getUserId(), originaltask.getTasksId().getTasksubjectId(),formattedDate, originaltask.getGrade(), null,originaltask);  
 }  
 rJournal.save(journal);  
 return originaltask;  
 }  
  
 @Transactional  
 public EntTasksCompleted updateTaskByCheck(EntTasksCompleted newtask, Long user\_id, Long task\_id, UserDetails userDetails) throws EntityException, AccessException {  
 EntTasksCompleted originaltask=rTasksCompleted.findById(newtask.getId()).orElse(  
 addTask(new EntTasksCompleted(null,null,null,null,null,null,null,null),task\_id,user\_id,null,userDetails)  
 );  
 originaltask.setFeedback(newtask.getFeedback());  
 originaltask.setDateofcheck(newtask.getDateofcheck());  
 originaltask.setGrade(newtask.getGrade());  
 rTasksCompleted.save(originaltask);  
 return originaltask;  
 }  
}

Листинг 19 является листингом сервиса электронного журнала, реализующим бизнес-логику для работы с электронным журналом

Листинг 19 – сервис электронного журнала

@Service  
public class ServiceJournal {  
 private final RepJournal rJournal;  
 private final RepUser rUser;  
 private final RepSubject rSubject;  
 private final RepGroup rGroup;  
  
 @Autowired  
 public ServiceJournal(RepJournal rJournal, RepUser rUser, RepSubject rSubject, RepGroup rGroup) {  
 this.rJournal = rJournal;  
 this.rUser = rUser;  
 this.rSubject = rSubject;  
 this.rGroup = rGroup;  
 }  
  
 public List<EntJournal> getAllJournal(){  
 return (List<EntJournal>) rJournal.findAll();  
 }  
  
 public List<EntJournal> getAllJournalForUserByName(String name) throws EntityException {  
 EntUser user=rUser.findByFirstname(name).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user not found when getting journal by user name "+name,  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 return rJournal.findAllByJournaluserId(user);  
 }  
  
 public EntJournal getJournalByCompletedTaskId(EntTasksCompleted completedTaskId) throws EntityException {  
 return rJournal.findByJournaltasksCompletedId(completedTaskId).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "journal not found by completedtask id "+ (completedTaskId!=null ? completedTaskId.getId() : "undefined"),  
 "Запись журнала не найдена",  
 EntJournal.class  
 ));  
 }  
  
 @Transactional  
 public EntJournal addJournalUser(EntJournal newjournal, Integer subject\_id, Long user\_id) throws EntityException {  
 EntUser user=rUser.findById(user\_id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user not found when adding journal to user "+user\_id,  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 EntSubject subject=rSubject.findById(subject\_id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "subject not found when adding journal with subject "+subject\_id,  
 "Предмет не найден",  
 EntSubject.class  
 ));  
 newjournal.setJournaluserId(user);  
 newjournal.setJournalsubjectId(subject);  
 rJournal.save(newjournal);  
 return newjournal;  
 }  
  
 @Transactional  
 public List<EntJournal> addJournalGroup(EntJournal newjournal, Integer subject\_id, Long group\_id) throws EntityException {  
 List<EntJournal> journalret=new ArrayList<>();  
 EntGroup group=rGroup.findById(group\_id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "group not found when adding journal to group with id "+group\_id,  
 "Группа не найдена",  
 EntGroup.class  
 ));  
 EntSubject subject=rSubject.findById(subject\_id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "subject not found when adding journal with subject "+subject\_id,  
 "Предмет не найден",  
 EntSubject.class  
 ));  
 List<EntUser> userlist=rUser.findAllByUsergroupId(group);  
 if(userlist==null){  
 throw new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "users in group not found when adding journal to group with id "+group\_id,  
 "Пользователи в группе не найдены",  
 EntUser.class  
 );  
 }  
 if(userlist.isEmpty()){  
 return null;  
 }  
 for(EntUser user:userlist){  
 EntJournal journal=new EntJournal(null,user,subject,newjournal.getDate(),newjournal.getGrade(),newjournal.getWaspresent());  
 rJournal.save(journal);  
 journalret.add(journal);  
 }  
 return journalret;  
 }  
  
 @Transactional  
 public EntJournal delJournalUserById(Long id, UserDetails userDetails) throws AccessException, EntityException {  
 EntJournal journal=rJournal.findById(id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "journal not found when deleting by id "+id,  
 "Запись в журнале не найдена",  
 EntJournal.class  
 ));  
 if(!Checker.*checkUserIdentity*(userDetails, journal.getJournaluserId(), rUser)){  
 throw new AccessException(HttpStatus.*FORBIDDEN*,"user identity check failed","Ошибка доступа",userDetails);  
 }  
 rJournal.delete(journal);  
 return journal;  
 }  
  
 @Transactional  
 public EntJournal updJournalUser(EntJournal newjournal, Integer subject\_id, Long user\_id, Long id, UserDetails userDetails) throws EntityException, AccessException {  
 EntJournal journal=rJournal.findById(id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "journal not found when updating by id "+id,  
 "Запись в журнале не найдена",  
 EntJournal.class  
 ));  
 EntUser user=rUser.findById(user\_id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user not found when adding journal to user "+user\_id,  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 if(!Checker.*checkUserIdentity*(userDetails,journal.getJournaluserId(),rUser)){  
 throw new AccessException(HttpStatus.*FORBIDDEN*,"user identity check failed","Ошибка доступа",userDetails);  
 }  
 EntSubject subject=rSubject.findById(subject\_id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "subject not found when adding journal with subject "+subject\_id,  
 "Предмет не найден",  
 EntSubject.class  
 ));  
 if(newjournal.getGrade()!=null && newjournal.getGrade()>=0) {  
 journal.setGrade(newjournal.getGrade());  
 }else{  
 journal.setGrade(null);  
 }  
 journal.setDate(newjournal.getDate());  
 journal.setWaspresent(newjournal.getWaspresent());  
 journal.setJournalsubjectId(subject);  
 journal.setJournaluserId(user);  
 rJournal.save(journal);  
 return journal;  
 }  
}

Листинг 20 является листингом сервиса конференций, реализующим бизнес-логику для работы с конференций

Листинг 20 – сервис конференций

@Service  
public class ServiceConference {  
 private final RepConferences rConferences;  
 private final RepUser rUser;  
 private final RepSubject rSubject;  
 private final RepGroup rGroup;  
  
 @Autowired  
 public ServiceConference(RepConferences rConferences, RepUser rUser, RepSubject rSubject, RepGroup rGroup) {  
 this.rConferences = rConferences;  
 this.rUser = rUser;  
 this.rSubject = rSubject;  
 this.rGroup = rGroup;  
 }  
  
 public List<EntConferences> getConferences(){  
 return rConferences.findAll();  
 }  
  
 public EntConferences getConferenceById(Long id) throws EntityException {  
 return rConferences.findById(id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "conference with id "+id+" not found",  
 "Конференция не найдена",  
 EntConferences.class  
 ));  
 }  
  
 public List<EntConferences> getConferencesByUserGroup(UserDetails userDetails, EntUser user) throws AccessException {  
 if(!Checker.*checkUserIdentity*(userDetails,user,rUser)){  
 throw new AccessException(HttpStatus.*UNAUTHORIZED*,"user identity check failed","Ошибка проверки данных аккаунта",userDetails);  
 }  
 return rConferences.findAllByGroupId(user.getUsergroupId());  
 }  
  
 public List<EntConferences> getConferencesCreatedByUser(UserDetails userDetails, EntUser user) throws AccessException {  
 if(!Checker.*checkUserIdentity*(userDetails,user,rUser)){  
 throw new AccessException(HttpStatus.*UNAUTHORIZED*,"user identity check failed","Ошибка проверки данных аккаунта",userDetails);  
 }  
 return rConferences.findAllByCreatedby(user);  
 }  
  
 @Transactional  
 public EntConferences addConference(EntConferences conference, String groups, String subjectname, UserDetails userDetails) throws EntityException {  
 EntUser user=rUser.findByLogin(Objects.*requireNonNull*(userDetails).getUsername()).orElseThrow(()->  
 new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user with login "+ userDetails.getUsername() +" not found when adding conference",  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 conference.setCreatedby(user);  
 EntSubject subject=rSubject.findByName(subjectname).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "subject "+subjectname+" not found when adding task",  
 "Предмет не найден",  
 EntSubject.class  
 ));  
 conference.setSubjectId(subject);  
 List<EntGroup> groupList = Arrays.*stream*(groups.split(","))  
 .map(group -> {  
 try {  
 return rGroup.findByName(group)  
 .orElseThrow(() -> new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "group " + group + " not found when adding task",  
 "Группы не найдены",  
 EntGroup.class  
 ));  
 } catch (EntityException e) {  
 throw new RuntimeException(e);  
 }  
 })  
 .toList();  
 conference.setGroupId(groupList);  
 rConferences.save(conference);  
 return conference;  
 }  
  
 @Transactional  
 public EntConferences updateConferenceById(EntConferences newconference, Long conferenceId, String groups, String subjectname, UserDetails userDetails) throws AccessException, EntityException {  
 EntConferences conference=rConferences.findById(conferenceId).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "conference with id "+conferenceId+" not found when updating",  
 "Конференция не найдена",  
 EntConferences.class  
 ));  
 if(!Checker.*checkUserIdentity*(userDetails,conference.getCreatedby(),rUser)){  
 throw new AccessException(HttpStatus.*UNAUTHORIZED*,"user identity check failed","Ошибка проверки данных аккаунта",userDetails);  
 }  
 if(!conference.getSubjectId().getName().equals(Objects.*requireNonNull*(subjectname))){  
 EntSubject subject=rSubject.findByName(subjectname).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "subject "+subjectname+" not found when updating conference",  
 "Предмет не найден",  
 EntSubject.class  
 ));  
 conference.setSubjectId(subject);  
 }  
 if(groups!=null) {  
 List<String> newgroups = Arrays.*stream*(groups.split(","))  
 .map(String::trim)  
 .filter(s -> !s.isEmpty())  
 .toList();  
 List<String> oldgroups = conference.getGroupId().stream()  
 .map(EntGroup::getName)  
 .toList();  
 if (!new HashSet<>(oldgroups).equals(new HashSet<>(newgroups))) {  
 List<EntGroup> groupList = new ArrayList<>();  
 for (String newgroup : newgroups) {  
 EntGroup group = rGroup.findByName(newgroup).orElseThrow(() -> new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "group " + newgroup + " not found when updating conference",  
 "Группа не найдена",  
 EntGroup.class  
 ));  
 groupList.add(group);  
 }  
 conference.setGroupId(groupList);  
 }  
 }  
 conference.setDatestart(newconference.getDatestart());  
 conference.setName(newconference.getName());  
 conference.setRepeatable(newconference.getRepeatable());  
 rConferences.save(conference);  
 return conference;  
 }  
  
 @Transactional  
 public EntConferences deleteConferenceById(Long id, UserDetails userDetails) throws AccessException, EntityException {  
 EntConferences conference=rConferences.findById(id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "conference not found when deleting by id "+id,  
 "Конференция не найдена",  
 EntConferences.class  
 ));  
 if(!Checker.*checkUserIdentity*(userDetails,conference.getCreatedby(),rUser)){  
 throw new AccessException(HttpStatus.*UNAUTHORIZED*,"user identity check failed","Ошибка проверки данных аккаунта",userDetails);  
 }  
 rConferences.delete(conference);  
 return conference;  
 }  
}

Листинг 21 является листингом сервиса активных конференций, реализующим бизнес-логику для работы с активными конференциями

Листинг 21 – сервис активных конференций

@Service  
public class ServiceVideocalls {  
 private final RepVideocalls rVideocalls;  
 private final RepUser rUser;  
 private final RepConferences rConferences;  
 private final RepVideocallsHasUser rVideocallsHasUser;  
 private final RepVideocallsHasUserProperties rVideocallsHasUserProperties;  
 private final ServiceVideocallsChat srvVideocallChat;  
 private final ServiceVideocallsJanusAPI srvVideocallsJanusAPI;  
 private final ConfUrls appurls;  
 public enum UpdateActions {*AUDIO*, *VIDEO*, *DEMONSTRATION*,*REACTION*,*BAN*,*SOUND*}  
 public enum LeaveReasons {*EXIT*,*RELOAD*}  
  
 @Autowired  
 public ServiceVideocalls(RepVideocalls rVideocalls, RepUser rUser, RepConferences rConferences, RepVideocallsHasUser rVideocallsHasUser, RepUserfiles rUserfiles, RepVideocallsHasUserProperties rVideocallsHasUserProperties, RepVideocallChat rVideocallChat, ServiceVideocallsChat srvVideocallChat, ServiceVideocallsJanusAPI srvVideocallsJanusAPI, ConfUrls appurls) {  
 this.rVideocalls = rVideocalls;  
 this.rUser = rUser;  
 this.rConferences = rConferences;  
 this.rVideocallsHasUser = rVideocallsHasUser;  
 this.rVideocallsHasUserProperties = rVideocallsHasUserProperties;  
 this.srvVideocallChat = srvVideocallChat;  
 this.srvVideocallsJanusAPI = srvVideocallsJanusAPI;  
 this.appurls = appurls;  
 }  
  
 public List<EntVideocalls> getVideocalls(){  
 return rVideocalls.findAll();  
 }  
  
 public EntVideocalls getVideocallById(Long id) throws EntityException {  
 return rVideocalls.findById(id).orElseThrow(()->new EntityException(HttpStatus.*NOT\_FOUND*,"videocall not found by id: " + id,"Видеозвонок не найден", EntVideocalls.class));  
 }  
  
 public List<EntVideocalls> getVideocallsByConferencesCreator(EntUser user, UserDetails userDetails) throws AccessException {  
 if(!Checker.*checkUserIdentity*(userDetails,user,rUser)){  
 throw new AccessException(HttpStatus.*FORBIDDEN*,"user identity check failed","Ошибка доступа",userDetails);  
 }  
 return rVideocalls.findAllByConferencesId\_Createdby(user);  
 }  
  
 public List<EntVideocalls> getVideocallsByGroup(EntUser user){  
 return rVideocalls.findAllByConferencesId\_GroupId(user.getUsergroupId());  
 }  
  
 public EntVideocallsHasUser getVideocallsHasUserByUserDetailsAndVideocallId(UserDetails userDetails, Long videocallId) throws EntityException, AccessException {  
 if(userDetails==null){  
 throw new AccessException(HttpStatus.*BAD\_REQUEST*,"user details was null","Ошибка данных вашего аккаунта", null);  
 }  
 EntUser user=rUser.findByLogin(userDetails.getUsername()).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,"user not found when getting videocallsHasUser",  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 EntVideocalls videocall=rVideocalls.findById(videocallId).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "videocall not found by id: " + videocallId,  
 "Видеозвонок не найден",  
 EntVideocalls.class  
 ));  
 return rVideocallsHasUser.findByVideocalluserId(user,videocall).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "videocallHasUser not found by user with username "+ userDetails.getUsername() +" and videocall id "+(videocall != null ? videocall.getId() : "undefined"),  
 "Участник конференции не найден",  
 EntVideocallsHasUser.class  
 ));  
 }  
  
 @Transactional(rollbackOn = {BaseException.class})  
 @Retryable(  
 retryFor = { PessimisticLockingFailureException.class, LockTimeoutException.class },  
 backoff = @Backoff(delay = 200),  
 maxAttempts = 5  
 )  
 public EntVideocallsHasUser joinVideocalls(Long videocallId, UserDetails userDetails) throws AccessException, EntityException, URISyntaxException, ExecutionException, JanusAPIException, InterruptedException {  
 if(userDetails==null){  
 throw new AccessException(HttpStatus.*BAD\_REQUEST*,"user details was null","Ошибка данных вашего аккаунта", null);  
 }  
 EntUser user =rUser.findByLogin(userDetails.getUsername()).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user "+userDetails.getUsername()+" not found when join to videocall with id "+videocallId,  
 "Данные вашего аккаунта не найдены",  
 EntUser.class  
 ));  
 EntVideocalls videocall=rVideocalls.findByIdwithLock(videocallId).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "videocall not found by id: "+videocallId,  
 "Видеозвонок не найден",  
 EntVideocalls.class  
 ));  
 Optional<EntVideocallsHasUser> videocalldataFromDb=rVideocallsHasUser.findByVideocalluserId(user,videocall);  
 EntVideocallsHasUser videocallhasuser;  
 if(videocalldataFromDb.isEmpty()) {  
 videocallhasuser=new EntVideocallsHasUser(null,videocall,user, EntVideocallsHasUser.defaultStates.*OFF*, EntVideocallsHasUser.defaultStates.*OFF*, EntVideocallsHasUser.defaultStates.*ON*, EntVideocallsHasUser.defaultStates.*OFF*,false, true);  
 rVideocallsHasUser.save(videocallhasuser);  
 }else {  
 videocallhasuser=videocalldataFromDb.get();  
 videocallhasuser.setConnected(true);  
 EntVideocallsHasUserProperties properties=rVideocallsHasUserProperties.findByvideocallHasUserId(videocallhasuser).orElse(null);  
 if(properties!=null && properties.isBanned()) {  
 throw new AccessException(  
 HttpStatus.*FORBIDDEN*,  
 "user with id "+(videocallhasuser.getVideocalluserId()!=null ? videocallhasuser.getVideocalluserId().getId() : "undefined")+  
 " trying to connect to videocall with id "+(videocallhasuser.getVideocallsId()!=null ? videocallhasuser.getVideocallsId().getId() : "undefined") + " where he was banned in videocall",  
 "Ошибка доступа: вы были забанены в этом звонке",  
 userDetails  
 );  
 }  
 }  
 rVideocallsHasUser.save(videocallhasuser);  
 videocall.setParticipants(videocall.getParticipants()+1);  
 if(!srvVideocallsJanusAPI.isWebsocketExist(videocall)) {  
 WebSocketJanus ws=srvVideocallsJanusAPI.connectToJanus(videocall);  
 srvVideocallsJanusAPI.saveWebsocket(ws,videocall);  
 }  
 rVideocalls.save(videocall);  
 return videocallhasuser;  
 }  
  
 @Transactional(rollbackOn = {BaseException.class, ExecutionException.class,InterruptedException.class})  
 @Retryable(  
 retryFor = { PessimisticLockingFailureException.class, LockTimeoutException.class },  
 backoff = @Backoff(delay = 200),  
 maxAttempts = 5  
 )  
 public EntVideocallsHasUser leaveVideocalls(Long videocallId, LeaveReasons reason, UserDetails userDetails) throws AccessException, ExecutionException, InterruptedException, EntityException, JanusAPIException {  
 if(userDetails==null){  
 throw new AccessException(HttpStatus.*BAD\_REQUEST*,"user details was null","Ошибка данных вашего аккаунта", null);  
 }  
 EntUser user =rUser.findByLogin(userDetails.getUsername()).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user "+userDetails.getUsername()+" not found when join to videocall with id "+videocallId,  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 EntVideocalls activecall=rVideocalls.findByIdwithLock(videocallId).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "videocall not found by id "+videocallId,  
 "Видеозвонок не найден",  
 EntVideocalls.class  
 ));  
 EntVideocallsHasUser videocallhasuser=rVideocallsHasUser.findByVideocalluserId(user,activecall).orElseThrow(()-> new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "videocallsHasUser not found by videocall id " +(activecall!=null ? activecall.getId() : "undefined") +" and user id "+(user!=null ? user.getId() : "undefined"),  
 "Участник конференции не найден",  
 EntVideocallsHasUser.class  
 ));  
 if(!videocallhasuser.getConnected()) {  
 throw new AccessException(HttpStatus.*FORBIDDEN*,"user with id "+videocallhasuser.getVideocalluserId().getId()+ " trying to disconnect to videocall with id "+videocallhasuser.getVideocallsId().getId()  
 +" where he was banned in videocall","Пользователь не подключен к видеоконференции",userDetails);  
 }  
 videocallhasuser.setConnected(false);  
 rVideocallsHasUser.save(videocallhasuser);  
 int participans=activecall.getParticipants()-1;  
 activecall.setParticipants(participans);  
 if(reason!=LeaveReasons.*RELOAD*) {  
 if (participans <= 0) {  
 rVideocallsHasUser.deleteAllByVideocallsId(activecall);  
 rVideocalls.delete(activecall);  
 srvVideocallsJanusAPI.deleteRoom(activecall);  
 } else {  
 rVideocalls.save(activecall);  
 }  
 }  
 return videocallhasuser;  
 }  
  
 @Transactional(rollbackOn = {BaseException.class, ExecutionException.class,InterruptedException.class})  
 @Retryable(  
 retryFor = { PessimisticLockingFailureException.class, LockTimeoutException.class },  
 backoff = @Backoff(delay = 200)  
 )  
 public EntVideocalls addVideocall(Long conferenceId) throws URISyntaxException, ExecutionException, InterruptedException, EntityException, JanusAPIException {  
 EntConferences conference=rConferences.findByIdwithLock(conferenceId).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "conference with id "+conferenceId+" not found when adding videocall",  
 "Конференция не найдена",  
 EntConferences.class  
 ));  
 Optional<EntVideocalls> activecall=rVideocalls.findByConferencesId(conference);  
 if(activecall.isEmpty()) {  
 EntVideocalls videocall = new EntVideocalls(null, conference, null, 0);  
 WebSocketJanus ws=srvVideocallsJanusAPI.connectToJanus(videocall);  
 rVideocalls.save(videocall);  
 srvVideocallsJanusAPI.saveWebsocket(ws,videocall);  
 return videocall;  
 }  
 return activecall.get();  
 }  
  
 public EntVideocallsHasUser updateVideocallHasUser(EntVideocallsHasUser newvideocallsHasUser) throws EntityException, DataProcessingException {  
 if(newvideocallsHasUser==null){  
 throw new DataProcessingException(HttpStatus.*BAD\_REQUEST*,"newvideocallsHasUser was null","Ошибка запроса: данные не могут быть null");  
 }  
 EntVideocallsHasUser videocallsHasUser=rVideocallsHasUser.findById(newvideocallsHasUser.getId()).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "videocall\_has\_user not found by id "+newvideocallsHasUser.getId(),  
 "Участник конференции не найден",  
 EntVideocallsHasUser.class  
 ));  
 videocallsHasUser.setVideocallsId(newvideocallsHasUser.getVideocallsId());  
 videocallsHasUser.setSignalstate(newvideocallsHasUser.getSignalstate());  
 videocallsHasUser.setConnected(newvideocallsHasUser.getConnected());  
 videocallsHasUser.setDemostate(newvideocallsHasUser.getDemostate());  
 videocallsHasUser.setSoundstate(newvideocallsHasUser.getSoundstate());  
 videocallsHasUser.setCamstate(newvideocallsHasUser.getCamstate());  
 videocallsHasUser.setMicrostate(newvideocallsHasUser.getMicrostate());  
 videocallsHasUser.setVideocalluserId(newvideocallsHasUser.getVideocalluserId());  
 rVideocallsHasUser.save(videocallsHasUser);  
 return videocallsHasUser;  
 }  
  
 public EntVideocalls deleteVideocallById(Long id) throws EntityException {  
 EntVideocalls videocall=rVideocalls.findById(id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "videocall not found when deleting by id "+id,  
 "Видеозвонок не найден",  
 EntVideocalls.class  
 ));  
 rVideocalls.delete(videocall);  
 return videocall;  
 }  
  
 @Transactional  
 public DTOVideocallUpdate updateUserSelfByAction(Long videocallId, UpdateActions action, EntVideocallsHasUser.defaultStates state, UserDetails userDetails) throws AccessException, EntityException {  
 if(userDetails==null){  
 throw new AccessException(HttpStatus.*BAD\_REQUEST*,"user details was null","Ошибка данных вашего аккаунта", null);  
 }  
 EntUser user=rUser.findByLogin(Objects.*requireNonNull*(userDetails.getUsername())).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user with username "+userDetails.getUsername()+" not found",  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 EntVideocalls videocall=rVideocalls.findById(videocallId).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "videocall not found by id "+videocallId+" when updating videocallHasUser by action",  
 "Видеозвонок не найден",  
 EntVideocalls.class  
 ));  
 EntVideocallsHasUser videocallsHasUser=rVideocallsHasUser.findByVideocalluserIdwithLock(user,videocall).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "videocall has user not found by videocall with id "+(videocall!=null ? videocall.getId() : "undefined")+" and user with id "+(user!=null ? user.getId(): "undefined"),  
 "Участник конференции не найден",  
 EntVideocallsHasUser.class  
 ));  
 EntVideocallsHasUserProperties properties=rVideocallsHasUserProperties.findByvideocallHasUserId(videocallsHasUser).orElse(null);  
 if(!videocallsHasUser.getConnected()){  
 throw new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user with id "+(videocallsHasUser.getVideocalluserId()!=null ? videocallsHasUser.getVideocalluserId().getId(): "undefined")+" not connected to videocall with id "+  
 (videocallsHasUser.getVideocallsId()!=null ? videocallsHasUser.getVideocallsId() : "undefined"),  
 "Пользователь не подключён к конференции",  
 EntVideocallsHasUser.class  
 );  
 }  
 boolean updated=false;  
 EntVideocallsHasUser.defaultStates newstate= EntVideocallsHasUser.defaultStates.*OFF*;  
 switch (action){  
 case *AUDIO* -> {  
 if(videocallsHasUser.getMicrostate()== EntVideocallsHasUser.defaultStates.*MUTED\_BY\_ADMIN*) {  
 if(properties!=null && !Checker.*isUserMorePowerful*(videocallsHasUser.getVideocalluserId(), properties.getMicromuted())){  
 throw new AccessException(HttpStatus.*BAD\_REQUEST*,"user with id "+videocallsHasUser.getVideocalluserId().getId()+ " trying to update his devices in videocall with id "+  
 videocallsHasUser.getVideocallsId().getId()  
 +" without enough role power","Ошибка доступа: недостаточно прав",userDetails);  
 }  
 }  
 updated= videocallsHasUser.getMicrostate()!=state;  
 videocallsHasUser.setMicrostate(state);  
 newstate=videocallsHasUser.getMicrostate();  
 }  
 case *VIDEO* -> {  
 if(videocallsHasUser.getCamstate()== EntVideocallsHasUser.defaultStates.*MUTED\_BY\_ADMIN*) {  
 if(properties!=null && !Checker.*isUserMorePowerful*(videocallsHasUser.getVideocalluserId(), properties.getCameramuted())){  
 throw new AccessException(HttpStatus.*BAD\_REQUEST*,"user with id "+videocallsHasUser.getVideocalluserId().getId()+ " trying to update his "+action+" in videocall with id "+  
 videocallsHasUser.getVideocallsId().getId()  
 +" without enough role power","Ошибка доступа: недостаточно прав",userDetails);  
 }  
 }  
 updated= videocallsHasUser.getCamstate()!=state;  
 videocallsHasUser.setCamstate(state);  
 newstate=videocallsHasUser.getCamstate();  
 }  
 case *DEMONSTRATION* -> {  
 if(videocallsHasUser.getDemostate()== EntVideocallsHasUser.defaultStates.*MUTED\_BY\_ADMIN*) {  
 if(properties!=null && !Checker.*isUserMorePowerful*(videocallsHasUser.getVideocalluserId(), properties.getDemomuted())){  
 throw new AccessException(HttpStatus.*BAD\_REQUEST*,"user with id "+videocallsHasUser.getVideocalluserId().getId()+ " trying to update his "+action+" in videocall with id "+  
 videocallsHasUser.getVideocallsId().getId()  
 +" without enough role power","Ошибка доступа: недостаточно прав",userDetails);  
 }  
 }  
 updated= true;  
 videocallsHasUser.setDemostate(  
 (videocallsHasUser.getDemostate()== EntVideocallsHasUser.defaultStates.*ON*) ? EntVideocallsHasUser.defaultStates.*OFF* : EntVideocallsHasUser.defaultStates.*ON* );  
 newstate=videocallsHasUser.getDemostate();  
 }  
 case *REACTION* -> {  
 //updated= videocallsHasUser.getSignalstate()!=state;  
 // videocallsHasUser.setSignalstate(state);  
 }  
 case *SOUND* -> {  
 if(videocallsHasUser.getSoundstate()== EntVideocallsHasUser.defaultStates.*MUTED\_BY\_ADMIN*) {  
 if(properties!=null && !Checker.*isUserMorePowerful*(videocallsHasUser.getVideocalluserId(), properties.getSoundmuted())){  
 throw new AccessException(HttpStatus.*BAD\_REQUEST*,"user with id "+videocallsHasUser.getVideocalluserId().getId()+ " trying to update his "+action+" in videocall with id "+  
 videocallsHasUser.getVideocallsId().getId()  
 +" without enough role power","Ошибка доступа: недостаточно прав",userDetails);  
 }  
 }  
 updated= videocallsHasUser.getSoundstate()!=state;  
 videocallsHasUser.setSoundstate(state);  
 newstate=videocallsHasUser.getSoundstate();  
 }  
 }  
 if(updated) {  
 rVideocallsHasUser.save(videocallsHasUser);  
 }  
 return new DTOVideocallUpdate(videocallsHasUser,null,newstate);  
 }  
  
 @Transactional  
 public DTOVideocallUpdate updateUserOtherByAction(Long videocallId, Long userId, UpdateActions action, EntVideocallsHasUser.defaultStates state, UserDetails userDetails) throws AccessException, EntityException, DataProcessingException {  
 if(userDetails==null){  
 throw new AccessException(HttpStatus.*BAD\_REQUEST*,"user details was null","Ошибка данных вашего аккаунта", null);  
 }  
 EntUser userUpdater=rUser.findByLogin(Objects.*requireNonNull*(userDetails.getUsername())).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user "+userDetails.getUsername()+" not found",  
 "Ваш аккаунт не найден",  
 EntUser.class  
 ));  
 EntUser userUpdated=rUser.findById(Objects.*requireNonNull*(userId)).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user "+userId+" not found",  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 EntVideocalls videocall=rVideocalls.findById(videocallId).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "videocall not found by id "+videocallId+" when updating videocallHasUser by action",  
 "Видеозвонок не найден",  
 EntVideocalls.class  
 ));  
 EntVideocallsHasUser videocallsHasUserUpdated=rVideocallsHasUser.findByVideocalluserIdwithLock(userUpdated,videocall).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "videocall has user not found by videocall with id "+(videocall!=null ? videocall.getId() : "undefined")+" and user with id "+(userUpdated!=null ? userUpdated.getId(): "undefined"),  
 "Участник конференции не найден",  
 EntVideocallsHasUser.class  
 ));  
 EntVideocallsHasUserProperties properties=rVideocallsHasUserProperties.findByvideocallHasUserId(videocallsHasUserUpdated).orElse(  
 new EntVideocallsHasUserProperties(null,videocallsHasUserUpdated,0,0,0,0,false)  
 );  
 if(!videocallsHasUserUpdated.getConnected()){  
 throw new AccessException(HttpStatus.*BAD\_REQUEST*,"user with id "+userUpdater.getId()+ " trying to update non-connected user with id "+  
 videocallsHasUserUpdated.getVideocalluserId().getId()+" in videocall with id "+videocallsHasUserUpdated.getVideocallsId().getId()  
 ,"Ошибка запроса: изменяемый пользователь не подключен к конференции",userDetails);  
 }  
 EntVideocallsHasUser.defaultStates newstate=null;  
 if(Checker.*isUserMorePowerful*(userUpdater,userUpdated,false)) {  
 boolean isBanned=false;  
 switch (action) {  
 case *AUDIO* -> {  
 if(state==null){  
 throw new DataProcessingException(HttpStatus.*BAD\_REQUEST*,"state is null","Ошибка запроса: state не может быть null для этого устройства");  
 }  
 if(state== EntVideocallsHasUser.defaultStates.*MUTED\_BY\_ADMIN*) {  
 videocallsHasUserUpdated.setMicrostate(EntVideocallsHasUser.defaultStates.*MUTED\_BY\_ADMIN*);  
 properties.setMicromuted(Parser.*parseUserRolesMaxPower*(userUpdater));  
 }else{  
 if(videocallsHasUserUpdated.getMicrostate()!= EntVideocallsHasUser.defaultStates.*MUTED\_BY\_ADMIN*){  
 throw new AccessException(HttpStatus.*BAD\_REQUEST*,"user with id "+userUpdater.getId()+ " trying to update non-muted user's "+action+" with id "+  
 videocallsHasUserUpdated.getVideocalluserId().getId()+  
 " in videocall with id "+videocallsHasUserUpdated.getVideocallsId().getId(),  
 "Ошибка запроса: изменяемый пользователь не был заглушен администратором",userDetails);  
 }  
 properties.setMicromuted(0);  
 videocallsHasUserUpdated.setMicrostate(EntVideocallsHasUser.defaultStates.*OFF*);  
 }  
 newstate=videocallsHasUserUpdated.getMicrostate();  
 }  
 case *VIDEO* -> {  
 if(state==null){  
 throw new DataProcessingException(HttpStatus.*BAD\_REQUEST*,"state is null","Ошибка запроса: state не может быть null для этого устройства");  
 }  
 if(state== EntVideocallsHasUser.defaultStates.*MUTED\_BY\_ADMIN*) {  
 videocallsHasUserUpdated.setCamstate(EntVideocallsHasUser.defaultStates.*MUTED\_BY\_ADMIN*);  
 properties.setCameramuted(Parser.*parseUserRolesMaxPower*(userUpdater));  
 }else {  
 if(videocallsHasUserUpdated.getCamstate()!= EntVideocallsHasUser.defaultStates.*MUTED\_BY\_ADMIN*){  
 throw new AccessException(HttpStatus.*BAD\_REQUEST*,"user with id "+userUpdater.getId()+ " trying to update non-muted user's "+action+" with id "+  
 videocallsHasUserUpdated.getVideocalluserId().getId()+  
 " in videocall with id "+videocallsHasUserUpdated.getVideocallsId().getId(),  
 "Ошибка запроса: изменяемый пользователь не был заглушен администратором",userDetails);  
 }  
 properties.setCameramuted(0);  
 videocallsHasUserUpdated.setCamstate(EntVideocallsHasUser.defaultStates.*OFF*);  
 }  
 newstate=videocallsHasUserUpdated.getCamstate();  
 }  
 case *DEMONSTRATION* -> {  
 if(state== EntVideocallsHasUser.defaultStates.*MUTED\_BY\_ADMIN*) {  
 videocallsHasUserUpdated.setDemostate(EntVideocallsHasUser.defaultStates.*MUTED\_BY\_ADMIN*);  
 properties.setDemomuted(Parser.*parseUserRolesMaxPower*(userUpdater));  
 }else {  
 videocallsHasUserUpdated.setDemostate(EntVideocallsHasUser.defaultStates.*OFF*);  
 properties.setDemomuted(0);  
 }  
 newstate=videocallsHasUserUpdated.getDemostate();  
 }  
 case *REACTION* -> {  
 }  
 case *SOUND* -> {  
 if(state== EntVideocallsHasUser.defaultStates.*MUTED\_BY\_ADMIN*) {  
 videocallsHasUserUpdated.setSoundstate(EntVideocallsHasUser.defaultStates.*MUTED\_BY\_ADMIN*);  
 properties.setSoundmuted(Parser.*parseUserRolesMaxPower*(userUpdater));  
 }else {  
 videocallsHasUserUpdated.setSoundstate(EntVideocallsHasUser.defaultStates.*OFF*);  
 properties.setSoundmuted(0);  
 }  
 newstate=videocallsHasUserUpdated.getSoundstate();  
 }  
 case *BAN* -> {  
 properties.setBanned(!properties.isBanned());  
 isBanned=true;  
 }  
 }  
 rVideocallsHasUser.save(videocallsHasUserUpdated);  
 rVideocallsHasUserProperties.save(properties);  
 return new DTOVideocallUpdate(videocallsHasUserUpdated,isBanned,newstate);  
 }else{  
 throw new AccessException(  
 HttpStatus.*FORBIDDEN*,"user with role power "+(userUpdater!=null ? Parser.*parseUserRolesMaxPower*(userUpdater) : "undefined") +  
 " tried to update user with role power "+(userUpdated!=null ? Parser.*parseUserRolesMaxPower*(userUpdated) : "undefined"),  
 "Ошибка доступа: недостаточно прав",  
 userDetails  
 );  
 }  
 }  
  
 public void updateUserConnectionStatus(Long userId,boolean connected){  
 rVideocallsHasUser.updateConnectionStatus(userId,connected);  
 }  
  
 private JSONObject generateRequestForUserUpdates(String action,boolean state,boolean janus){  
 JSONObject message=new JSONObject()  
 .put("request","configure")  
 .put(action,state);  
 JSONObject data=new JSONObject()  
 .put("message",message);  
 JSONObject request=new JSONObject()  
 .put("event","configure")  
 .put("type",janus ? "janus" : "other")  
 .put("data",data);  
 return request;  
 }  
}

Листинг 22 является листингом сервиса чатов, реализующим бизнес-логику для работы с чатами

Листинг 22 – сервис чатов

@Service  
public class ServiceChat {  
 private final RepChat rChat;  
 private final RepChatUser rChatUser;  
 private final RepSubject rSubject;  
 private final RepUser rUser;  
 private final RepGroup rGroup;  
  
 @Autowired  
 public ServiceChat(RepChat rChat, RepChatUser rChatUser, RepSubject rSubject, RepUser rUser, RepGroup rGroup) {  
 this.rChat = rChat;  
 this.rChatUser = rChatUser;  
 this.rSubject = rSubject;  
 this.rUser = rUser;  
 this.rGroup = rGroup;  
 }  
  
 public List<EntChat> getChatsAll(){  
 return rChat.findAll();  
 }  
  
 public List<DTOChatDisplay> getChatsDTO(UserDetails userDetails,int page) throws AccessException, EntityException {  
 if(userDetails==null){  
 throw new AccessException(  
 HttpStatus.*UNAUTHORIZED*,  
 "user details was null",  
 "Ошибка проверки данных аккаунта",  
 null  
 );  
 }  
 EntUser user=rUser.findByLogin(userDetails.getUsername()).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user with username "+userDetails.getUsername()+" not found",  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 Pageable pageable = PageRequest.*of*(page, 20);  
 List<EntChatUser> chats=rChatUser.findAllByUserId(user,pageable).getContent();  
 List<DTOChatDisplay> dtoChats=new ArrayList<>();  
 for(EntChatUser chatuser:chats){  
 EntChat chat=chatuser.getChatId();  
 List<EntUser> users = chat.getChatUsers().stream().map(EntChatUser::getUserId).toList();  
 dtoChats.add(new DTOChatDisplay(  
 chat,  
 users  
 ));  
 }  
 return dtoChats;  
 }  
  
 public List<DTOUserUpdate> getUsersInChat(EntChat entChat, int page){  
 Pageable pageable = PageRequest.*of*(page, 20);  
 Page<EntChatUser> chatusers=rChatUser.findAllByChatId(entChat,pageable);  
 List<EntUser> userList=chatusers.getContent().stream().map(EntChatUser::getUserId).toList();  
 List<DTOUserUpdate> users=new ArrayList<>();  
 for(EntUser user:userList){  
 users.add(new DTOUserUpdate(  
 user.getId(),  
 user.getLogin(),  
 user.getFirstname(),  
 user.getLastname(),  
 user.getSurname()  
 ));  
 }  
 return users;  
 }  
  
 public EntChat getChat(Long id) throws EntityException {  
 return rChat.findById(id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "chat with id "+id+" not found",  
 "Чат не найден",  
 EntChat.class  
 ));  
 }  
  
 @Transactional  
 public DTOChat addChat(DTOChat newChat, UserDetails userDetails) throws EntityException, AccessException {  
 EntSubject subject=rSubject.findByName(newChat.getSubjectName()).orElse(null);  
 EntGroup group=rGroup.findByName(newChat.getGroupName()).orElse(null);  
 if(userDetails==null){  
 throw new AccessException(  
 HttpStatus.*UNAUTHORIZED*,  
 "user details was null",  
 "Ошибка проверки данных аккаунта",  
 null  
 );  
 }  
 EntUser creator=rUser.findByLogin(userDetails.getUsername()).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user with username "+userDetails.getUsername()+" not found",  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 Set<EntUser> members=new HashSet<>(rUser.findAllByLoginIn(Arrays.*stream*(newChat.getMembers()).toList()));  
 members.add(creator);  
 if(group!=null){  
 members.addAll(group.getUsers\_list());  
 }  
 EntChat chat=new EntChat(null, newChat.getName(), subject, creator, group);  
 List<EntChatUser> chatUsers = members.stream()  
 .map(user -> new EntChatUser(null, chat, user))  
 .toList();  
 chat.setChatUsers(chatUsers);  
 rChat.save(chat);  
 DTOChat retChat= ConverterChatToChat.*convertEntityToDTO*(chat);  
 retChat.setMembersList(members.stream().map(user -> new DTOUserUpdate(  
 user.getId(),  
 user.getLogin(),  
 user.getFirstname(),  
 user.getLastname(),  
 user.getSurname()  
 )).toList());  
 return retChat;  
 }  
}

Листинг 23 является листингом сервиса сообщений чата, реализующим бизнес-логику для работы с сообщениями чата

Листинг 23 – сервис сообщений чата

@Service  
public class ServiceChatMessage {  
 private final RepChatMessage rChatMessage;  
 private final RepChat rChat;  
 private final RepUser rUser;  
 private final RepChatUser rChatUser;  
  
 @Autowired  
 public ServiceChatMessage(RepChatMessage rChatMessage, RepChat rChat, RepUser rUser, RepChatUser rChatUser) {  
 this.rChatMessage = rChatMessage;  
 this.rChat = rChat;  
 this.rUser = rUser;  
 this.rChatUser = rChatUser;  
 }  
  
 public List<DTOChatMessage> getMessagesInChat(Long chatId,int page,UserDetails userDetails) throws EntityException, AccessException {  
 if(userDetails==null){  
 throw new AccessException(  
 HttpStatus.*UNAUTHORIZED*,  
 "user details was null",  
 "Ошибка проверки данных аккаунта",  
 null  
 );  
 }  
 EntUser user=rUser.findByLogin(userDetails.getUsername()).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user "+userDetails.getUsername()+" not found",  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 EntChat chat=rChat.findById(chatId).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "chat with id "+chatId+" not found",  
 "Чат не найден",  
 EntChat.class  
 ));  
 EntChatUser chatUser=rChatUser.findByChatIdAndUserId(chat,user).orElseThrow(()->new EntityException(  
 HttpStatus.*FORBIDDEN*,  
 "user with id "+user.getId()+" isn't member of chat with id "+chat.getId(),  
 "Нет доступа к этому чату",  
 EntChatUser.class  
 ));  
 Pageable pageable = PageRequest.*of*(page, 40);  
 List<EntChatMessage> messages= rChatMessage.findAllByChatId(chat,pageable).getContent();  
 return messages.stream().map(ConverterChatMessageToChatMessage::*convertEntityToDTO*).collect(Collectors.*toList*());  
 }  
  
 @Transactional  
 public DTOChatMessage addMessageToChat(Long chatId, Long replyId, EntChatMessage msg, UserDetails userDetails) throws EntityException, AccessException {  
 if(userDetails==null){  
 throw new AccessException(  
 HttpStatus.*UNAUTHORIZED*,  
 "user details was null",  
 "Ошибка проверки данных аккаунта",  
 null  
 );  
 }  
 EntChat chat=rChat.findById(chatId).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "chat with id "+chatId+" not found",  
 "Чат не найден",  
 EntChat.class  
 ));  
 EntUser user=rUser.findByLogin(userDetails.getUsername()).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user "+userDetails.getUsername()+" not found",  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 EntChatUser chatUser=rChatUser.findByChatIdAndUserId(chat,user).orElseThrow(()->new EntityException(  
 HttpStatus.*FORBIDDEN*,  
 "user with id "+user.getId()+" isn't member of chat with id "+chat.getId(),  
 "Нет доступа к этому чату",  
 EntChatUser.class  
 ));  
 EntChatMessage replyTo=null;  
 if(replyId!=null){  
 replyTo=rChatMessage.findById(replyId).orElse(null);  
 }  
 EntChatMessage message=new EntChatMessage(  
 null,  
 msg.getText(),  
 LocalDateTime.*now*(),  
 chat,  
 user,  
 replyTo  
 );  
 rChatMessage.save(message);  
 return ConverterChatMessageToChatMessage.*convertEntityToDTO*(message);  
 }  
}

Листинг 24 является листингом сервиса пользователей, реализующим бизнес-логику для работы с пользователями

Листинг 24 – сервис пользователей

@Service  
public class ServiceUser {  
 private final RepUser rUser;  
 private final RepRole rRole;  
 private final RepGroup rGroup;  
 private final GenPasswordCrypt pwd\_generator=new GenPasswordCrypt();  
  
 @Autowired  
 public ServiceUser(RepUser rUser, RepRole rRole, RepGroup rGroup) {  
 this.rUser = rUser;  
 this.rRole = rRole;  
 this.rGroup = rGroup;  
 }  
  
 public List<DTOUserUpdate> getUsersAll(int page){  
 Pageable pageable = PageRequest.*of*(page, 20);  
 List<EntUser> users= rUser.findAll(pageable).getContent();  
 List<DTOUserUpdate> userList=new ArrayList<>();  
 for(EntUser user:users){  
 userList.add(new DTOUserUpdate(  
 user.getId(),  
 user.getLogin(),  
 user.getFirstname(),  
 user.getLastname(),  
 user.getSurname()  
 ));  
 }  
 return userList;  
 }  
  
 public List<DTOUserUpdate> getUsersDTO(){  
 List<EntUser> users= (List<EntUser>) rUser.findAll();  
 List<DTOUserUpdate> userList=new ArrayList<>();  
 for(EntUser user:users){  
 DTOUserUpdate dtoUser=new DTOUserUpdate(  
 user.getId(),  
 user.getLogin(),  
 user.getFirstname(),  
 user.getLastname(),  
 user.getSurname(),  
 user.getDateofbirth(),  
 user.getEmail(),  
 user.getQwestion().toString(),  
 user.getQwestionanswer(),  
 user.getStudentcard(),  
 (user.getUsergroupId()!=null ? user.getUsergroupId().getName() : null),  
 new ArrayList<>(user.getRoles().stream().map(EntRole::getName).toList())  
 );  
 userList.add(dtoUser);  
 }  
 return userList;  
 }  
  
 public List<EntUser> getUsersByRole(String role) throws EntityException {  
 EntRole userrole=rRole.findByName(role).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "role not found when getting users "+role,  
 "Роль не найдена",  
 EntRole.class  
 ));  
 List<EntRole> roleList=new ArrayList<>();  
 roleList.add(userrole);  
 return rUser.findAllByRolesIn(roleList);  
 }  
  
 public DTOUserUpdate getDTOUserById(Long id) throws EntityException {  
 EntUser user=rUser.findById(id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user not found when getting by id "+id,  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 return new DTOUserUpdate(  
 user.getId(),  
 user.getLogin(),  
 user.getFirstname(),  
 user.getLastname(),  
 user.getSurname(),  
 user.getDateofbirth(),  
 user.getEmail(),  
 user.getQwestion().toString(),  
 user.getQwestionanswer(),  
 user.getStudentcard(),  
 (user.getUsergroupId()!=null ? user.getUsergroupId().getName() : null),  
 new ArrayList<>(user.getRoles().stream().map(EntRole::getName).toList())  
 );  
 }  
  
 public EntUser getUserById(Long id) throws EntityException {  
 return rUser.findById(id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user not found when getting by id "+id,  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 }  
  
 @Transactional  
 public EntUser getUserByUsername(String username) throws EntityException {  
 EntUser user=rUser.findByLogin(username).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user not found when getting by username "+username,  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 user.getRoles().size();  
 return user;  
 }  
  
 public EntUser getUserByLogin(UserDetails userDetails) throws EntityException {  
 return rUser.findByLogin(userDetails.getUsername()).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user not found when getting by username "+userDetails.getUsername(),  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 }  
  
 public EntUser checkUserExistForPWRform(EntUser pwruser) throws EntityException {  
 EntUser user=rUser.findByStudentcard(pwruser.getStudentcard()).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user not found when restoring password",  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 if(user.getQwestionanswer().equals(pwruser.getQwestionanswer())){  
 return user;  
 }  
 return null;  
 }  
  
 @Transactional  
 public EntUser addUser(EntUser newuser, String groupname) throws EntityException {  
 EntUser user=rUser.findByLogin(newuser.getLogin()).orElse(null);  
 if(user!=null){  
 return null;  
 }  
 EntRole role=rRole.findByName("ROLE\_STUDENT").orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "role not found when adding user",  
 "Роль не найдена",  
 EntRole.class  
 ));  
 List<EntRole> roles=new ArrayList<>();  
 roles.add(role);  
 newuser.setRoles(roles);  
 newuser.setPassword(pwd\_generator.generateCryptedPassword(newuser.getPassword()));  
 EntGroup group=rGroup.findByName(groupname).orElse(null);  
 newuser.setUsergroupId(group);  
 rUser.save(newuser);  
 return newuser;  
 }  
  
 @Transactional  
 public EntUser updUser(DTOUserUpdate dtouser, Long id, String groupname, UserDetails userDetails, boolean admin) throws AccessException, EntityException {  
 EntUser user=rUser.findById(id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user not found when updating with id "+id,  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 EntUser newuser= ConverterUserUpdateToUser.*convertDTOtoUser*(dtouser);  
 if(!(Checker.*checkUserIdentity*(userDetails,user,rUser) || (Checker.*isUserHasRole*(userDetails,rUser)))){  
 throw new AccessException(HttpStatus.*FORBIDDEN*,"user identity check failed","Ошибка доступа",userDetails);  
 }  
 if (newuser.getPassword() != null && !admin) {  
 user.setPassword(pwd\_generator.generateCryptedPassword(newuser.getPassword()));  
 } else {  
 user.setPassword(user.getPassword());  
 }  
 user.setLogin(newuser.getLogin());  
 user.setEmail(newuser.getEmail());  
 user.setUsergroupId(rGroup.findByName(groupname).orElse(null));  
 user.setDateofbirth(newuser.getDateofbirth());  
 if(!admin) {  
 user.setQwestion(newuser.getQwestion());  
 user.setQwestionanswer(newuser.getQwestionanswer());  
 }  
 user.setFirstname(newuser.getFirstname());  
 user.setLastname(newuser.getLastname());  
 user.setSurname(newuser.getSurname());  
 user.setStudentcard(newuser.getStudentcard());  
 if(admin){  
 List<EntRole> roleList=rRole.findByNameIn(dtouser.getUserRoles());  
 user.setRoles(roleList);  
 }  
 rUser.save(user);  
 return user;  
 }  
  
 @Transactional  
 public EntUser updUserByPwRestore(EntUser newuser, Long id, UserDetails userDetails) throws AccessException, EntityException {  
 EntUser user=rUser.findById(id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user not found when updating with id "+id,  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 if(!(Checker.*checkUserIdentity*(userDetails,user,rUser) || (Checker.*isUserHasRole*(userDetails,rUser)))){  
 throw new AccessException(HttpStatus.*FORBIDDEN*,"user identity check failed","Ошибка доступа",userDetails);  
 }  
 user.setPassword(pwd\_generator.generateCryptedPassword(newuser.getPassword()));  
 user.setLogin(newuser.getLogin());  
 user.setEmail(newuser.getEmail());  
 user.setRoles(newuser.getRoles());  
 user.setUsergroupId(user.getUsergroupId());  
 user.setDateofbirth(newuser.getDateofbirth());  
 user.setQwestion(newuser.getQwestion());  
 user.setQwestionanswer(newuser.getQwestionanswer());  
 user.setFirstname(newuser.getFirstname());  
 user.setLastname(newuser.getLastname());  
 user.setSurname(newuser.getSurname());  
 user.setStudentcard(newuser.getStudentcard());  
 rUser.save(user);  
 return user;  
 }  
  
 @Transactional  
 public DTOUserUpdate delUserById(Long id,UserDetails userDetails) throws AccessException, EntityException {  
 EntUser user=rUser.findById(id).orElseThrow(()->new EntityException(  
 HttpStatus.*NOT\_FOUND*,  
 "user not found when deleting by id "+id,  
 "Пользователь не найден",  
 EntUser.class  
 ));  
 if(!(Checker.*checkUserIdentity*(userDetails,user,rUser) || (Checker.*isUserHasRole*(userDetails,rUser)))){  
 throw new AccessException(HttpStatus.*FORBIDDEN*,"user identity check failed","Ошибка доступа",userDetails);  
 }  
 rUser.delete(user);  
 return ConverterUserUpdateToUser.*convertUserToDTO*(user);  
 }  
}

**ВЫВОДЫ ПО 3 ГЛАВЕ**

При проектировании системы были описаны архитектура информационной системы, использованы различные технологии и библиотеки, упростившие написание кода и помогающие выполнить различные, требуемые при разработке продукта, операции. Многоуровневая архитектура приложения обеспечивает удобство сопровождения и модульность итоговой системы. Также монолитная архитектура позволяет уменьшить время, затраченное на разработку.

Проектирования инфологической модели помогло определить связи между таблицами и их поля, а СУБД MySQL предоставила необходимые инструменты для работы с данными. Вспомогательное программное обеспечение, используемое при разработке информационной системы, помогло создать функционал системы, соответствующий требованиям из раздела 2.2.3.

Все классы, методы и интерфейсы имеют чёткую и понятную структуру методов и полей, реализуют все необходимые функции и играют огромную роль при масштабировании и наращивании функционала системы. Благодаря использованию языка Java и разделению системы на уровни, получилось добиться повышенной производительности, отказоустойчивости и гибкости настройки.

В дереве проекта используется чёткая и понятная структура хранения директорий и файлов. Все файлы разделены по конкретным признакам и назначению.

Весь спроектированный интерфейс был реализован с учётом имеющегося функционала и потребностей пользователя. Для каждой пользовательской роли в системе был реализован отдельный вариант интерфейса, предоставляющий доступ к функциям системы.

**ЗАКЛЮЧЕНИЕ**

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы была выявлена и проанализирована проблема организации дистанционного обучения.

Исходя из этой проблемы были проанализированы существующие решения образовательных систем, анализ которых выявил и преимущества, и недостатки таких ИС и позволил сформировать требования к разрабатываемой системе, чтобы решить поставленную задачу. Образовательная онлайн-платформа также должна содержать в себе и другие необходимые функции, облегчающие жизнь пользователям.

С помощью методологии IDEF0, были построены диаграммы, дающие понятие о том, как работают образовательные онлайн-платформы существующих решений и как будет работать данная система. Также были сформулированы функциональные и нефункциональные требования к системе по методологии FURPS+.

По итогу проведённой работы было решено разработать информационную систему для решения поставленной цели и задач.

Разработанная информационная система представляет собой WEB-приложение, совмещающее в себе функции обмена заданиями, проведения видеоконференций, ведения электронного журнала, администраторского контроля и участия в текстовых чатах. Также администраторам предоставляется широкий инструментарий для управления всей системой и её пользователями.

Внедрение данной системы позволит учебному заведению самостоятельно проводить точную настройку образовательной онлайн-системы под свои нужды и управлять онлайн-образовательным процессом.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

IntelliJ IDEA. Официальный сайт [Электронный ресурс]. Режим доступа: [https://www.jetbrains.com/idea/]

TADVISER «Онлайн-образование (Рынок России)» [Электронный ресурс]. Режим доступа: [https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%9E%D0%BD%D0%BB%D0%B0%D0%B9%D0%BD-%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5\_%28%D1%80%D1%8B%D0%BD%D0%BE%D0%BA\_%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D0%B8%29#.D0.A0.D0.BE.D1.81.D1.81.D0.B8.D0.B9.D1.81.D0.BA.D0.B8.D0.B9\_.D1.80.D1.8B.D0.BD.D0.BE.D0.BA\_EdTech\_.D0.B2.D1.8B.D1.80.D0.BE.D1.81\_.D0.BD.D0.B0\_21.25\_.D0.B8\_.D0.B4.D0.BE.D1.81.D1.82.D0.B8.D0.B3\_.E2.82.BD149\_.D0.BC.D0.BB.D1.80.D0.B4\_.D0.B2\_2024\_.D0.B3.D0.BE.D0.B4.D1.83]

РБК Тренды «₽51 тыс. готовы потратить россияне на обучение IT и аналитике в 2025 году [Электронный ресурс]. Режим доступа: [https://trends.rbc.ru/trends/education/673c951d9a79477ab92c9f0b]

RgRu «В 2021 году рынок онлайн-образования в России продолжит взрывной рост» [Электронный ресурс]. Режим доступа: [https://rg.ru/2021/01/12/v-2021-godu-rynok-onlajn-obrazovaniia-v-rossii-prodolzhit-vzryvnoj-rost.html]

TADVISER «Онлайн-образование (Рынок России)» [Электронный ресурс]. Режим доступа: [https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%9E%D0%BD%D0%BB%D0%B0%D0%B9%D0%BD-%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5\_%28%D1%80%D1%8B%D0%BD%D0%BE%D0%BA\_%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D0%B8%29#.2A\_.D0.9F.D0.BE.D0.BB.D0.BE.D0.B2.D0.B8.D0.BD.D0.B0\_.D1.81.D1.82.D1.83.D0.B4.D0.B5.D0.BD.D1.82.D0.BE.D0.B2\_.D1.80.D0.BE.D1.81.D1.81.D0.B8.D0.B9.D1.81.D0.BA.D0.B8.D1.85\_.D0.B2.D1.83.D0.B7.D0.BE.D0.B2\_.D0.BE.D0.B1.D1.83.D1.87.D0.B0.D0.B5.D1.82.D1.81.D1.8F\_.D0.BE.D0.BD.D0.BB.D0.B0.D0.B9.D0.BD]

Java Rush. Официальный сайт [Электронный ресурс]. Режим доступа: [https://javarush.com/]

Java documentation. Официальный сайт [Электронный ресурс]. Режим доступа: [https://docs.oracle.com/en/java/index.html]

Microsoft Teams. Официальный сайт [Электронный ресурс]. Режим доступа: [https://teams.microsoft.com/v2/]

Metanit.com. [Электронный ресурс]. Режим доступа: [https://metanit.com/java/]

Skypro wiki [Электронный ресурс]. Режим доступа: [https://sky.pro/wiki/]

Baeldung [Электронный ресурс]. Режим доступа: [https://www.baeldung.com/]

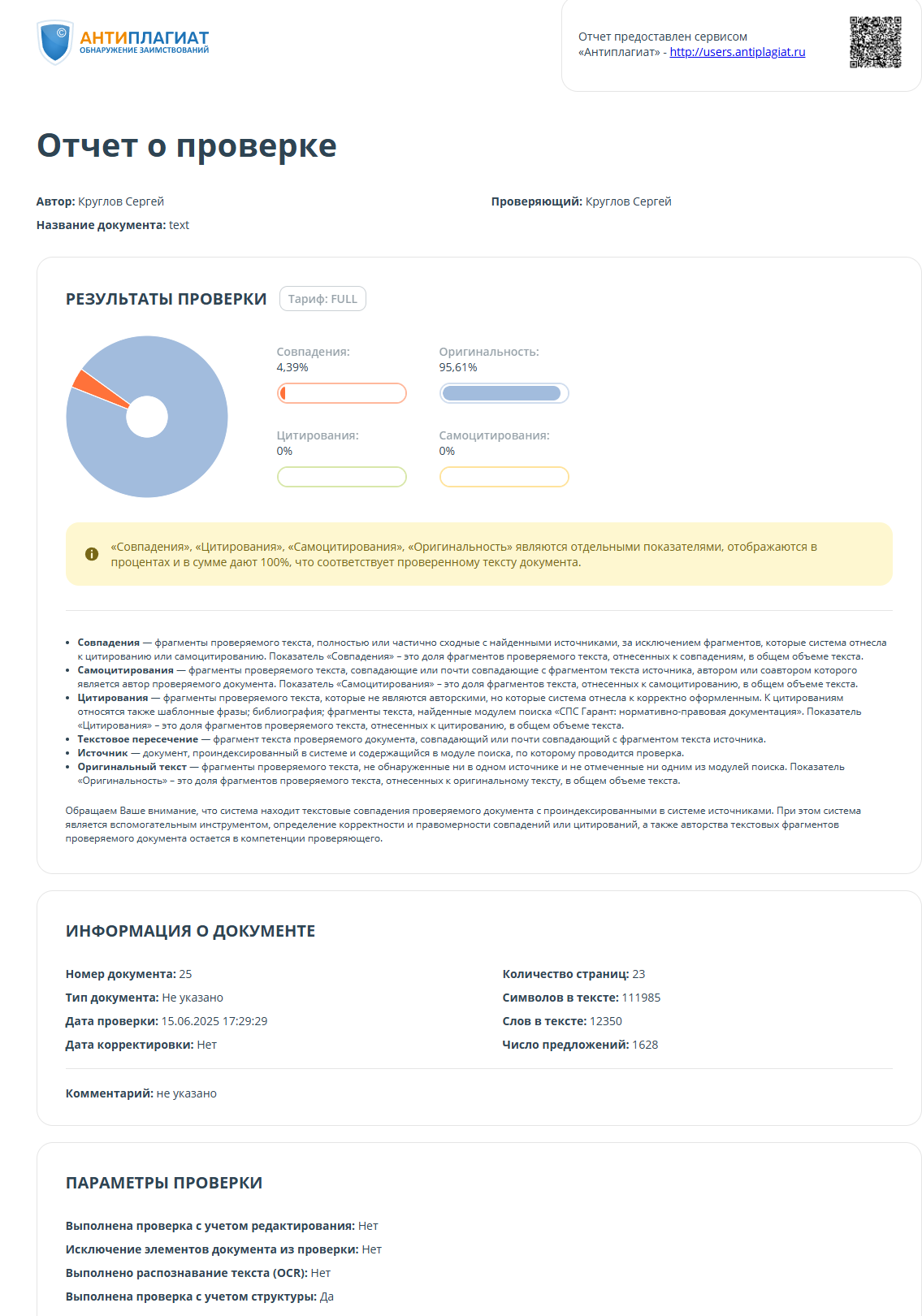
JavaScript.ru. [Электронный ресурс]. Режим доступа: [https://learn.javascript.ru/]

Mozilla [Электронный ресурс]. Режим доступа: [https://developer.mozilla.org/ru/]

MySQL. Официальный сайт [Электронный ресурс]. Режим доступа: [https://dev.mysql.com/doc/]

GlobeNewsWire [Электронный ресурс]. Режим доступа: [https://www.globenewswire.com/news-release/2022/08/24/2504123/0/en/Online-Education-Market-Size-is-projected-to-reach-USD-198-9-Billion-by-2030-growing-at-a-CAGR-of-23-12-Straits-Research.html]

Learning Management System Adoption in Higher Education Using the Extended Technology Acceptance Model [Статья]. Режим доступа: [https://www.researchgate.net/publication/342988531\_Learning\_Management\_System\_Adoption\_in\_Higher\_Education\_Using\_the\_Extended\_Technology\_Acceptance\_Model]





ОТЗЫВ

на выпускную квалификационную работу бакалавра студента группы АУБП-21-1

Уфимцева Егора Александровича

«Проектирование и разработка образовательной онлайн-платформы»

Выпускная квалификационная работа студента группы АУБП-21-1 Е.А. Уфимцева выполнена в соответствии с заданием и содержит все разделы, предусмотренные пунктами задания.

Выпускная квалификационная работа посвящена исследованию, проектированию и разработке онлайн-образовательной платформы. Цель работы – создание системы, которая совмещала бы видеоконференции, подсистему заданий, чаты и электронный журнал, а также автоматизировала ведение электронного журнала и обмен заданиями. Работа содержит 174 страницы, 91 рисунок, 24 листинга программного кода, 16 литературных источников.

В работе выполнено исследование предметной области, спроектирована информацион- ная система образовательной онлайн-платформы, дано техническое обоснование необходимости создания такой системы, сформулированы требования к системе.

Спроектировано, разработано и представлено программное обеспечение, представляющее собой прототип информационной системы образовательной онлайн-платформы с возмож- ностями выдачи и проверки заданий, ведения электронного журнала, проведения видеоконфе- ренций и общения внутри платформы.

Выпускная квалификационная работа состоит из аннотации, введения, трех глав, заключения.

**Существенных замечаний по работе нет. Имеются незначительные замечания по оформлению.**

За время работы над выпускной квалификационной работой Е.А. Уфимцев продемон- стрировал умение самостоятельно ставить и решать научные и практические задачи, прово- дить теоретические исследования и внедрять полученные результаты на практике.

Выполненная работа полностью соответствует требованиям, предъявляемым к выпускной квалификационной работе, студент Е.А. Уфимцев достоин звания дипломированного бакалавра. Рекомендуемая оценка «**отлично**» при соответствующей защите.

|  |  |  |
| --- | --- | --- |
| Научный руководитель, к.т.н., доцент кафедры ИНФ | sif | Е.А. Волкова |
|  |  | 9 июня 2025 г. |